Decentralized POMDPs

Frans A. Oliehoek

Abstract This chapter presents an overview of the decentralized POMDP (Dec-POMDP) framework. In a Dec-POMDP, a team of agents collaborates to maximize a global reward based on local information only. This means that agents do not observe a Markovian signal during execution and therefore the agents' individual policies map from histories to actions. Searching for an optimal joint policy is an extremely hard problem: it is NEXP-complete. This suggests, assuming NEXP≠EXP, that any optimal solution method will require doubly exponential time in the worst case. This chapter focuses on planning for Dec-POMDPs over a finite horizon. It covers the forward heuristic search approach to solving Dec-POMDPs, as well as the backward dynamic programming approach. Also, it discusses how these relate to the optimal Q-value function of a Dec-POMDP. Finally, it provides pointers to other solution methods and further related topics.

1 Introduction

Previous chapters generalized decision making to multiple agents (Chapter **??**) and to acting under state uncertainty as in POMDPs (Chapter **??**). This chapter generalizes further by considering situations with both state uncertainty and multiple agents. In particular, it focuses on teams of *collaborative* agents: the agents share a single objective. Such settings can be formalized by the framework of *decentralized POMDPs (Dec-POMDPs)* (Bernstein et al, 2002) or the roughly equivalent *multi-agent team decision problem* (Pynadath and Tambe, 2002). The basic idea of this model is illustrated in Figure 1, which depicts the two-agent case. At each stage, the agents independently take an action. The environment undergoes a state transition and generates

CSAIL, Massachusetts Institute of Technology, e-mail: fao@csail.mit.edu

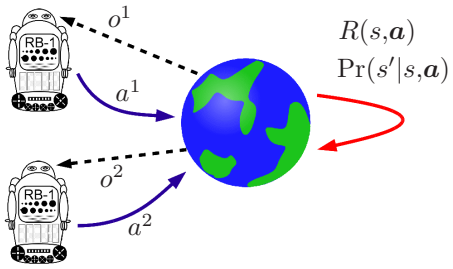


Fig. 1: Execution of a decentralized POMDP.

a reward depending on the state and the actions of both agents. Finally, each agent receives an individual observation of the new state.

This framework allows modeling important real-world tasks for which the models in the previous chapters do not suffice. An example of such a task is load balancing among queues (Cogill et al, 2004). Each agent represents a processing unit with a queue that has to decide whether to accept new jobs or pass them to another queue, based only on the local observations of its own queue size and that of immediate neighbors. Another important application area for Dec-POMDPs is communication networks. For instance, consider a packet routing task in which agents are routers that have to decide at each time step to which neighbor to send each packet in order to minimize the average transfer time of packets (Peshkin, 2001). An application domain that receives much attention in the Dec-POMDP community is that of sensor networks (Nair et al, 2005; Varakantham et al, 2007; Kumar and Zilberstein, 2009). Other areas of interests are teams of robotic agents (Becker et al, 2004b; Emery-Montemerlo et al, 2005; Seuken and Zilberstein, 2007a) and crisis management (Nair et al, 2003a,b; Paquet et al, 2005).

Most research on multi-agent systems under partial observability is relatively recent and has focused almost exclusively on planning—settings where the model of the environment is given—rather than the full reinforcement learning (RL) setting. This chapter also focuses exclusively on planning. Some pointers to RL approaches are given at the end of the chapter.

A common assumption is that planning takes place in an *off-line* phase, after which the plans are executed in an *on-line* phase. This on-line phase is completely *decentralized* as shown in Figure 1: each agent receives its individual part of the joint policy found in the planning phase[1] and its individual history of actions and observations. The off-line planning phase, however, is *centralized.* We assume a single computer that computes the joint plan and

[1] In some cases it is assumed that the agents are given the joint policy. This enables the computation of a *joint belief* from broadcast local observations (see Section 5.4).

subsequently distributes it to the agents (who then merely execute the plan on-line).[2]

2 The Decentralized POMDP Framework

In this section we more formally introduce the Dec-POMDP model. We start by giving a mathematical definition of its components.

Definition 1 (Dec-POMDP). A *decentralized partially observable Markov decision process* is defined as a tuple $\langle \mathcal{D}, S, \boldsymbol{A}, T, R, \boldsymbol{O}, O, h, I \rangle$, where

- $\mathcal{D} = \{1, \ldots, n\}$ is the set of n agents.
- S is a finite set of states s in which the environment can be.
- $\boldsymbol{A}$ is the finite set of joint actions.
- T is the transition probability function.
- R is the immediate reward function.
- $\boldsymbol{O}$ is the finite set of joint observations.
- O is the observation probability function.
- h is the horizon of the problem.
- $I \in \mathcal{P}(\mathcal{S})$, is the initial state distribution at stage $t = 0$.

The Dec-POMDP model extends single-agent POMDP models by considering *joint* actions and observations. In particular, $\boldsymbol{A} = \times_{i \in \mathcal{D}} A^i$ is the set of *joint actions*. Here, A^i is the set of actions available to agent i, which can be different for each agent. Every time step, one joint action $\boldsymbol{a} = \langle a^1, \ldots, a^n \rangle$ is taken. How this joint action influences the environment is described by the transition function T, which specifies $\Pr(s' | s, \boldsymbol{a})$. In a Dec-POMDP, agents only know their own individual action; they do not observe each other's actions. Similar to the set of joint actions, $\boldsymbol{O} = \times_{i \in \mathcal{D}} O^i$ is the set of joint observations, where O^i is a set of observations available to agent i. Every time step the environment emits one joint observation $\boldsymbol{o} = \langle o^1, \ldots, o^n \rangle$ from which each agent i only observes its own component o^i. The observation function O specifies the probabilities $\Pr(\boldsymbol{o} | \boldsymbol{a}, s')$ of joint observations. Figure 2 further illustrates the dynamics of the Dec-POMDP model.

During execution, the agents are assumed to act based on their individual observations only and no additional communication is assumed. This does not mean that Dec-POMDPs cannot model settings which concern communication. For instance, if one agent has an action "mark blackboard" and the other agent has an observation "marked blackboard", the agents have a mechanism of communication through the state of the environment. However, rather than making this communication explicit, we say that the Dec-POMDP can model communication implicitly through the actions, states and observations. This

[2] Alternatively, each agent runs the same planning algorithm in parallel.

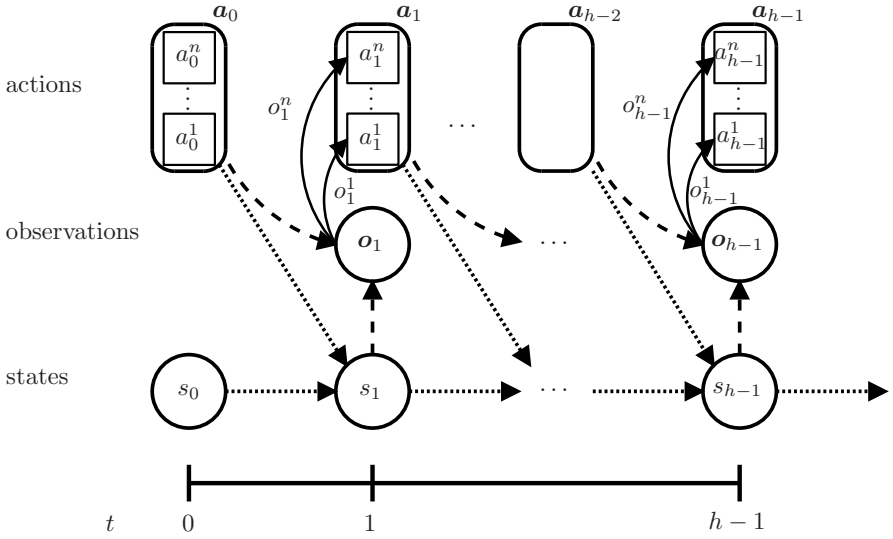


Fig. 2: A more detailed illustration of the dynamics of a Dec-POMDP. At every stage the environment is in a particular state. This state emits a joint observation according to the observation model (dashed arrows) from which each agent observes its individual component (indicated by solid arrows). Then each agent selects an action, together forming the joint action, which leads to a state transition according to the transition model (dotted arrows).

means that in a Dec-POMDP, communication has no special semantics. Section 5.4 further elaborates on communication in Dec-POMDPs.

