Q-Value Functions for Decentralized POMDPs

Frans A. Oliehoek
Informatics Institute,
University of Amsterdam
Kruislaan 403, 1098 SJ
Amsterdam, The Netherlands
faolieho@science.uva.nl

Nikos Vlassis
Informatics Institute,
University of Amsterdam
Kruislaan 403, 1098 SJ
Amsterdam, The Netherlands
vlassis@science.uva.nl

ABSTRACT

Planning in single-agent models like MDPs and POMDPs can be carried out by resorting to Q-value functions: a (near-) optimal Q-value function is computed in a recursive manner by dynamic programming, and then a policy is extracted from this value function. In this paper we study whether similar Q-value functions can be defined in decentralized POMDP models (Dec-POMDPs), what the cost of computing such value functions is, and how policies can be extracted from such value functions. Using the framework of Bayesian games, we argue that searching for the optimal Q-value function may be as costly as exhaustive policy search. Then we analyze various approximate Q-value functions that allow efficient computation. Finally, we describe a family of algorithms for extracting policies from such Q-value functions.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Performance, Experimentation, Theory

Keywords

Planning under uncertainty, cooperative multiagent systems, decentralized POMDPs

1. INTRODUCTION

In recent years the artificial intelligence community has shown an increasing interest in multiagent systems [18, 15, 17]. For multiagent decision making under uncertainty in particular, frameworks based on Markov decision processes (MDPs) have received considerable attention [3, 5, 8]. In this paper we focus on the decentralized partially observable Markov decision process (Dec-POMDP), a model for multiagent (decentralized) planning in stochastic environments that are only partially observable [2].

Examples of application fields for Dec-POMDPs are cooperative robotics, distributed sensor networks and communication networks. Becker et al. introduced a multi-robot space exploration example in which the agents have to decide on how to proceed in their mission [1]. Emery-Montemerlo et al. considered multi-robot navigation in which a team of agents with noisy sensors have to act to capture a target [4].

For single-agent MDPs various results are known. In particular it is known that an optimal policy π^* can be extracted from the optimal action value (Q-value) function $Q^*(s,a)$, and that the latter can be calculated efficiently. For POMDPs, similar results are available, although finding the optimal solution is harder (PSPACE-complete for finite-horizon problems [12]).

On the other hand, for Dec-POMDPs relatively little is known except that they are provably intractable (NEXP-complete [2]). In particular, an outstanding issue is whether Q-value functions can be defined for Dec-POMDPs just as in MDPs/POMDPs, and whether policies can be extracted from such Q-value functions. Currently most algorithms for planning in Dec-POMDPs are based on some version of policy search [10, 6, 16], and a proper theory for Q-value functions in Dec-POMDPs is still lacking.

In this paper we address the above issue, showing that an optimal Q-function Q^* *can* be defined for a Dec-POMDP, although its computation may be as hard as exhaustive policy search. We also show that given Q^*, an optimal policy can be computed by the solution of a sequence of Bayesian games through time, thereby extending the solution technique of Montemerlo et al. [4] to the exact setting (Section 3). We also analyze three different approximate Q-value functions that can be efficiently computed and can be used in place of Q^* (Section 4). Finally we describe a generic policy search algorithm, a generalization of the MAA* algorithm of Szer et al. [16], that can be used for extracting a policy from an approximate Q-value function (Section 5).

2. THE DEC-POMDP MODEL

In this section we formally introduce the Dec-POMDP model and describe the planning problem. A *decentralized partially observable Markov decision process (Dec-POMDP)* with m agents is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, O \rangle$ where:

- $\mathcal{S}$ is a finite set of states.
- The set $\mathcal{A} = \times_i \mathcal{A}_i$ is the set of *joint actions*, where $\mathcal{A}_i$

AAMAS'07 May 14–18 2007, Honolulu, Hawai'i, USA.

is the set of actions available to agent i. Every time step one joint action $\mathbf{a} = \langle a_1,...,a_m \rangle$ is taken.

- T is the transition function, a mapping from states and joint actions to probability distributions over next states: $T : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{P}(\mathcal{S})$.[1]
- R is the reward function, a mapping from states and joint actions to real numbers: $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$.
- $\mathcal{O} = \times_i \mathcal{O}_i$ is the set of joint observations, with $\mathcal{O}_i$ the set of observations available to agent i. Every time step one joint observation $\mathbf{o} = \langle o_1,...,o_m \rangle$ is received.
- O is the observation function, a mapping from joint actions and successor states to probability distributions over joint observations: $O : \mathcal{A} \times \mathcal{S} \rightarrow \mathcal{P}(\mathcal{O})$.

In a Dec-POMDP, an agent i only knows its own individual actions a_i and observations o_i. The planning problem involves finding the best *policy* for each agent, where a policy is a mapping from the individual histories an agent can observe to its actions. In the models we consider we assume a finite planning *horizon* of h time steps, and an initial 'belief' $b^0 \in \mathcal{P}(\mathcal{S})$; this is the initial state distribution at time $t = 0$.

The *action-observation history for agent i*, $\vec{\theta}_i^t$, is the sequence of actions taken and observations received by agent i until time step t:

$$\vec{\theta}_i^t = \left(a_i^0,o_i^1,a_i^1,...,a_i^{t-1},o_i^t\right). \quad (1)$$

The *joint action-observation history* is a tuple with the action-observation history for all agents $\vec{\theta}^t = \langle \vec{\theta}_1^t,...,\vec{\theta}_m^t \rangle$. The set of all action-observation histories for agent i at time t is denoted $\vec{\Theta}_i$. The *observation history for agent i* is the sequence of observations an agent has received:

$$\vec{o}_i^t = \left(o_i^1,...,o_i^t\right). \quad (2)$$

Similar to action-observation histories, $\vec{\mathbf{o}}^t$ denotes a joint observation history and $\vec{\mathcal{O}}_i$ denotes the set of all observation histories for agent i.

Now we can give a definition for deterministic policies: A *pure* or *deterministic policy*, π_i, for agent i in a Dec-POMDP is a mapping from observation histories to actions, $\pi_i : \vec{\mathcal{O}}_i \rightarrow \mathcal{A}_i$. A pure joint policy π is a tuple containing a pure policy for each agent.

Bernstein et al. [2] have shown that optimally solving a Dec-POMDP is NEXP-complete, implying that any optimal algorithm will be doubly exponential in the horizon. This becomes apparent when realizing that the number of pure joint policies is:

$$O\left(\left(|\mathcal{A}_*|^{\frac{(|\mathcal{O}_*|^h-1)}{|\mathcal{O}_*|-1}}\right)^m\right), \quad (3)$$

where $|\mathcal{A}_*|$ and $|\mathcal{O}_*|$ denote the largest individual action and observation sets.