This chapter focuses on planning over a finite horizon, for which the (undiscounted) *expected cumulative reward* is the commonly used optimality criterion. The planning problem thus amounts to finding a tuple of policies, called a *joint policy* that maximizes the expected cumulative reward.

We will consider the decentralized tiger (Dec-Tiger) problem —a frequently used Dec-POMDP benchmark introduced by Nair et al (2003c)—as an example. 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 3 actions at their disposal: open the left door (a_{OL}), open the right door (a_{OR}) and listen (a_{Li}). They cannot observe each other's actions. In fact, they can only receive 2 observations: either they hear the tiger make a sound from behind the left (o_{HL}) or right (o_{HR}) door.

At $t = 0$ the state is s_l or s_r with probability 0.5. As long as no agent opens a door, the state does not change. When a door is opened, the state resets to s_l or s_r with probability 0.5. The observation probabilities are independent and

identical for both agents. For instance, when the state is s_l and both perform action a_{Li}, each agent has a 85% chance of observing o_{HL}, and the probability that both hear the tiger behind the left door is $0.85 \times 0.85 = 0.72$. When one of the agents opens the door to the treasure they receive a positive reward ($+9$), while they receive a penalty for opening the wrong door (-101). When opening the wrong door jointly, the penalty is less severe (-50). Opening the correct door jointly leads to a higher reward ($+20$). The full transition, observation and reward models are listed by Nair et al (2003c).

Note that, when the wrong door is opened by one or both agents, they are attacked by the tiger and receive a penalty. However, neither of the agents observe this attack nor the penalty (remember, the only possible observations are o_{HL} and o_{HR}) and the episode continues. Intuitively, an optimal joint policy for Dec-Tiger should specify that the agents listen until they are certain enough to open one of the doors. At the same time, the policy should be 'as coordinated' as possible, i.e., maximize the probability of acting jointly.

3 Histories and Policies

In an MDP, the agent uses a policy that maps states to actions. In selecting its action, it can ignore the history because of the Markov property. In a POMDP, the agent can no longer observe the state, but it can compute a belief b that summarizes the history; it is also a Markovian signal. In a Dec-POMDP, however, during execution each agent will only have access to its *individual* actions and observations and there is no method known to summarize this individual history. It is not possible to maintain and update an individual belief in the same way as in a POMDP, because the transition and observation function are specified in terms of joint actions and observations.[3]

This means that in a Dec-POMDP *the agents do not have access to a Markovian signal during execution.* The consequence of this is that planning for Dec-POMDPs involves searching the space of tuples of individual Dec-POMDP policies that map full-length individual histories to actions. We will see later that this also means that solving Dec-POMDPs is even harder than solving POMDPs.

[3] Different forms of beliefs for Dec-POMDP-like settings have been considered Nair et al (2003c); Hansen et al (2004); Oliehoek et al (2009); Zettlemoyer et al (2009). These are not specified over only states, but also specify probabilities over histories/policies/types/beliefs of the other agents. The key point is that from an individual agent's perspective just knowing a probability distribution over states is insufficient; it also needs to predict what actions the other agents will take.

3.1 Histories

First, we define histories of observations, actions and both.

Definition 2 (Action-observation history). The *action-observation history (AOH) for agent i,* $\bar{\theta}^i$, is the sequence of actions taken by and observations received by agent i. At a specific time step t, this is

$$\bar{\theta}^i_t = \left(a^i_0, o^i_1, \ldots, a^i_{t-1}, o^i_t\right).$$

The *joint action-observation history* $\bar{\boldsymbol{\theta}}_t = \langle \bar{\theta}^1_t, \ldots, \bar{\theta}^n_t \rangle$ specifies the AOH for all agents. Agent i's set of possible AOHs at time t is $\bar{\Theta}^i_t$. The set of AOHs possible for all stages for agent i is $\bar{\Theta}^i$ and $\bar{\theta}^i$ denotes an AOH from this set.[4] Finally the set of all possible *joint* AOHs $\bar{\boldsymbol{\theta}}$ is denoted $\bar{\boldsymbol{\Theta}}$. At $t = 0$, the (joint) AOH is empty $\bar{\boldsymbol{\theta}}_0 = ()$.

Definition 3 (Observation history). The *observation history (OH) for agent i,* $\bar{o}^i$, is defined as the sequence of observations an agent has received. At a specific time step t, this is:

$$\bar{o}^i_t = \left(o^i_1, \ldots, o^i_t\right).$$

The *joint observation history,* is the OH for all agents: $\bar{\boldsymbol{o}}_t = \langle \bar{o}^1_t, \ldots, \bar{o}^n_t \rangle$. The set of observation histories for agent i at time t is denoted $\bar{O}^i_t$. Similar to the notation for action-observation histories, we also use $\bar{o}^i \in \bar{O}^i$ and $\bar{\boldsymbol{o}} \in \bar{\boldsymbol{O}}$.

Definition 4 (Action history). The *action history (AH) for agent i,* $\bar{a}^i$, is the sequence of actions an agent has performed:

$$\bar{a}^i_t = \left(a^i_0, a^i_1, \ldots, a^i_{t-1}\right).$$

Notation for joint action histories and sets are analogous to those for observation histories. Finally, note that a (joint) AOH consists of a (joint) action- and a (joint) observation history: $\bar{\boldsymbol{\theta}}_t = \langle \bar{\boldsymbol{o}}_t, \bar{\boldsymbol{a}}_t \rangle$.

3.2 Policies

A policy π^i for an agent i maps from histories to actions. In the general case, these histories are AOHs, since they contain all information an agent has. The number of AOHs grows exponentially with the horizon of the problem: At time step t, there are $\left(\left|A^i\right| \times \left|O^i\right|\right)^t$ possible AOHs for agent i. A policy π^i assigns an action to each of these histories. As a result, the number of possible policies π^i is doubly exponential in the horizon.

[4] In a particular Dec-POMDP, it may be the case that not all of these histories can actually be realized, because of the probabilities specified by the transition and observation model.

It is possible to reduce the number of policies under consideration by realizing that many policies of the form considered above specify the same behavior. This is illustrated by the left side of Figure 3: under a deterministic policy only a subset of possible action-observation histories can be reached. Policies that only differ with respect to an AOH that can never be reached, manifest the same behavior. The consequence is that in order to specify a deterministic policy, the observation history suffices: when an agent selects actions deterministically, it will be able to infer what action it took from only the observation history. This means that a deterministic policy can conveniently be represented as a tree, as illustrated by the right side of Figure 3.

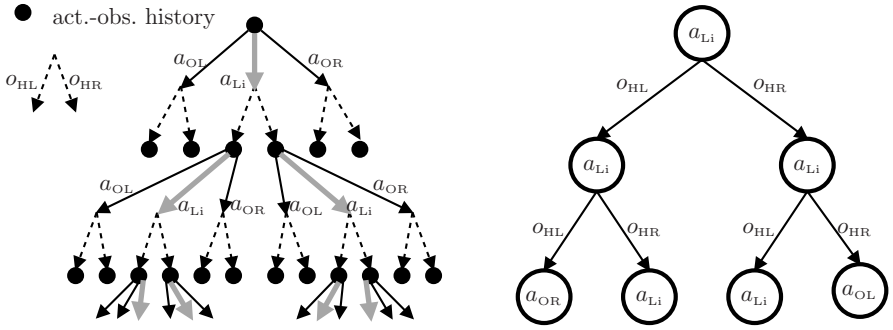


Fig. 3: A deterministic policy can be represented as a tree. Left: a tree of action-observation histories $\bar{\theta}^i$ for one of the agents from the Dec-Tiger problem. A deterministic policy π^i is highlighted. Clearly shown is that π^i only reaches a subset of histories $\bar{\theta}^i$. ($\bar{\theta}^i$ that are not reached are not further expanded.) Right: The same policy can be shown in a simplified policy tree. When both agents execute this policy in the $h = 3$ Dec-Tiger problem, the joint policy is optimal.