3. BAYESIAN GAMES AND Q-VALUES

Bayesian games were first used in the context of Dec-POMDPs by Emery-Montemerlo et al. for approximate policy search [4]. Here we show that Bayesian games can also be used for computing an *optimal* policy of the Dec-POMDP. Our main result is that an optimal Dec-POMDP policy can be computed by the solution of a sequence of Bayesian games, if the payoff function of those games coincides with the *optimal* Q-function Q^*. The proof is constructive, in that it provides the means for actually computing Q^*, however this computation may involve a prohibitively large cost.

[1]We use $\mathcal{P}(X)$ to denote the infinite set of probability distributions over the finite set X.

$\vec{\theta}_1^{t=0}$	$\vec{\theta}_2^{t=0}$	()	
		a_2	$\bar{a}_2$
()	a_1	+2.75	−4.1
	$\bar{a}_1$	−0.9	+0.3

$\vec{\theta}_1^{t=1}$	$\vec{\theta}_2^{t=1}$	(a_2,o_2)		$(a_2,\bar{o}_2)$		...
		a_2	$\bar{a}_2$	a_2	$\bar{a}_2$	
(a_1,o_1)	a_1	−0.3	+0.6	−0.6	+4.0	...
	$\bar{a}_1$	−0.6	+2.0	−1.3	+3.6	...
$(a_1,\bar{o}_1)$	a_1	+3.1	+4.4	−1.9	+1.0	...
	$\bar{a}_1$	+1.1	−2.9	+2.0	−0.4	
$(\bar{a}_1,o_1)$	a_1	−0.4	−0.9	−0.5	−1.0	...
	$\bar{a}_1$	−0.9	−4.5	−1.0	+3.5	...
$(\bar{a}_1,\bar{o}_1)$	...	...	...	...	...	...

Figure 1: The Bayesian game for the first and second time step (top: $t = 0$, bottom: $t = 1$). The entries $\vec{\theta}^t$, $\mathbf{a}^t$ are given by the payoff function $Q(\vec{\theta}^t,\mathbf{a}^t)$. Light shaded entries indicate the solutions. Dark entries will not be realized given $\langle a_1,a_2 \rangle$ the solution of the BG for $t = 0$.

3.1 Bayesian games for Dec-POMDPs

At a specific time step t in a Dec-POMDP, the main difficulty in coordinating action selection is presented by the fact that each agent has its own individual action-observation history. This situation can be conveniently modeled by a Bayesian game (BG) [11].

A BG is a normal form game where the players have some private information, which defines their *type*. Formally, a BG is a tuple $\langle \mathcal{A},\Theta,P(\Theta),\langle Q_1,...Q_m \rangle \rangle$, where $\mathcal{A}$ is the set of joint actions, $\Theta = \times_i \Theta_i$ is the set of joint types over which a probability function $P(\Theta)$ is specified, and $Q_i : \Theta \times \mathcal{A} \rightarrow \mathbb{R}$ is the payoff function of agent i. For a BG, we denote a joint policy $\beta = \langle \beta_1,...,\beta_m \rangle$, where the individual policies are mappings from types to actions: $\beta_i : \Theta_i \rightarrow \mathcal{A}_i$. For a BG with identical payoffs, i.e., $\forall_{i,j} \forall_\theta \forall_{\mathbf{a}}\ Q_i(\theta,\mathbf{a}) = Q_j(\theta,\mathbf{a})$, the solution is given by:

$$\beta^* = \arg\max_\beta \sum_{\theta \in \Theta} P(\theta) Q(\theta,\beta(\theta)), \quad (4)$$

where $\beta(\theta) = \langle \beta_1(\theta_1),...,\beta_m(\theta_m) \rangle$ is the joint action specified by β for joint type θ.

At a time step t, one can directly associate the primitives of a Dec-POMDP with those of a BG: the types of agent i correspond to its action-observation histories $\Theta_i \equiv \vec{\Theta}_i^t$, and the probability distribution $P(\vec{\Theta}^t)$ can be computed as the marginal of the joint probability distribution over states and action-observation histories, which can be recursively defined as:

$$P(s^t,\vec{\theta}^t) = \sum_{s^{t-1}} P(\mathbf{o}^t|\mathbf{a}^{t-1},s^t) P(s^t|s^{t-1},\mathbf{a}^{t-1}) P(\mathbf{a}^{t-1}|\vec{\theta}^{t-1}) P(s^{t-1},\vec{\theta}^{t-1}). \quad (5)$$

The probabilities of previous actions $P(\mathbf{a}^{t-1}|\vec{\theta}^{t-1})$ are specified by the policy for time steps 0,...,$t-1$. Finally $Q(\vec{\theta}^t,\mathbf{a})$, the payoff function of the BG, should be naturally defined in accordance with the value function of the planning task. For instance, in [4], $Q(\vec{\theta}^t,\mathbf{a})$ is defined as the Q_{MDP}-value of the underlying MDP (see section 4 for details). Figure 1 shows the Bayesian games for $t = 0$ and $t = 1$ for a fictitious Dec-POMDP with 2 agents.

3.2 The optimal Q-value function Q^*

Here we will show that there is an optimal Q-value function for the Dec-POMDP, denoted Q^*, and that using this Q^* as the payoff function for a BG representing time step t of a Dec-POMDP, results in the optimal policy for that time step. Starting from the expected value of the optimal policy in a Dec-POMDP, we will be able to derive the form of this Q^*.

First, let us define the expected cumulative reward for the optimal joint policy π^* as the summation of $E\left[R^t(\pi^*)\right]$ the expected rewards it yields for each time step:

$$V(\pi^*) = E_{\pi^*}\left[\sum_{t=0}^{h-1} R(t)\right] = \sum_{t=0}^{h-1} \underbrace{\sum_{\vec{\theta}^t} P(\vec{\theta}^t) R(\vec{\theta}^t,\pi^*(\vec{\theta}^t))}_{E[R^t(\pi^*)]}, \tag{6}$$

where

$$R(\vec{\theta}^t,\mathbf{a}^t) = \sum_{s^t} R(s^t,\mathbf{a}^t) P(s^t|\vec{\theta}^t). \tag{7}$$

Again, the probability distributions $P(\vec{\theta}^t)$ and $P(s^t|\vec{\theta}^t)$ in (6) and (7) can be derived from (5). Because we know that there is an optimal pure joint policy π^*, we can write:

$$E\left[R^t(\pi^*)\right] = \sum_{\vec{\theta}^t \text{ s.t. } C(\vec{\theta}^t,\pi^*)=1} P(\vec{\theta}^t) R(\vec{\theta}^t,\pi^*(\vec{\theta}^t)), \tag{8}$$

where $C(\vec{\theta}^t,\pi)$ is a term that filters out the $\vec{\theta}^t$ that are inconsistent with the joint pure policy π:

$$C(\vec{\theta}^t,\pi) = \begin{cases} 1 & , \vec{\theta}^t = \left(\mathbf{o}^0,\pi(\mathbf{o}^0),\mathbf{o}^1,\pi(\mathbf{o}^0,\mathbf{o}^1),...\right)) \\ 0 & , \text{otherwise.} \end{cases} \tag{9}$$