Definition 5. A *deterministic policy* π^i for agent i is a mapping from observation histories to actions, $\pi^i : \bar{O}^i \rightarrow A^i$.

For a deterministic policy, $\pi^i(\bar{\theta}^i)$ denotes the action that it specifies for the observation history contained in $\bar{\theta}^i$. For instance, let $\bar{\theta}^i = \langle \bar{o}^i, \bar{a}^i \rangle$, then $\pi^i(\bar{\theta}^i) \triangleq \pi^i(\bar{o}^i)$. We use $\boldsymbol{\pi} = \langle \pi^1, ..., \pi^n \rangle$ to denote a *joint policy.* We say that a deterministic joint policy is an *induced mapping* from joint observation histories to joint actions $\boldsymbol{\pi} : \bar{\boldsymbol{O}} \rightarrow \boldsymbol{A}$. That is, the mapping is induced by individual policies π^i that make up the joint policy. Note, however, that only a subset of possible mappings $f : \bar{\boldsymbol{O}} \rightarrow \boldsymbol{A}$ correspond to valid joint policies: when f does not specify the same individual action for each $\bar{o}^i$ of each agent i, it will not be possible to execute f in a decentralized manner. That is, such a policy is *centralized*: it would describe that an agent should base its choice

of action on the *joint* history. However, during execution it will only be able to observe its *individual* history, not the joint history.

Agents can also execute stochastic policies, but we restrict our attention to deterministic policies without sacrificing optimality, since a finite-horizon Dec-POMDP has at least one optimal pure joint policy (Oliehoek et al, 2008b).

3.3 Structure in Policies

Policies specify actions for all stages of the Dec-POMDP. A common way to represent the temporal structure in a policy is to split it into *decision rules* δ^i that specify the policy for each stage. An individual policy is then represented as a sequence of decision rules $\pi^i = (\delta^i_0, \ldots, \delta^i_{h-1})$. In case of a deterministic policy, the form of the decision rule for stage t is a mapping from length-t observation histories to actions $\delta^i_t : \bar{O}^i_t \to A^i$. In the more general case its domain is the set of AOHs $\delta^i_t : \bar{\Theta}^i_t \to A^i$. A joint decision rule $\boldsymbol{\delta}_t = \langle \delta^1_t, \ldots, \delta^n_t \rangle$ specifies a decision rule for each agent.

We will also consider policies that are partially specified with respect to time. Formally, $\boldsymbol{\varphi}_t = (\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{t-1})$ denotes the *past joint policy* at stage t, which is a partial joint policy specified for stages 0,...,$t-1$. By appending a joint decision rule for stage t, we can 'grow' such a past joint policy.

Definition 6 (Policy concatenation). We write

$$\boldsymbol{\varphi}_{t+1} = (\boldsymbol{\delta}_0, \ldots, \boldsymbol{\delta}_{t-1}, \boldsymbol{\delta}_t) = \langle \boldsymbol{\varphi}_t \circ \boldsymbol{\delta}_t \rangle \tag{1}$$

to denote policy concatenation.

A *future policy* ψ^i_t of agent i specifies all the future behavior relative to stage t. That is, $\psi^i_t = \left(\delta^i_{t+1}, \ldots, \delta^i_{h-1}\right)$. We also consider future joint policies $\boldsymbol{\psi}_t = (\boldsymbol{\delta}_{t+1}, \ldots, \boldsymbol{\delta}_{h-1})$. The structure of a policy π^i can be represented as

$$\pi^i = (\underbrace{\delta^i_0, \delta^i_1, \ldots, \delta^i_{t-1}}_{\varphi^i_t}, \delta^i_t, \underbrace{\delta^i_{t+1}, \ldots, \delta^i_{h-1}}_{\psi^i_t}) \tag{2}$$

and similarly for joint policies.

Since policies can be represented as trees (remember Figure 3), a different way to decompose them is by considering sub-trees. Define the *time-to-go* τ at stage t as

$$\tau = h - t.$$

Now $q^i_{\tau=k}$ denotes a k-steps-to-go *sub-tree policy* for agent i. That is, $q^i_{\tau=k}$ is a policy tree that has the same form as a full policy for the horizon-k problem. Within the original horizon-h problem $q^i_{\tau=k}$ is a candidate for execution starting at stage $t = h-k$. The set of k-steps-to-go sub-tree policies

for agent i is denoted $Q^i_{\tau=k}$. A joint sub-tree policy $\boldsymbol{q}_{\tau=k} \in \boldsymbol{Q}_{\tau=k}$ specifies a sub-tree policy for each agent.

Figure 4 shows the different structures in a policy for a fictitious Dec-POMDP with $h = 3$. It represents decision rules by dotted ellipses. It also shows a past policy φ^i_2 and illustrates how policy concatenation $\langle \varphi^i_2 \circ \delta^i_2 \rangle = \pi^i$ forms the full policy. This full policy also corresponds to a 3-steps-to-go sub-tree policy $q^i_{\tau=3}$; two of the sub-tree policies are indicated using dashed ellipses.

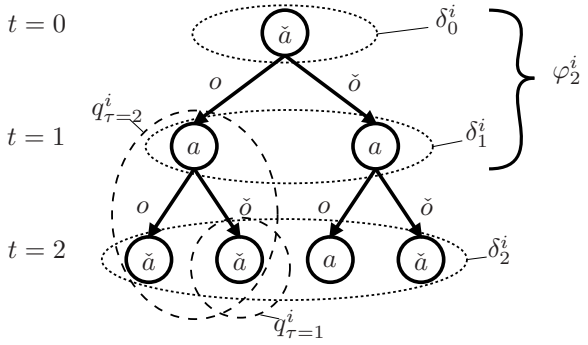


Fig. 4: Structure of a policy for an agent with actions $\{a,\check{a}\}$ and observations $\{o,\check{o}\}$. A policy π^i can be divided into decision rules δ^i or sub-tree policies q^i.

Definition 7 (Policy consumption). Providing a length-k (joint) sub-tree policy $\boldsymbol{q}_{\tau=k}$ with a sequence of $l < k$ (joint) observations *consumes* a part of $\boldsymbol{q}_{\tau=k}$ leading to a (joint) sub-tree policy which is a sub-tree of $\boldsymbol{q}_{\tau=k}$. In particular, consumption $\Downarrow$ by a single joint observation $\boldsymbol{o}$ is written as

$$\boldsymbol{q}_{\tau=k-1} = \boldsymbol{q}_{\tau=k}\Downarrow_{\boldsymbol{o}}. \tag{3}$$

For instance, in Figure 4, $q^i_{\tau=1} = q^i_{\tau=2}\Downarrow_{\check{o}}$.

3.4 The Quality of Joint Policies

Joint policies differ in how much reward they can expect to accumulate, which will serve as a criterion of their quality. Formally, we consider the expected cumulative reward of a joint policy, also referred to as its *value*.

Definition 8. The *value* $V(\boldsymbol{\pi})$ of a joint policy $\boldsymbol{\pi}$ is defined as

$$V(\boldsymbol{\pi}) \triangleq E\Big[\sum_{t=0}^{h-1} R(s_t,\boldsymbol{a}_t)\Big| I,\boldsymbol{\pi}\Big],$$

where the expectation is over states and observations.

This expectation can be computed using a recursive formulation. For the last stage $t = h - 1$, the value is given simply by the immediate reward

$$V^{\boldsymbol{\pi}}(s_{h-1},\bar{\boldsymbol{o}}_{h-1}) = R\left(s_{h-1},\boldsymbol{\pi}(\bar{\boldsymbol{o}}_{h-1})\right).$$

For all other stages, the expected value is given by:

$$V^{\boldsymbol{\pi}}(s_t,\bar{\boldsymbol{o}}_t) = R\left(s_t,\boldsymbol{\pi}(\bar{\boldsymbol{o}}_t)\right) + \sum_{s_{t+1}\in S}\sum_{\boldsymbol{o}\in\boldsymbol{O}} \Pr(s_{t+1},\boldsymbol{o}|s_t,\boldsymbol{\pi}(\bar{\boldsymbol{o}}_t))V^{\boldsymbol{\pi}}(s_{t+1},\bar{\boldsymbol{o}}_{t+1}). \tag{4}$$

Here, the probability is simply the product of the transition and observation probabilities $\Pr(s',\boldsymbol{o}|s,\boldsymbol{a}) = \Pr(\boldsymbol{o}|\boldsymbol{a},s')\Pr(s'|s,\boldsymbol{a})$. In essence, fixing the joint policy transforms the Dec-POMDP to a Markov chain with states $(s_t,\bar{\boldsymbol{o}}_t)$. Evaluating this equation via dynamic programming will result in the value for all $(s_0,\bar{\boldsymbol{o}}_0)$-pairs. The value $V(\boldsymbol{\pi})$ is then given by weighting these pairs according to the initial state distribution I. Note that given a fixed joint policy $\boldsymbol{\pi}$, a history $\bar{\boldsymbol{o}}_t$ actually induces a joint sub-tree policy. As such, it is possible to rewrite (4) in terms of sub-tree policies. Executing $\boldsymbol{q}_{\tau=k}$ over the last k stages, starting from a state s at stage $t = h - k$ will achieve:

$$V(s_t,\boldsymbol{q}_{\tau=k}) = R(s_t,\boldsymbol{a}_t) + \sum_{s_{t+1}\in S}\sum_{\boldsymbol{o}\in\boldsymbol{O}} \Pr(s_{t+1},\boldsymbol{o}|s_t,\boldsymbol{a}_t)V(s_{t+1},\boldsymbol{q}_{\tau=k}\Downarrow_{\boldsymbol{o}}) \tag{5}$$

where $\boldsymbol{a}_t$ is the joint action specified by (the root of) $\boldsymbol{q}_{\tau=k}$.

Finally, as is apparent from the above equations, the probabilities of states and histories are important in many computations. The following equation recursively specifies the probabilities of states and joint AOHs under a (potentially stochastic) past joint policy:

$$\Pr(s_t,\bar{\boldsymbol{\theta}}_t|I,\boldsymbol{\varphi}_t) = \sum_{s_{t-1}\in S}\sum_{\boldsymbol{a}_{t-1}\in\boldsymbol{A}} \Pr(s_t,\boldsymbol{o}_t|s_{t-1},\boldsymbol{a}_{t-1})\Pr(\boldsymbol{a}_{t-1}|\bar{\boldsymbol{\theta}}_{t-1},\boldsymbol{\varphi}_t) \Pr(s_{t-1},\bar{\boldsymbol{\theta}}_{t-1}|I,\boldsymbol{\varphi}_t). \tag{6}$$

4 Solution of Finite-Horizon Dec-POMDPs

This section gives an overview of methods proposed for finding exact and approximate solutions for finite-horizon Dec-POMDPs. For the infinite-horizon problem, which is significantly different, some pointers are provided in Section 5.

4.1 Brute Force Search & Dec-POMDP Complexity

Because there exists an optimal deterministic joint policy for a finite-horizon Dec-POMDP, it is possible to enumerate all joint policies, evaluate them as described in Section 3.4 and choose the best one. However, the number of such joint policies is

$$O\left(|A^\dagger|^{\frac{n(|O^\dagger|^h-1)}{|O^\dagger|-1}}\right),$$

where $|A^\dagger|$ and $|O^\dagger|$ denote the largest individual action and observation sets. The cost of evaluating each joint policy is $O\left(|S| \times |O^\dagger|^{nh}\right)$. It is clear that this approach therefore is only suitable for very small problems. This analysis provides some intuition about how hard the problem is. This intuition is supported by the complexity result due to Bernstein et al (2002).

Theorem 1 (Dec-POMDP complexity). *The problem of finding the optimal solution for a finite-horizon Dec-POMDP with $n \geq 2$ is NEXP-complete.*

NEXP is the class of problems that takes non-deterministic exponential time. Non-deterministic means that, similar to NP, it requires generating a guess about the solution in a non-deterministic way. Exponential time means that verifying whether the guess is a solution takes exponential time. In practice this means that (assuming NEXP $\neq$ EXP) solving a Dec-POMDP takes doubly exponential time in the worst case. Moreover, Dec-POMDPs cannot be approximated efficiently: Rabinovich et al (2003) showed that even finding an ϵ-approximate solution is NEXP-complete.

4.2 Alternating Maximization

Joint Equilibrium based Search for Policies (JESP) (Nair et al, 2003c) is a method that is guaranteed to find a locally optimal joint policy, more specifically, a *Nash equilibrium*: a tuple of policies such that for each agent i its policy π^i is a best response for the policies employed by the other agents π^{-i}. It relies on a process called *alternating maximization*. This is a procedure that computes a policy π^i for an agent i that maximizes the joint reward, while keeping the policies of the other agents fixed. Next, another agent is chosen to maximize the joint reward by finding its best response. This process is repeated until the joint policy converges to a Nash equilibrium, which is a local optimum. This process is also referred to as *hill-climbing* or *coordinate ascent*. Note that the local optimum reached can be arbitrarily bad. For instance, if agent 1 opens the left (a_{OL}) door right away in the Dec-Tiger problem, the best response for agent 2 is to also select a_{OL}. To reduce the impact of such bad local optima, JESP can use random restarts.

JESP uses a dynamic programming approach to compute the best-response policy for a selected agent i. In essence, fixing π^{-i} allows for a reformulation of the problem as an augmented POMDP. In this augmented POMDP a state $\check{s} = \langle s,\bar{o}^{-i}\rangle$ consists of a nominal state s and the observation histories of the other agents $\bar{o}^{-i}$. Given the fixed deterministic policies of other agents π^{-i}, such an augmented state $\check{s}$ is Markovian and all transition and observation probabilities can be derived from π^{-i} and the transition and observation model of the original Dec-POMDP.

4.3 Optimal Value Functions for Dec-POMDPs

This section describes an approach more in line with methods for single agent MDPs and POMDPs: we identify an optimal value function Q^* that can be used to derive an optimal policy. Even though computation of Q^* itself is intractable, the insight it provides is valuable. In particular, it has a clear relation with the two dominant approaches to solving Dec-POMDPs: the forward and the backward approach which will be explained in the following subsections.

4.3.1 Selecting Sub-Tree Policies

Let us start by considering Figure 5, which illustrates a tree of joint AOHs. For a particular joint AOH (a node in Figure 5), we can try to determine which joint sub-tree policy $\boldsymbol{q}_{\tau=k}$ is optimal. Recall that $V(s_t,\boldsymbol{q}_{\tau=k})$ the value of $\boldsymbol{q}_{\tau=k}$ starting from s_t is specified by (5). Also, let $\boldsymbol{b}(s) \triangleq \Pr(s|I,\bar{\boldsymbol{\theta}}_t)$ be the *joint belief* corresponding to $\bar{\boldsymbol{\theta}}_t$ which can be computed using Bayes' rule in the same way as the POMDP belief update (see Chapter **??**). Given an initial belief I and joint AOH $\bar{\boldsymbol{\theta}}_t$, we can compute a value for each joint sub-tree policy $\boldsymbol{q}_{\tau=k}$ that can be used from that node onward via

$$V(I,\bar{\boldsymbol{\theta}}_t,\boldsymbol{q}_{\tau=k}) = \sum_{s\in S} \boldsymbol{b}(s) V(s,\boldsymbol{q}_{\tau=k}). \tag{7}$$

Now, it is possible to rewrite (7) recursively:

$$V(I,\bar{\boldsymbol{\theta}}_t,\boldsymbol{q}_{\tau=k}) = R(\bar{\boldsymbol{\theta}}_t,\boldsymbol{a}_t) + \sum_{\boldsymbol{o}} \Pr(\boldsymbol{o}|\boldsymbol{b},\boldsymbol{a}) V(I,\bar{\boldsymbol{\theta}}_{t+1},\boldsymbol{q}_{\tau=k}\Downarrow_{\boldsymbol{o}}), \tag{8}$$

where the expected immediate reward is given by:

$$R(\bar{\boldsymbol{\theta}}_t,\boldsymbol{a}) = \sum_{s_t\in S} \boldsymbol{b}(s_t) R(s_t,\boldsymbol{a}). \tag{9}$$