We will also write $\vec{\Theta}^t_\pi \equiv \{\vec{\theta}^t \mid C(\vec{\theta}^t,\pi) = 1\}$. Now, let us define the value starting from time step t:

$$\begin{aligned} V^t(\pi^*) &= E\left[R^t(\pi^*)\right] + V^{t+1}(\pi^*) \\ &= \sum_{\vec{\theta}^t \in \vec{\Theta}^t_{\pi^*}} P(\vec{\theta}^t) R(\vec{\theta}^t,\pi^*(\vec{\theta}^t)) + V^{t+1}(\pi^*). \end{aligned} \tag{10}$$

For the last time step $h-1$ there is no expected future reward, so we get:

$$V^{h-1}(\pi^*) = \sum_{\vec{\theta}^{h-1} \in \vec{\Theta}^{h-1}_{\pi^*}} P(\vec{\theta}^{h-1}) \underbrace{R(\vec{\theta}^{h-1},\pi^*(\vec{\theta}^{h-1}))}_{Q^*(\vec{\theta}^{h-1},\pi^*(\vec{\theta}^{h-1}))}. \tag{11}$$

For time step $h-2$ this becomes:

$$\begin{aligned} V^{h-2}(\pi^*) \equiv\ & E\left[R^{h-2}(\pi^*)\right] + V^{h-1}(\pi^*) = \\ & \sum_{\vec{\theta}^{h-2} \in \vec{\Theta}^{h-2}_{\pi^*}} P(\vec{\theta}^{h-2}) R(\vec{\theta}^{h-2},\pi^*(\vec{\theta}^{h-2})) + \\ & \sum_{\vec{\theta}^{h-1} \in \vec{\Theta}^{h-1}_{\pi^*}} P(\vec{\theta}^{h-1}) Q^*(\vec{\theta}^{h-1},\pi^*(\vec{\theta}^{h-1})). \end{aligned} \tag{12}$$

According to (5) we can factorize $P(\vec{\theta}^{h-1})$. Because we assumed that π^* is a pure joint policy, and the summation is over $\vec{\theta}^{h-1} \in \vec{\Theta}^{h-1}_{\pi^*}$, the action probabilities $P(\mathbf{a}^{t-1}|\vec{\theta}^{t-1})$ from (5) are 1. Therefore we can factorize as follows:

$$P(\vec{\theta}^{h-1}) = P(\vec{\theta}^{h-2}) P(\mathbf{o}^{h-1}|\vec{\theta}^{h-2},\pi^*(\vec{\theta}^{h-2})) \tag{13}$$

Which allows us to rewrite (12) to:

$$V^{h-2}(\pi^*) = \sum_{\vec{\theta}^{h-2} \in \vec{\Theta}^{h-2}_{\pi^*}} P(\vec{\theta}^{h-2}) Q^*(\vec{\theta}^{h-2},\pi^*(\vec{\theta}^{h-2})), \tag{14}$$

with

$$\begin{aligned} Q^*(\vec{\theta}^{h-2},\pi^*(\vec{\theta}^{h-2})) &= R(\vec{\theta}^{h-2},\pi^*(\vec{\theta}^{h-2})) + \\ & \sum_{\mathbf{o}^{h-1}} P(\mathbf{o}^{h-1}|\vec{\theta}^{h-2},\pi^*(\vec{\theta}^{h-2})) Q^*(\vec{\theta}^{h-1},\pi^*(\vec{\theta}^{h-1})). \end{aligned} \tag{15}$$

Reasoning in the same way we can find a generic expression for the expected cumulative future reward starting from time step t, namely:

$$V^t(\pi^*) = \sum_{\vec{\theta}^t \in \vec{\Theta}^t_{\pi^*}} P(\vec{\theta}^t) Q^*(\vec{\theta}^t,\pi^*(\vec{\theta}^t)), \tag{16}$$

with

$$Q^*(\vec{\theta}^t,\mathbf{a}) = R(\vec{\theta}^t,\mathbf{a}) + \sum_{\mathbf{o}^{t+1}} P(\mathbf{o}^{t+1}|\vec{\theta}^t,\mathbf{a}) Q^*(\vec{\theta}^{t+1},\pi^*(\vec{\theta}^{t+1})). \tag{17}$$

Note that, per definition, the optimal Dec-POMDP policy π^* maximizes the expected future reward $V^t(\pi^*)$ specified by (16). Therefore, the optimal policy π^* restricted to time step t, denoted $\pi^{t,*}$, is identical to the optimal policy $\beta^{t,*}$ for the Bayesian game for time step t, if the payoff function of the BG is given by Q^*, that is:

$$\pi^{t,*} \equiv \beta^{t,*} = \arg\max_{\beta^t} \sum_{\vec{\theta}^t \in \vec{\Theta}^t_{\pi^*}} P(\vec{\theta}^t) Q^*(\vec{\theta}^t,\beta^t(\vec{\theta}^t)). \tag{18}$$

3.3 Computing π^* from Q^*

Here we describe how the optimal policy π^* for the Dec-POMDP can be computed by solving Bayesian games with the optimal payoff function Q^* in a single sweep forward through time. Equation (18) tells us that $\pi^{t,*} \equiv \beta^{t,*}$. This means that it is possible to construct the complete optimal Dec-POMDP policy π^*, by computing $\pi^{t,*}$ for all t.

A subtlety is that (18) itself is dependent on the optimal policy, as the summation is over all $\vec{\theta}^t \in \vec{\Theta}^t_{\pi^*} \equiv \{\vec{\theta}^t \mid C(\vec{\theta}^t,\pi^*) = 1\}$. This is resolved by realizing that only the past actions influence which action-observation histories can be reached at time step t. Formally, let ψ^t denote a 'past joint policy', i.e., a joint policy π restricted to time steps

0,...,$t-1$. If we denote the optimal past joint policy by $\psi^{t,*}$, we have that $\vec{\Theta}^t_{\pi^*} = \vec{\Theta}^t_{\psi^{t,*}}$, and therefore that:

$$\beta^{t,*} = \arg\max_{\beta^t} \sum_{\vec{\theta}^t \in \vec{\Theta}^t_{\psi^{t,*}}} P(\vec{\theta}^t) Q^*(\vec{\theta}^t, \beta^t(\vec{\theta}^t)). \quad (19)$$

This can be solved in a forward manner for time steps t = 0,1,2,...,$h-1$, because at every time step $\psi^{t,*} = (\pi^{0,*},...,\pi^{t-1,*})$ will be available: it is specified by $(\beta^{0,*},...,\beta^{t-1,*})$ the solutions of the previously solved BGs.

The above analysis extends the results of Emery-Montemerlo et al. [4] by showing that when using BGs with payoff function Q^* as defined by (17), forward-sweep policy computation produces an exact solution.

3.4 Computing Q^* is non-trivial

As discussed in section 3.3 we can use Q^* to compute π^* via BGs. Earlier, in section 3.2, we have shown that it is easy to compute Q^* given the optimal policy π^* using (17). Here we argue that if π^* is not known, computing Q^* is hard; potentially as hard as finding π^* itself.

For MDPs and POMDPs, the optimal Q-values can be found relatively easy in a sweep backward through time: those for time step t can be found from those for $t+1$ by applying a backup operator. This is possible because there is a single agent that perceives a Markovian signal (i.e., control is centralized). This allows the sole agent to (1) select the optimal action (policy) for the next time step and (2) use the expected value for the found optimal action (policy). For instance, the backup operator for a POMDP is given by:

$$Q^*(b^t,a) = R(b^t,a) + \sum_o P(o|b^t,a) \max_a Q^*(b^{t+1},a),$$

which can be rewritten as a 2-step procedure:

1. $\pi^{t+1,*}(b^{t+1}) = \arg\max_{a'} Q^*(b^{t+1},a')$
2. $Q^*(b^t,a) = R(b^t,a) + \sum_o P(o|b^t,a) Q^*(b^{t+1},\pi^{t+1,*}(b^{t+1}))$.

In the case of Dec-POMDPs, step 2 would correspond to calculating Q^* using (17), as discussed. Step 1, however, would correspond to (19) and therefore is dependent on $\psi^{t+1,*}$ (the optimal policy for time steps 0,...,t).

Put more elaborately, calculating Q^*-values for the last time step $t = h-1$ is easy, as it just involves the immediate expected reward. However, in order to compute Q^* for any $t < h-2$, it is necessary to calculate the Q^* values for $t = h-2$. According to (17), this depends on the policy for the action-observation histories at $t = h-1$, i.e., on $\pi^{h-1,*}$. In turn this joint policy $\pi^{h-1,*} \equiv \beta^{h-1,*}$ depends on the optimal past policy $\psi^{h-1,*}$ because (19) requires $\vec{\Theta}^t_{\psi^{h-1,*}}$. It seems that the only trivial solution to this problem is to evaluate (19) for all possible $\vec{\Theta}^t_{\psi^{h-1}}$, and select the $\vec{\Theta}^t_{\psi^{h-1}}$ and $\beta^{h-1,*}$ that maximize the total expected reward over all h time steps. However, as all possible $\vec{\Theta}^t_{\psi^{h-1}}$ correspond to all ψ^{h-1}, this procedure corresponds to brute force enumeration of all joint policies. Although the above does not constitute a proof, we conjecture that computing Q^* may be as costly as exhaustive policy search.

4. APPROXIMATE VALUE FUNCTIONS

As described in the previous section, finding Q^* is impractical when looking for an optimal policy. In this section we describe three approximate Q-value functions $\widehat{Q}(\vec{\theta}^t,\mathbf{a})$ with less computational cost that can be used instead. Section 3.4 illustrated that backwards calculation of Q^*-values for POMDPs is possible because $\pi^{t+1,*}$ can be calculated solely from the $Q^{*,t+1}$-values. We will show that this is also the case for the approximate Q-value functions discussed here. First we will cover $\mathrm{Q_{MDP}}$, a well-known approximate Q-value function, then we will discuss $\mathrm{Q_{POMDP}}$, and finally we will explain $\mathrm{Q_{BG}}$ which is a new approximate Q-value function.

4.1 $\mathrm{Q_{MDP}}$

$\mathrm{Q_{MDP}}$ was originally proposed to approximately solve POMDPs [9], but has also been applied to Dec-POMDPs [4, 16]. The idea is that Q^* can be approximated using the state-action values $Q_{\mathrm{M}}(s,\mathbf{a})$ found when solving the 'underlying MDP' of a Dec-POMDP: the horizon-h MDP defined by a single agent that takes joint actions $\mathbf{a} \in \mathcal{A}$ and observes the nominal state s, but with the same transition model T and reward model R as the original Dec-POMDP. Solving this underlying MDP can be efficiently done using dynamic programming techniques [13], resulting in the optimal MDP Q-value function:

$$Q^{t,*}_{\mathrm{M}}(s^t,\mathbf{a}) = R(s^t,\mathbf{a}) + \sum_{s^{t+1}} P(s^{t+1}|s^t,\mathbf{a}) \max_{\mathbf{a}} Q^{t+1,*}_{\mathrm{M}}(s^{t+1},\mathbf{a}), \quad (20)$$

where, as in section 3.4, the maximization is an implicit selection of $\pi^{t+1,*}_{\mathrm{M}}$, the optimal MDP policy at the next time step. In order to transform to approximate $\widehat{Q}_{\mathrm{M}}(\vec{\theta}^t,\mathbf{a})$-values to be used in Dec-POMDPs, we compute:

$$\widehat{Q}_{\mathrm{M}}(\vec{\theta}^t,\mathbf{a}) = \sum_{s \in \mathcal{S}} Q^{t,*}_{\mathrm{M}}(s,\mathbf{a}) P(s|\vec{\theta}^t), \quad (21)$$

where $P(s|\vec{\theta}^t)$ can be computed from (5). Combining (20) and (21) and making the selection of $\pi^{t+1,*}_{\mathrm{M}}$ explicit we get:

$$\widehat{Q}_{\mathrm{M}}(\vec{\theta}^t,\mathbf{a}) = R(\vec{\theta}^t,\mathbf{a}) + \sum_{s^{t+1}} P(s^{t+1}|\vec{\theta}^t,\mathbf{a}) \max_{\pi^{t+1}_{\mathrm{M}}(s^{t+1})} Q^{t+1,*}_{\mathrm{M}}(s^{t+1},\pi^{t+1}_{\mathrm{M}}(s^{t+1})), \quad (22)$$

which defines the approximate Q-value function that can be used as payoff function for the various BGs of the Dec-POMDP. Note that $\widehat{Q}_{\mathrm{M}}$ is consistent with the established definition of Q-value functions since it is defined as the expected immediate reward of performing (joint) action $\mathbf{a}$ plus the value of following the optimal policy (in this case the optimal MDP-policy) thereafter.

The cost of computation of $\mathrm{Q_{MDP}}$ is dependent on the cost of evaluation of (22), which is $O(|\mathcal{S}|)$. This evaluation has to be done for all $\vec{\theta}^t$ and $\mathbf{a}$. The total number of joint action-observation histories is $\sum_{t=0}^{h-1} (|\mathcal{A}|\,|\mathcal{O}|)^t = \frac{(|\mathcal{A}||\mathcal{O}|)^h}{(|\mathcal{A}||\mathcal{O}|)-1}$, leading to a total computational cost of[2]:

$$O\left(\frac{|\mathcal{A}| \cdot (|\mathcal{A}|\,|\mathcal{O}|)^h}{(|\mathcal{A}|\,|\mathcal{O}|)-1} \cdot |\mathcal{S}|\right). \quad (23)$$

[2]Dynamic programming to calculate the $Q^t_{\mathrm{M}}(s,\mathbf{a})$-values is done in a separate phase and has a cost of $O(|\mathcal{S}| \cdot h)$.