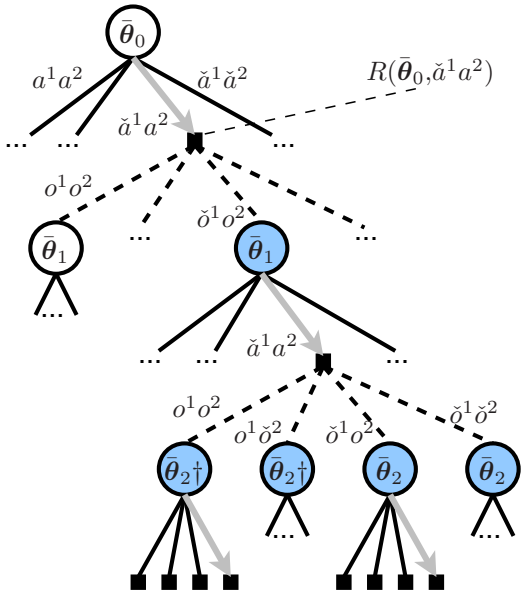


Fig. 5: Tree of joint AOHs $\bar{\boldsymbol{\theta}}$ for a fictitious 2-agent Dec-POMDP with actions $\left\{\left\{a^1,\check{a}^1\right\},\left\{a^2,\check{a}^2\right\}\right\}$ and observations $\left\{\left\{o^1,\check{o}^1\right\},\left\{o^2,\check{o}^2\right\}\right\}$. Given I, the AOHs induce a 'joint belief' $\boldsymbol{b}(s)$ over states. Solid lines represent joint actions and dashed lines joint observations. Due to the size of the tree it is only partially shown. Highlighted joint actions represent a joint policy. Given a joint sub-tree policy at a node (the action choices made in the sub-tree below it), the value is given by (8). However, action choices are not independent in different parts of the trees: e.g., the two nodes marked † have the same $\bar{\theta}^1$ and therefore should specify the same sub-tree policy for agent 1.

Therefore one would hope that a dynamic programming approach would be possible, where, for each $\bar{\boldsymbol{\theta}}_t$ one could choose the maximizing $\boldsymbol{q}_{\tau=k}$. Unfortunately, running such a procedure on the entire tree is not possible because of the decentralized nature of a Dec-POMDP: it is not possible to choose maximizing joint sub-tree policies $\boldsymbol{q}_{\tau=k}$ independently, since this could lead to a centralized joint policy.

The consequence is that, even though (8) can be used to compute the value for a $(\bar{\boldsymbol{\theta}}_t,\boldsymbol{q}_{\tau=k})$-pair, it does not directly help to optimize the joint policy, because we cannot reason about parts of the joint AOH tree independently. Instead, one should decide what sub-tree policies to select by considering all $\bar{\boldsymbol{\theta}}_t$ of an entire stage t *at the same time, assuming a past joint policy* $\boldsymbol{\varphi}_t$. That is, when we assume we have computed $V(I,\bar{\boldsymbol{\theta}}_t,\boldsymbol{q}_{\tau=k})$ for all $\bar{\boldsymbol{\theta}}_t$ and for all $\boldsymbol{q}_{\tau=k}$, then we can compute a special form of joint decision rule $\Gamma_t = \left\langle \Gamma_t^i \right\rangle_{i\in\mathcal{D}}$ for stage t. Here, the individual decision rules map individual histories to individual sub-tree policies $\Gamma_t^i : \bar{\Theta}_t^i \to Q_{\tau=k}^i$. The optimal Γ_t satisfies:

$$\Gamma_t^* = \arg\max_{\Gamma_t} \sum_{\bar{\boldsymbol{\theta}}_t \in \bar{\boldsymbol{\Theta}}_t} \Pr(\bar{\boldsymbol{\theta}}_t | I, \boldsymbol{\varphi}_t) V(I, \bar{\boldsymbol{\theta}}_t, \Gamma_t(\bar{\boldsymbol{\theta}}_t)), \tag{10}$$

where $\Gamma_t(\bar{\boldsymbol{\theta}}_t) = \left\langle \Gamma_t^i(\bar{\theta}_t^i) \right\rangle_{i \in \mathcal{D}}$ denotes the joint sub-tree policy $\boldsymbol{q}_{\tau=k}$ resulting from application of the individual decision rules and the probability is a marginal of (6).

This equation clearly illustrates that the optimal joint policy at a stage t of a Dec-POMDP depends on $\boldsymbol{\varphi}_t$, the joint policy followed up to stage t. Moreover, there are additional complications that make (10) impractical to use:

1. It sums over joint AOHs, the number of which is exponential in both the number of agents and t.
2. It assumes computation of $V(I, \bar{\boldsymbol{\theta}}_t, \boldsymbol{q}_{\tau=k})$ for all $\bar{\boldsymbol{\theta}}_t$, for all $\boldsymbol{q}_{\tau=k}$.
3. The number of Γ_t to be evaluated is $O(|Q^\dagger_{\tau=k}|^{|\bar{\Theta}_t^\dagger| n})$, where '$\dagger$' denotes the largest set. $|Q^\dagger_{\tau=k}|$ is doubly exponential in k and $|\bar{\Theta}_t^\dagger|$ is exponential in t. Therefore the number of Γ_t is doubly exponential in $h = t + k$.

Note that by restricting our attention to deterministic $\boldsymbol{\varphi}_t$ it is possible to reformulate (10) as a summation over OHs, rather than AOHs (this involves adapting V to take $\boldsymbol{\varphi}_t$ as an argument). However, for such a reformulation, the same complications hold.

4.3.2 Selecting Optimal Decision Rules

This section shifts the focus back to regular decision rules δ^i—as introduced in Section 3.3—that map from OHs (or AOHs) to *actions*. We will specify a value function that quantifies the expected value of taking actions as specified by $\boldsymbol{\delta}_t$ and continuing optimally afterward. That is, we replace the value of sub-trees in (10) by the optimal value of decision rules. The optimal value function for a finite-horizon Dec-POMDP is defined as follows.

Theorem 2 (Optimal Q^*). *The optimal Q-value function $Q^*(I, \boldsymbol{\varphi}_t, \bar{\boldsymbol{\theta}}_t, \boldsymbol{\delta}_t)$ is a function of the initial state distribution and joint past policy, AOH and decision rule. For the last stage, it is given by*

$$Q^*(I, \boldsymbol{\varphi}_{h-1}, \bar{\boldsymbol{\theta}}_{h-1}, \boldsymbol{\delta}_{h-1}) = R(\bar{\boldsymbol{\theta}}_{h-1}, \boldsymbol{\delta}_{h-1}(\bar{\boldsymbol{\theta}}_{h-1})), \tag{11}$$

as defined by (9)*, and, for all $0 \leq t < h-1$, by*

$$Q^*(I, \boldsymbol{\varphi}_t, \bar{\boldsymbol{\theta}}_t, \boldsymbol{\delta}_t) = R(\bar{\boldsymbol{\theta}}_t, \boldsymbol{\delta}_t(\bar{\boldsymbol{\theta}}_t)) + \sum_{\boldsymbol{o}} \Pr(\boldsymbol{o} | \bar{\boldsymbol{\theta}}_t, \boldsymbol{\delta}_t(\bar{\boldsymbol{\theta}}_t)) Q^*(I, \boldsymbol{\varphi}_{t+1}, \bar{\boldsymbol{\theta}}_{t+1}, \boldsymbol{\delta}^*_{t+1}), \tag{12}$$

with $\boldsymbol{\varphi}_{t+1} = \langle \boldsymbol{\varphi}_t \circ \boldsymbol{\delta}_t \rangle$, $\bar{\boldsymbol{\theta}}_{t+1} = (\bar{\boldsymbol{\theta}}_t, \boldsymbol{\delta}_t(\bar{\boldsymbol{\theta}}_t), \boldsymbol{o})$ and