4.2 Q_{POMDP}

In the previous section we explained that a Dec-POMDP has an 'underlying MDP'. Similarly one can define the 'underlying POMDP' of a Dec-POMDP as the POMDP with the same T,O and R, but in which there is only a single agent that takes joint actions $\mathbf{a} \in \mathcal{A}$ and receives joint observations $\mathbf{o} \in \mathcal{O}$. Q_{POMDP} approximates Q^* using the solution of the underlying POMDP [16, 14].

In particular, the optimal Q_{POMDP} value function for an underlying POMDP satisfies:

$$Q^*_{\mathrm{P}}(b^{\vec{\theta}^t},\mathbf{a}) = R(b^{\vec{\theta}^t},\mathbf{a}) + \sum_{\mathbf{o}^{t+1}} P(\mathbf{o}^{t+1}|b^{\vec{\theta}^t},\mathbf{a}) \max_{\pi^{t+1}_{\mathrm{P}}(b^{\vec{\theta}^{t+1}})} Q^*_{\mathrm{P}}(b^{\vec{\theta}^{t+1}},\pi^{t+1}_{\mathrm{P}}(b^{\vec{\theta}^{t+1}})), \quad (24)$$

where $b^{\vec{\theta}^t}$ is the *joint belief* of the single agent that selects joint actions and receives joint observations at time step t and $b^{\vec{\theta}^{t+1}}$ is the joint belief resulting from $b^{\vec{\theta}^t}$ by action $\mathbf{a}$ and joint observation $\mathbf{o}^{t+1}$. The joint belief corresponds to $P(s|\vec{\theta}^t)$ which can be derived from (5). The maximization in (24) is stated in its explicit form: a maximization over time step $t+1$ POMDP policies. However, it should be clear that this maximization effectively is one over joint actions, as it is conditional on the received joint observation $\mathbf{o}^{t+1}$ and thus the resulting belief $b^{\vec{\theta}^{t+1}}$.

Q^*_{P} can be computed by generating all possible joint beliefs and solving the 'belief MDP'. Generating all possible beliefs is easy: starting with b^0 corresponding to the empty joint action-observation history $\vec{\theta}^{t=0}$, for each $\mathbf{a}$ and $\mathbf{o}$ we calculate the resulting $\vec{\theta}^{t=1}$ and corresponding belief $b^{\vec{\theta}^1}$ and continue recursively. Solving the belief MDP amounts to recursively applying (24).

As there is a one-to-one correspondence between joint beliefs and action-observation histories, we can directly use the computed Q_{POMDP} values as payoffs for the BGs of the Dec-POMDP, that is, we define:

$$\widehat{Q}_{\mathrm{P}}(\vec{\theta}^t,\mathbf{a}) \equiv Q^*_{\mathrm{P}}(b^{\vec{\theta}^t},\mathbf{a}). \quad (25)$$

The cost of calculating Q_{POMDP} can be divided in the cost of calculating the immediate reward for all $\vec{\theta}^t$,$\mathbf{a}$, and the cost of evaluating future reward for all $\vec{\theta}^t$,$\mathbf{a}$, with $t = 0,...,h-2$. The former is identical to the Q_{MDP} case, the latter requires selecting the maximizing joint action for each joint observation, leading to a total cost of:

$$O\left(\frac{|\mathcal{A}| \cdot (|\mathcal{A}|\,|\mathcal{O}|)^h}{(|\mathcal{A}|\,|\mathcal{O}|)-1} \cdot (|\mathcal{S}|+1)\right), \quad (26)$$

which is only slightly worse than the complexity of calculating Q_{MDP}.

4.3 Q_{BG}

Q_{MDP} approximates Q^* by assuming that the state becomes fully observable in the next time step, while Q_{POMDP} assumes that at every time step t the agents know the joint action-observation history $\vec{\theta}^t$. Here we present a new approximate Q-value function, called Q_{BG}, that relaxes the assumptions further: it assumes that the agents know $\vec{\theta}^{t-1}$, the joint action-observation history up to time step $t-1$, and the joint action $\mathbf{a}^{t-1}$ that was taken at the previous time step. This means that the agents are uncertain regarding each other's last observation, which effectively defines a BG for each $\vec{\theta}^{t-1}$,$\mathbf{a}$. Note, that these BGs are different than the BGs used in section 3: the former have types that correspond to single observations, whereas the latter have types that correspond to complete action-observation histories. Hence, the BGs of Q_{BG} are much smaller in size and thus easier to solve. Formally Q_{BG} is defined as:

$$Q^*_{\mathrm{B}}(\vec{\theta}^t,\mathbf{a}) = R(\vec{\theta}^t,\mathbf{a}) + \max_{\beta_{\langle\vec{\theta}^t,\mathbf{a}\rangle}} \sum_{\mathbf{o}^{t+1}} P(\mathbf{o}^{t+1}|\vec{\theta}^t,\mathbf{a}) Q^*_{\mathrm{B}}(\vec{\theta}^{t+1},\beta_{\langle\vec{\theta}^t,\mathbf{a}\rangle}(\mathbf{o}^{t+1})), \quad (27)$$

where $\beta_{\langle\vec{\theta}^t,\mathbf{a}\rangle} = \left\langle \beta_{\langle\vec{\theta}^t,\mathbf{a}\rangle,1}(o_1^{t+1}),...,\beta_{\langle\vec{\theta}^t,\mathbf{a}\rangle,m}(o_m^{t+1}) \right\rangle$ is a tuple of individual policies $\beta_{\langle\vec{\theta}^t,\mathbf{a}\rangle,i} : \mathcal{O}_i \rightarrow \mathcal{A}_i$ for the BG played for $\vec{\theta}^t$,$\mathbf{a}$.

Note that the only difference between (27) and (24) is the position and argument of the maximization operator: (27) maximizes over a (conditional) BG-policy, while the maximization in (24) is effectively over unconditional joint actions.

Q_{BG} also shows a clear relation to the calculation of the optimal payoff function Q^*. Equation (17) tells us that the $Q^*(\vec{\theta}^t,\mathbf{a})$ values can be calculated from $Q^*(\vec{\theta}^{t+1},\mathbf{a})$ and an optimal policy $\pi^*(\vec{\theta}^{t+1})$, and (18) tells us that this $\pi^*(\vec{\theta}^{t+1})$ corresponds to the solution of a BG over action-observation histories of length $t-1$. Analogously, Q_{BG} derives $Q^*_{\mathrm{B}}(\vec{\theta}^t,\mathbf{a})$ from $Q^*_{\mathrm{B}}(\vec{\theta}^{t+1},\mathbf{a})$ and a policy $\beta_{\langle\vec{\theta}^t,\mathbf{a}\rangle}$, but this policy now corresponds to a solution of a BG over observation histories of length 1.

The fictitious Dec-POMDP in figure 1 illustrates the computation of Q_{BG}. The probability distribution $P(\vec{\Theta}^1_{\langle a_1,a_2\rangle})$ over joint action-observation histories that can be reached given $\langle a_1,a_2\rangle$ at $t = 0$ is uniform and the immediate reward for $\langle a_1,a_2\rangle$ is 0. Therefore, we have that $2.75 = 0.25 \cdot 2.0 + 0.25 \cdot 3.6 + 0.25 \cdot 4.4 + 0.25 \cdot 1.0$.

The cost of computing Q_{BG} for all $\vec{\theta}^t$,$\mathbf{a}$ can be split up in the cost of computing the immediate reward and the cost of computing the future reward (solving a BG over the last received observation), leading to a total cost of:

$$O\left(\frac{|\mathcal{A}| \cdot (|\mathcal{A}|\,|\mathcal{O}|)^{h-1}}{(|\mathcal{A}|\,|\mathcal{O}|)-1} \cdot \left[(|\mathcal{S}|\,|\mathcal{A}|\,|\mathcal{O}|) + \left(|\mathcal{A}_*|^{|\mathcal{O}_*|}\right)^m\right]\right). \quad (28)$$

Comparing to the cost of computing Q_{MDP} and Q_{POMDP}, this contains an additional exponential term, but this term does not depend on the horizon of the problem.

5. VALUE-DIRECTED POLICY SEARCH

It is well-known that Q_{MDP} yields an approximate $\widehat{Q}_{\mathrm{M}}$-value function that is guaranteed to be an over-estimation of the optimal Q-value function. This is intuitively clear as Q_{MDP} assumes more information than is actually available, namely, full observability of nominal states. In a similar fashion Q_{POMDP} and Q_{BG} are also upper bounds to the optimal value. Note that Q^* from (17) and $\widehat{Q}_{\mathrm{B}}$ (27) are very similar, both have the form:

Algorithm 1 GMAA*

maxLB := $-\infty$ {the maximum lowerbound}
P := $\mathcal{A}$ {the policy pool}
repeat
  select φ^t from P {typic. according to $\widehat{V}(\varphi^t)$}
  Π := `Next`(φ^t)
  if Π contains a subset of full policies π **then**
    π' := $\arg\max_{\pi\in\Pi} V(\pi)$
    if $V(\pi')$ > maxLB **then**
      maxLB := $V(\pi')$
      $\pi^{\text{best}} := \pi'$
      Prune all $\varphi \in$ P with $\widehat{V}(\varphi) \leq$ maxLB
    end if
    $\Pi = \Pi \setminus \{\pi\}$ {remove full policies}
  end if
  P := P $\setminus\, \varphi^t \cup \Pi$
until P is empty

Algorithm 2 `Next`(φ^t) — MAA*

{Construct the set of all φ^{t+1} consistent with φ^t:}
$\Pi^{t+1} := \left\{\varphi^{t+1} = \langle\varphi_1^{t+1},...,\varphi_m^{t+1}\rangle \mid \varphi_i^{t+1} : \vec{\mathcal{O}}_i^{t+1} \to \mathcal{A}_i\right\}$
{Valuate the policies:} $\forall_{\varphi^{t+1}\in\Pi^{t+1}}$
$\widehat{V}(\varphi^{t+1}) := V^{0\ldots t}(\varphi^{t+1}) + R^{t+1}(\varphi^{t+1}) + \widehat{V}^{(t+2)\ldots h}(\varphi^{t+1})$
return Π^{t+1}

Algorithm 3 `Next`(φ^t) — Emery-Montemerlo et al.

$BG := \left\langle \mathcal{A}, \vec{\Theta}_\pi^{t+1}, P(\vec{\Theta}_\pi^{t+1}), \widehat{Q}^{t+1}\right\rangle$
for all $\beta = \langle\beta_1,...,\beta_m\rangle$ s.t. $\beta_i : \vec{\mathcal{O}}_i^{t+1} \to \mathcal{A}_i$ **do**
  $\widehat{V}^{t+1}(\beta) := \sum_{\vec{\theta}^{t+1}\in\vec{\Theta}_{\varphi^t}^{t+1}} P(\vec{\theta}^{t+1})\widehat{Q}^{t+1}(\vec{\theta}^{t+1},\beta(\vec{\theta}^{t+1}))$
  $\widehat{V}(\varphi^{t+1}) := V^{0\ldots t}(\varphi^{t+1}) + \widehat{V}^{t+1}(\beta)$
end for
return $\arg\max_{\varphi^{t+1}} \widehat{V}(\varphi^{t+1})$

$$Q(\vec{\theta}^t,\mathbf{a}) = R(\vec{\theta}^t,\mathbf{a}) + \sum_{\mathbf{o}^{t+1}} P(\mathbf{o}^{t+1}|\vec{\theta}^t,\mathbf{a})Q(\vec{\theta}^{t+1},\pi^{t+1}(\vec{\theta}^{t+1})). \tag{29}$$

The difference is that $\mathrm{Q_{BG}}$ uses:

$$\arg\max_{\beta_{\langle\vec{\theta}^t,\mathbf{a}\rangle}} \sum_{\mathbf{o}^{t+1}\in\mathcal{O}} P(\mathbf{o}^{t+1}|\vec{\theta}^t,\mathbf{a})Q^*_{\mathrm{B}}(\vec{\theta}^{t+1},\beta_{\langle\vec{\theta}^t,\mathbf{a}\rangle}(\mathbf{o}^{t+1})), \tag{30}$$

as the optimal next time step policy π^{t+1}, while (17) uses the actual optimal joint policy that maximizes (16), that is,

$$\arg\max_{\beta^{t+1}} \sum_{\vec{\theta}^{t+1}\in\vec{\Theta}_{\pi^*}^{t+1}} P(\vec{\theta}^{t+1})Q^*(\vec{\theta}^{t+1},\beta^{t+1}(\vec{\theta}^{t+1})). \tag{31}$$

Although the latter is optimal when considering all $\vec{\theta}^{t+1} \in \vec{\Theta}_{\pi^*}^{t+1}$, it can be sub-optimal when only considering the $\mathbf{o}^{t+1} \in \mathcal{O}$ given $\vec{\theta}^t$,$\mathbf{a}$. Therefore it is clear that the value specified by (27) is an upper bound to that specified by (17). Also by just considering (24) and (27), it is clear that $\mathrm{Q_{POMDP}}$ is an upper bound to $\mathrm{Q_{BG}}$ and thus to Q^*.