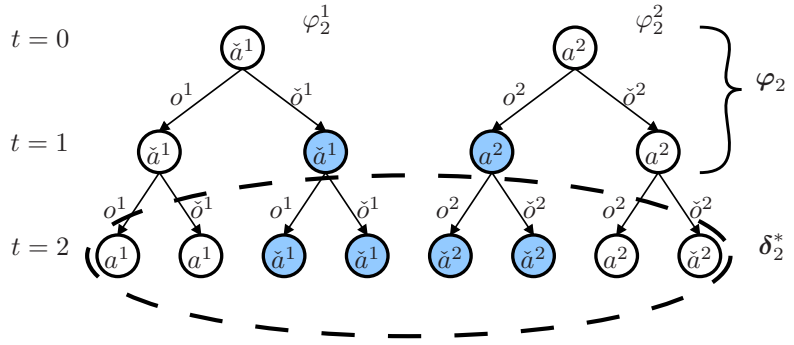


Fig. 6: Computation of Q^*. The dashed ellipse indicates the optimal decision rule $\boldsymbol{\delta}_2^*$ for stage $t=2$, given that $\boldsymbol{\varphi}_2 = \langle \boldsymbol{\varphi}_1 \circ \boldsymbol{\delta}_1 \rangle$ is followed for the first two stages. The entries $Q^*(I,\bar{\boldsymbol{\theta}}_1,\boldsymbol{\varphi}_1,\boldsymbol{\delta}_1)$ are computed by propagating relevant Q^*-values of the next stage. For instance, the Q^*-value under $\boldsymbol{\varphi}_2$ for the highlighted joint history $\bar{\boldsymbol{\theta}}_1 = \langle (\check{a}^1,\check{o}^1),(a^2,o^2) \rangle$ is computed by propagating the values of the four successor joint histories, as per (12).

$$\boldsymbol{\delta}_{t+1}^* = \arg\max_{\boldsymbol{\delta}_{t+1}} \sum_{\bar{\boldsymbol{\theta}}_{t+1} \in \bar{\boldsymbol{\Theta}}_{t+1}} \Pr(\bar{\boldsymbol{\theta}}_{t+1} | I, \boldsymbol{\varphi}_{t+1}) Q^*(I, \boldsymbol{\varphi}_{t+1}, \bar{\boldsymbol{\theta}}_{t+1}, \boldsymbol{\delta}_{t+1}). \tag{13}$$

Proof. Because of (11)*, application of* (13) *for the last stage will maximize the expected reward and thus is optimal. Equation* (12) *propagates these optimal values to the preceding stage. Optimality for all stages follows by induction.*

Note that $\boldsymbol{\varphi}_t$ is necessary in order to compute $\boldsymbol{\delta}_{t+1}^*$, the optimal joint decision rule at the next stage, because (13) requires $\boldsymbol{\varphi}_{t+1}$ and thus $\boldsymbol{\varphi}_t$.

The above equations constitute a dynamic program. When assuming that only deterministic joint past policies $\boldsymbol{\varphi}$ can be used, the dynamic program can be evaluated from the end ($t = h-1$) to the beginning ($t=0$). Figure 6 illustrates the computation of Q^*. When arriving at stage 0, the past joint policy is empty $\boldsymbol{\varphi}_0 = ()$ and joint decision rules are simply joint actions, thus it is possible to select

$$\boldsymbol{\delta}_0^* = \arg\max_{\boldsymbol{\delta}_0} Q^*(I,\boldsymbol{\varphi}_0,\bar{\boldsymbol{\theta}}_0,\boldsymbol{\delta}_0) = \arg\max_{\boldsymbol{a}} Q^*(I,(),(),\boldsymbol{a}).$$

Then given $\boldsymbol{\varphi}_1 = \boldsymbol{\delta}_0^*$ we can determine $\boldsymbol{\delta}_1^*$ using (13), etc.[5] This procedure, we refer to as *forward-sweep policy computation (FSPC)* using Q^*. The principle of FSPC is that a new decision rule is selected given the past joint policy found so far and is illustrated in Figure 7a.

Unfortunately, computing Q^* itself is intractable, since it means evaluating the dynamic program of Theorem 2 for all past joint policies. In particular,

[5] Note that performing the maximization in (13) has already been done and can be cached.

(11) will need to be evaluated for all $(\boldsymbol{\varphi}_{h-1},\boldsymbol{\delta}_{h-1})$ and these pairs have a direct correspondence to all joint policies: $\boldsymbol{\pi} = \langle \boldsymbol{\varphi}_{h-1} \circ \boldsymbol{\delta}_{h-1} \rangle$. Therefore, the time needed to evaluate this DP is doubly exponential in h. This means that the practical value of Q^* is limited.

The identified Q^* has a form quite different from Q-value functions encountered in MDPs and POMDPs. We still use the symbol 'Q', because $\boldsymbol{\delta}_t$ can be seen as an action on the meta-level of the planning process. In this process $(I,\boldsymbol{\varphi}_t)$ can be interpreted as the state and we can define V and Q with their usual interpretations. In particular, it is possible to write

$$V^*(I,\boldsymbol{\varphi}_t) = \max_{\boldsymbol{\delta}_t} Q^*(I,\boldsymbol{\varphi}_t,\boldsymbol{\delta}_t) \tag{14}$$

where Q^* is defined as

$$Q^*(I,\boldsymbol{\varphi}_t,\boldsymbol{\delta}_t) = \sum_{\bar{\boldsymbol{\theta}}_t} \Pr(\bar{\boldsymbol{\theta}}_t|I,\boldsymbol{\varphi}_t) Q^*(I,\boldsymbol{\varphi}_t,\bar{\boldsymbol{\theta}}_t,\boldsymbol{\delta}_t).$$

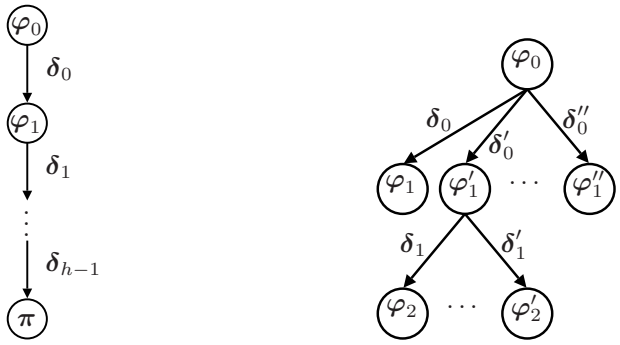
By expanding this definition of Q^* using (12), one can verify that it indeed has the regular interpretation of the expected immediate reward induced by first taking 'action' $\boldsymbol{\delta}_t$ plus the cumulative reward of continuing optimally afterward (Oliehoek, 2010).

4.4 Forward Approach: Heuristic Search

The previous section explained that once Q^* is computed, it is possible to extract $\boldsymbol{\pi}^*$ by performing forward-sweep policy computation: repeatedly applying (13) for consecutive stages $t = 0,1,\ldots,h-1$. When processing stage t, stages $0 \ldots t-1$ have been processed already. Therefore a past joint policy $\boldsymbol{\varphi}_t = (\boldsymbol{\delta}_0,\ldots,\boldsymbol{\delta}_{t-1})$ is available and the probability $\Pr(\bar{\boldsymbol{\theta}}_t|I,\boldsymbol{\varphi}_t)$ is defined. Unfortunately, computing Q^* itself is intractable. One idea to overcome this problem is to use an approximation $\widehat{Q}$ that is easier to compute. We refer to this as *the forward approach to Dec-POMDPs.*

4.4.1 Dec-POMDPs as Series of Bayesian Games

A straightforward approach is to try and apply forward-sweep policy computation using a heuristic Q-value function $\widehat{Q}$. This is essentially what the method introduced by Emery-Montemerlo et al (2004) does. It represents a Dec-POMDP as a series of *collaborative Bayesian games (CBGs),* one for each stage t, with an approximate payoff function $\widehat{Q}(\bar{\boldsymbol{\theta}}_t,\boldsymbol{a})$. A Bayesian game (BG) (Osborne and Rubinstein, 1994) is an extension of a strategic form game in which the agents have private information. A CBG is a BG with



(a) Forward-sweep policy computation (FSPC) can be used with Q^* or a heuristic $\widehat{Q}$.