The above implies that the approximate Q-value functions described in section 4 can be used as *admissible heuristics* in A^*-like policy search methods, as we describe in the rest of this section. In particular, we will describe a general heuristic policy search framework which we will call Generalized MAA* (GMAA*). GMAA* generalizes Szer et al.'s MAA* [16] by concretizing two different procedures that are implicit in MAA*: (1) iterating over a pool of joint policies, pruning this pool whenever possible and (2) finding some new joint policies given a previous policy. The former procedure is the core of GMAA* while the second procedure, which we will refer to as `Next`, can be performed in many ways. The original MAA* can be seen as an instance of the generalized case with a particular `Next`-operator, namely that shown in algorithm 2. First, we will briefly explain the original MAA* algorithm, then we will describe how the approach of Emery-Montemerlo et al. [4], can be interpreted within the GMAA* framework.

MAA* works on partially specified policies. A 'full' joint policy π specifies actions at h time steps $t = 0,...,h-1$, whereas a 'partial' policy $\varphi^t = \langle\pi^0,...,\pi^t\rangle$ specifies actions for the first $t+1$ time steps. It is possible to compute a heuristic value for a partial policy φ^t by evaluating its actual expected reward for time steps 0,...,t plus a heuristic value for the remaining time steps:

$$\widehat{V}(\varphi^t) = V^{0\ldots t}(\varphi^t) + \widehat{V}^{(t+1)\ldots h-1}(\varphi^t).$$

By using admissible heuristics $\widehat{V}^{(t+1)\ldots h-1}(\varphi^t)$, the function $\widehat{V}(\varphi^t)$ is also guaranteed to be an admissible heuristic. This allows an A^*-like search which is described by algorithms 1 and 2 and can be summarized as follows.

The policy pool P is initialized with the set of joint actions $\mathcal{A}$ (this is the set of all φ^0) and the maximum lower bound found so far is set to $-\infty$. Then the partial policy with the highest heuristic value is selected and all φ^{t+1} consistent with φ^t are generated by the `Next` operator of algorithm 2. When `Next` returns one or more full policies π, $\widehat{V}(\pi) = V(\pi)$ provides a lower bound for the optimal policy, which can then be used to prune the search space. This process is repeated until the policy pool is empty, at which point we know that we have found the optimal policy.

The approach of Emery-Montemerlo et al. [4] is described by algorithms 1 and 3 jointly. Given a partial joint policy φ^t, the `Next` operator now constructs and solves a BG for time step $t+1$. Because `Next` in algorithm 3 only returns the best policy, P will never contain more than 1 joint policy and the whole search process reduces to solving BGs for time steps 0,...,$h-1$ as described in [4].[3]

Also note that the two different `Next` operators of algorithms 2 and 3 are closely related: line 2 and 4 in the respective algorithms specify the same value if:

$$\sum_{\vec{\theta}^{t+1}\in\vec{\Theta}_\pi^{t+1}} P(\vec{\theta}^{t+1})\widehat{Q}^{t+1}(\vec{\theta}^{t+1},\beta(\vec{\theta}^{t+1})) = R^{t+1}(\pi^{t+1}) + \widehat{V}^{(t+2)\ldots h}(\pi^{t+1}) \tag{32}$$

It turns out that these are identical when the used heuristic is of the form:

$$\widehat{Q}^{t+1}(\vec{\theta}^{t+1},\pi^{t+1}(\vec{\theta}^{t+1})) = R(\vec{\theta}^{t+1},\pi^{t+1}(\vec{\theta}^{t+1})) + \sum_{\mathbf{o}^{t+2}} P(\mathbf{o}^{t+2}|\vec{\theta}^{t+1},\pi^{t+1})\widehat{V}^{(t+2)\ldots h}(\vec{\theta}^{t+2}), \tag{33}$$

[3] In fact in [4] smaller BGs are created by discarding or clustering low probability action-observation histories, and the BGs are approximately solved by alternating maximization.

	$h=1$	$h=2$	$h=3$	$h=4$
V^*	-2.0	-4.0	$+5.191$	$+4.803$
Q_{MDP}	9	252	105,228	3.177e9
	0	9	249	105,209
	$\leq$ 0.01s	$\leq$ 0.01s	0.41s (0.02s)	4.133e5s (0.96s)
Q_{POMDP}	9	90	6,651	5.597e8
	0	9	89	39,362
	$\leq$ 0.01s	$\leq$ 0.01s	0.06s (0.03s)	10,571s (1.05s)
Q_{BG}	9	9	6,651	3.013e8
	0	9	89	6,649
	$\leq$ 0.01s	$\leq$ 0.01s	0.09s (0.06s)	5,299s (3.41s)

Table 1: MAA* results for different heuristics. The optimal policy was found using all three heuristics. Shown are for each heuristic, the number of joint policies evaluated, the maximum size of the policy pool and the computation time including the time to calculate the heuristic (indicated between brackets).

which is the case for all the Q-value functions we discussed in this paper. As a result, when using these, algorithms 2 and 3 calculate identical heuristic values for the same next time step joint policies; the sole difference in this case is that the latter returns only the joint policy with the highest heuristic value.

The GMAA* framework also allows other choices. For example, the `Next` operator could return the k best joint policies, or construct constrained BGs such that the β (and thus, φ^{t+1}) in algorithm 3 are chosen from a constrained class. It is also possible to substitute `Next` by a more general operator without the requirement to return next time step policies, thus integrating in the GMAA* framework other algorithms like JESP [10].

6. RESULTS

We tested the different heuristics discussed in section (4) on the decentralized tiger (Dec-Tiger) test problem, which was introduced by Nair et al. [10] and originates from the (single agent) tiger problem [7]. It concerns two agents that are standing in a hallway with two doors. Behind one of the doors is a tiger, behind the other a treasure. Therefore there are two states, the tiger is behind the left door (s_l) or behind the right door (s_r). Both agents have three actions at their disposal: open the left door (OL), open the right door (OR) and listen (Li), and can only receive 2 observations: they hear the tiger in the left or in the right room.

At $t=0$ the state is s_l or s_r with 50% probability. Only when both agents perform Li, the state remains unchanged. In all other cases the state is reset to s_l or s_r with 50% probability. Opening the correct door yields a reward and opening the wrong door results in a penalty. Listening has a minor cost of -2. Acting jointly increases the reward and reduces the penalty. For a more elaborate description and the exact transition-, observation- and reward function we refer to [10].

In order to get a feeling for the differences between the heuristics, we generated all possible $\vec{\theta}^t$ and the corresponding $P(s_l|\vec{\theta}^t)$ for the $h=4$ Dec-Tiger problem. For each of these, the maximal $Q(\vec{\theta}^t,\mathbf{a})$-value is plotted in figure 2. The figure clearly shows that $Q_{BG} \leq Q_{POMDP} \leq Q_{MDP}$.

Next, we optimally solved the Dec-Tiger problem using MAA* for $h=1,...,4$ using the three heuristics. Table 1

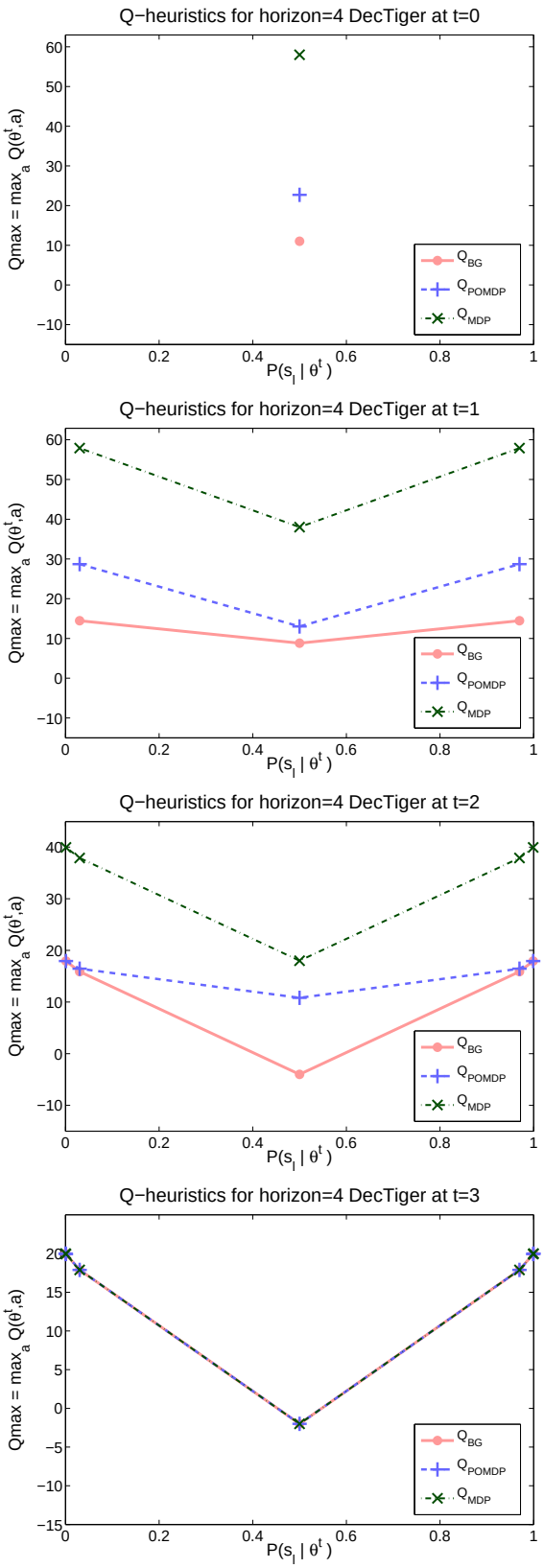

Figure 2: Q-values for horizon 4 Dec-Tiger. For each $\vec{\theta}^t$, corresponding to some $P(s_l|\vec{\theta}^t)$, the maximal $Q(\vec{\theta}^t,\mathbf{a})$-value is plotted.

summarizes the results. Shown are the value of the optimal policy, and for each heuristic: the number of (partial) joint policies evaluated, the maximum size of the policy pool and the computation time needed to calculate π^*. Clearly Q_{MDP} is outperformed by the other heuristics; it uses more time for $h = 1,...,4$. Another interesting point to notice is that while Q_{POMDP} performs better than Q_{BG} for $h = 3$, because the cost of calculating this heuristic is a significant part of the total time in this case, Q_{BG} clearly performs better for $h = 4$ as this cost becomes negligible with respect to the total time.

7. CONCLUSIONS

In this work we presented a value based characterization of the optimal policy in a Dec-POMDP, thus deriving an equation for the optimal $Q^*(\vec{\theta}^t,\mathbf{a})$-function and showed that it is possible to construct the optimal policy π^* by solving BGs for time steps $t = 0,...,h-1$ which use $Q^*(\vec{\theta}^t,\mathbf{a})$ as the payoff function.

We analyzed the problem of finding Q^* and argued that computing this Q^* may be as difficult as finding the optimal solution to the Dec-POMDP itself. We focused on approximate Q-value functions that can be more efficiently calculated and discussed how they relate to Q^*. We covered Q_{MDP} and Q_{POMDP} and proposed Q_{BG}, a new approximate Q-value function that gives a tighter upper bound to Q^*.

Additionally we showed how these heuristics can be applied in a generalized policy search method, thereby unifying recent approximate Dec-POMDP solution techniques [4, 16]. Finally we performed an empirical evaluation of the discussed heuristics showing a clear benefit of using tighter heuristics.

There are a couple of directions for future research. One is to further generalize GMAA*, by defining other `Next` operators as discussed at the end of section 5, with the hope that the resulting algorithms will be able to scale to larger problems. A different direction is to try to achieve tighter heuristic bounds of Q_{BG} by assuming uncertainty over the last k observations instead of only the last observation (with additional computational cost). Another important research direction research is to establish bounds on the performance and learning curves of GMAA* in combination with different `Next` operators and heuristics.

Acknowledgments

We would like to thank Matthijs Spaan for all the fruitful discussions. The research reported here is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

8. REFERENCES

[1] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research (JAIR)*, 22:423–455, December 2004.

[2] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.*, 27(4):819–840, 2002.

[3] C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *TARK '96: Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*, pages 195–210, 1996.

[4] R. Emery-Montemerlo, G. Gordon, J. Schneider, and S. Thrun. Approximate solutions for partially observable stochastic games with common payoffs. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 136–143, 2004.

[5] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14*, pages 1523–1530, 2002.

[6] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, pages 709–715, 2004.

[7] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998.

[8] J. R. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.

[9] M. Littman, A. Cassandra, and L. Kaelbling. Learning policies for partially observable environments: Scaling up. In *International Conference on Machine Learning*, pages 362–370, 1995.

[10] R. Nair, M. Tambe, M. Yokoo, D. V. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 705–711, 2003.

[11] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, July 1994.

[12] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–451, 1987.

[13] M. L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

[14] M. Roth, R. Simmons, and M. Veloso. Reasoning about joint beliefs for execution-time communication decisions. In *Proc. of Int. Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 786–793, 2005.

[15] P. Stone and M. Veloso. Multiagent systems: a survey from a machine learning perspective. *Autonomous Robots*, 8(3), 2000.

[16] D. Szer, F. Charpillet, and S. Zilberstein. MAA*: A heuristic search algorithm for solving decentralized POMDPs. In *Proc. of the Twenty First Conference on Uncertainty in Artificial Intelligence*, 2005.

[17] N. Vlassis. A concise introduction to multiagent systems and distributed AI. Informatics Institute, University of Amsterdam, Sept. 2003.

[18] G. Weiss, editor. *Multiagent Systems: a Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.