(b) (Generalized) MAA* performs backtracking and hence only is useful with (admissible) heuristics $\widehat{Q}$.

Fig. 7: Forward approach to Dec-POMDPs.

identical payoffs. By solving CBGs for consecutive stages it is possible to find an approximate solution. This is forward-sweep policy computation (with $\widehat{Q}$).

In a Dec-POMDP, the crucial difficulty in making a decision at some stage t is that the agents lack a common signal on which to condition their actions. They must instead base their actions on their individual histories. Given I and $\boldsymbol{\varphi}_t$, this situation can be modeled as a CBG. Such a CBG $B(I,\boldsymbol{\varphi}_t)$ consists of:

- the set of agents,
- their joint actions $\boldsymbol{A}$,
- the set of their joint AOHs $\bar{\boldsymbol{\Theta}}_t$,
- a probability distribution over them $\Pr(\bar{\boldsymbol{\theta}}_t|I,\boldsymbol{\varphi}_t)$, and
- a payoff function $\widehat{Q}(\bar{\boldsymbol{\theta}}_t,\boldsymbol{a})$.

In the CBG agents use policies that map from their individual AOHs to actions. That is, a policy of an agent i for a CBG corresponds to a decision rule δ_t^i for the Dec-POMDP. The solution of the CBG is the joint decision rule $\boldsymbol{\delta}_t$ that maximizes the expected payoff with respect to $\widehat{Q}$:

$$\hat{\boldsymbol{\delta}}_t^{\,*} = \arg\max_{\boldsymbol{\delta}_t} \sum_{\bar{\boldsymbol{\theta}}_t \in \bar{\boldsymbol{\Theta}}_t} \Pr(\bar{\boldsymbol{\theta}}_t|I,\boldsymbol{\varphi}_t)\widehat{Q}(\bar{\boldsymbol{\theta}}_t,\boldsymbol{\delta}_t(\bar{\boldsymbol{\theta}}_t)). \tag{15}$$

Again, if $\boldsymbol{\varphi}_t$ is deterministic, the probability of $\bar{\boldsymbol{\theta}}_t = \langle \bar{\boldsymbol{a}}_t,\bar{\boldsymbol{o}}_t \rangle$ is non-zero for exactly one $\bar{\boldsymbol{a}}_t$, which means that attention can be restricted to OHs and decision rules that map from OHs to actions.

4.4.2 Heuristic Q-Value Functions

While the CBG for a stage is fully specified given I,$\boldsymbol{\varphi}_t$ and $\widehat{Q}$, it is not obvious how to choose $\widehat{Q}$. Here we discuss this issue.

Note that, for the last stage $t = h-1$, $\hat{\boldsymbol{\delta}}_t^*$ has a close relation[6] with the optimal decision rule selected by (13): if for the last stage the heuristic specifies the immediate reward $\widehat{Q}(\bar{\boldsymbol{\theta}}_t,\boldsymbol{a}) = R(\bar{\boldsymbol{\theta}}_t,\boldsymbol{a})$, both will select the same actions. That is, in this case $\hat{\boldsymbol{\delta}}_t^* = \boldsymbol{\delta}_t^*$.

While for other stages it is not possible to specify such a strong correspondence, note that FSPC via CBGs is not sub-optimal per se: It is possible to compute a value function of the form $Q^{\boldsymbol{\pi}}(\bar{\boldsymbol{\theta}}_t,\boldsymbol{a})$ for any $\boldsymbol{\pi}$. Doing this for a $\boldsymbol{\pi}^*$ yields $Q^{\boldsymbol{\pi}^*}$ and when using the latter as the payoff functions for the CBGs, FSPC is exact (Oliehoek et al, 2008b).[7]

However, the practical value of this insight is limited, since it requires knowing an optimal policy to start with. In practice, research has considered using an approximate value function. For instance, it is possible to compute the value function $Q_{\text{M}}(s,\boldsymbol{a})$ of the 'underlying MDP': the MDP with the same transition and reward function as the Dec-POMDP (Emery-Montemerlo et al, 2004; Szer et al, 2005). This can be used to compute $\widehat{Q}(\bar{\boldsymbol{\theta}}_t,\boldsymbol{a}) = \sum_s \boldsymbol{b}(s) Q_{\text{M}}(s,\boldsymbol{a})$, which can be used as the payoff function for the CBGs. This is called Q_{MDP}. Similarly, it is possible to use the value function of the 'underlying POMDP' (Q_{POMDP}) (Roth et al, 2005b; Szer et al, 2005), or the value function of the problem with 1-step delayed communication (Q_{BG}) (Oliehoek and Vlassis, 2007).

A problem in FSPC is that (15) still maximizes over $\boldsymbol{\delta}_t$ that map from histories to actions; the number of such $\boldsymbol{\delta}_t$ is doubly exponential in t. There are two main approaches to gain leverage. First, the maximization in (15) can be performed more efficiently: approximately via alternating maximization (Emery-Montemerlo et al, 2004), or exactly via heuristic search (Kumar and Zilberstein, 2010b; Oliehoek et al, 2010). Second, it is possible to reduce the number of histories under concern via pruning (Emery-Montemerlo et al, 2004), approximate clustering (Emery-Montemerlo et al, 2005) or lossless clustering (Oliehoek et al, 2009).

---

[6] Because Q^* is a function of $\boldsymbol{\varphi}_t$ and $\boldsymbol{\delta}_t$, (13) has a slightly different form than (15). The former technically does not correspond to a CBG, while the latter does.

[7] There is a subtle but important difference between $Q^{\boldsymbol{\pi}^*}(\bar{\boldsymbol{\theta}}_t,\boldsymbol{a})$ and $Q^*(I,\boldsymbol{\varphi}_t,\bar{\boldsymbol{\theta}}_t,\boldsymbol{\delta}_t)$: the latter specifies the optimal value given *any* past joint policy $\boldsymbol{\varphi}_t$ while the former only specifies optimal value given that $\boldsymbol{\pi}^*$ is actually being followed.

4.4.3 Multi-Agent A*

Since FSPC using $\widehat{Q}$ can be seen as a single trace in a search tree, a natural idea is to allow for back-tracking and perform a full heuristic search as in *multi-agent* A* *(*MAA**)* (Szer et al, 2005), illustrated in Figure 7b.

MAA* performs an A* search over past joint policies $\boldsymbol{\varphi}_t$. It computes a heuristic value $\widehat{V}(\boldsymbol{\varphi}_t)$ by taking $V^{0\ldots t-1}(\boldsymbol{\varphi}_t)$, the actual expected reward over the first t stages, and adding $\widehat{V}^{t\ldots h-1}$, a heuristic value for the remaining $h-t$ stages. When the heuristic is *admissible*—a guaranteed overestimation—so is $\widehat{V}(\boldsymbol{\varphi}_t)$. MAA* performs standard A* search (Russell and Norvig, 2003): it maintains an open list P of partial joint policies $\boldsymbol{\varphi}_t$ and their heuristic values $\widehat{V}(\boldsymbol{\varphi}_t)$. On every iteration MAA* selects the highest ranked $\boldsymbol{\varphi}_t$ and expands it, generating and heuristically evaluating all $\boldsymbol{\varphi}_{t+1} = \langle \boldsymbol{\varphi}_t \circ \boldsymbol{\delta}_t \rangle$ and placing them in P. When using an admissible heuristic, the heuristic values $\widehat{V}(\boldsymbol{\varphi}_{t+1})$ of the newly expanded policies are an upper bound to the true values and any lower bound $\underline{v}^\star$ that has been found can be used to prune P. The search ends when the list becomes empty, at which point an optimal fully specified joint policy has been found.

There is a direct relation between MAA* and the optimal value functions described in the previous section: V^* given by (14) is the optimal heuristic $\widehat{V}^{t\ldots h-1}$ (note that V^* only specifies reward from stage t onward).

MAA* suffers from the same problem as FSPC via CBGs: the number of $\boldsymbol{\delta}_t$ grows doubly exponential with t, which means that the number of children of a node grows doubly exponential in its depth. In order to mitigate the problem, it is possible to apply lossless clustering (Oliehoek et al, 2009), or to try and avoid the expansion of all child nodes by incrementally expanding nodes only when needed (Spaan et al, 2011).

4.4.4 Generalized MAA*

Even though Figure 7 shows a clear relation between FSPC and MAA*, it is not directly obvious how they relate: the former solves CBGs, while the latter performs heuristic search. Generalized MAA* (GMAA*) (Oliehoek et al, 2008b) unifies these two approaches by making explicit the '`Expand`' operator.

Algorithm 1 shows GMAA*. When the `Select` operator selects the highest ranked $\boldsymbol{\varphi}_t$ and when the `Expand` operator works as described for MAA*, GMAA* simply is MAA*. Alternatively, the `Expand` operator can construct a CBG $B(I,\boldsymbol{\varphi}_t)$ for which all joint CBG-policies $\boldsymbol{\delta}_t$ are evaluated. These can then be used to construct a new set of partial policies $\boldsymbol{\Phi}_{\texttt{Expand}} = \{\langle \boldsymbol{\varphi}_t \circ \boldsymbol{\delta}_t \rangle\}$ and their heuristic values. This corresponds to MAA* reformulated to work on CBGs. It can be shown that when using a particular form of $\widehat{Q}$ (including the mentioned heuristics $\mathrm{Q}_{\mathrm{MDP}}$, $\mathrm{Q}_{\mathrm{POMDP}}$ and Q_{BG}), the approaches are

Initialize: $\underline{\mathrm{v}}^\star \leftarrow -\infty$, $\mathrm{P} \leftarrow \{\boldsymbol{\varphi}_0 = ()\}$
repeat
  $\boldsymbol{\varphi}_t \leftarrow$ `Select`(P)
  $\boldsymbol{\Phi}_{\text{Expand}} \leftarrow$ `Expand`(I,$\boldsymbol{\varphi}_t$)
  if $\boldsymbol{\Phi}_{\text{Expand}}$ contains full policies $\boldsymbol{\Pi}_{\text{Expand}} \subseteq \boldsymbol{\Phi}_{\text{Expand}}$ **then**
    $\boldsymbol{\pi}' \leftarrow \arg\max_{\boldsymbol{\pi} \in \boldsymbol{\Pi}_{\text{Expand}}} V(\boldsymbol{\pi})$
    if $V(\boldsymbol{\pi}') > \underline{\mathrm{v}}^\star$ **then**
      $\underline{\mathrm{v}}^\star \leftarrow V(\boldsymbol{\pi}')$ {found new lower bound}
      $\boldsymbol{\pi}^\star \leftarrow \boldsymbol{\pi}'$
      $\mathrm{P} \leftarrow \{\boldsymbol{\varphi} \in \mathrm{P} \mid \widehat{V}(\boldsymbol{\varphi}) > \underline{\mathrm{v}}^\star\}$ {prune P}
    $\boldsymbol{\Phi}_{\text{Expand}} \leftarrow \boldsymbol{\Phi}_{\text{Expand}} \setminus \boldsymbol{\Pi}_{\text{Expand}}$ {remove full policies}
  $\mathrm{P} \leftarrow (\mathrm{P} \setminus \boldsymbol{\varphi}_t) \cup \{\boldsymbol{\varphi} \in \boldsymbol{\Phi}_{\text{Expand}} \mid \widehat{V}(\boldsymbol{\varphi}) > \underline{\mathrm{v}}^\star\}$ {remove processed/add new $\boldsymbol{\varphi}$ }
until P is empty

Algorithm 1: (Generalized) MAA*

identical (Oliehoek et al, 2008b). GMAA* can also use an `Expand` operator that does not construct all new partial policies, but only the best-ranked one, $\boldsymbol{\Phi}_{\text{Expand}} = \{\langle \boldsymbol{\varphi}_t \circ \boldsymbol{\delta}_t^* \rangle\}$. As a result the open list P will never contain more than one partial policy and behavior reduces to FSPC. A generalization called k-GMAA* constructs the k best-ranked partial policies, allowing to trade off computation time and solution quality. Clustering of histories can also be applied in GMAA*, but only lossless clustering will preserve optimality.

4.5 Backwards Approach: Dynamic Programming

The forward approach to Dec-POMDPs incrementally builds policies from the first stage $t = 0$ to the last $t = h - 1$. Prior to doing this, a Q-value function (optimal Q^* or approximate $\widehat{Q}$) needs to be computed. This computation itself, the dynamic program represented in Theorem 2, starts with the last stage and works its way back. The resulting optimal values correspond to the expected values of a joint decision rule and continuing optimally afterwards. That is, in the light of (10) this can be interpreted as the computation of the value for a subset of optimal (useful) joint sub-tree policies.

This section treats dynamic programming (DP) for Dec-POMDPs (Hansen, Bernstein, and Zilberstein, 2004). This method also works backwards, but rather than computing a Q-value function, it directly computes a set of useful sub-tree policies.

4.5.1 Dynamic Programming for Dec-POMDPs

The core idea of DP is to incrementally construct sets of longer sub-tree policies for the agents: starting with a set of one-step-to-go ($\tau = 1$) sub-tree policies (actions) that can be executed at the last stage, construct a set of 2-step policies to be executed at $h - 2$, etc. That is, DP constructs $Q^i_{\tau=1}$,$Q^i_{\tau=2}$,...,$Q^i_{\tau=h}$ for all agents i. When the last backup step is completed, the optimal policy can be found by evaluating all induced joint policies $\boldsymbol{\pi} \in Q^1_{\tau=h} \times \cdots \times Q^n_{\tau=h}$ for the initial belief I as described in Section 3.4.

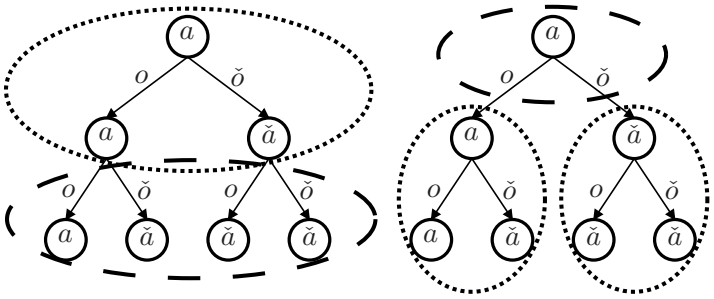


Fig. 8: Difference between policy construction in MAA* (left) and dynamic programming (right) for an agent with actions a,$\breve{a}$ and observations o,$\breve{o}$. Dashed components are newly generated, dotted components result from the previous iteration.

DP formalizes this idea using *backup* operations that construct $Q^i_{\tau=k+1}$ from $Q^i_{\tau=k}$. For instance, the right side of Figure 8 shows how $q^i_{\tau=3}$, a 3-steps-to-go sub-tree policy, is constructed from two $q^i_{\tau=2} \in Q^i_{\tau=2}$. In general, a one step extended policy $q^i_{\tau=k+1}$ is created by selecting a sub-tree policy for each observation and an action for the root. An exhaustive backup generates *all* possible $q^i_{\tau=k+1}$ that have policies from the previously generated set $q^i_{\tau=k} \in Q^i_{\tau=k}$ as their sub-trees. We will denote the sets of sub-tree policies resulting from exhaustive backup for each agent i by $Q^{e,i}_{\tau=k+1}$.

Unfortunately, sets of sub-tree policies maintained grow doubly exponentially with k.[8] To counter this source of intractability, it is possible to prune dominated sub-tree policies from $Q^{e,i}_{\tau=k}$, resulting in smaller maintained sets $Q^{m,i}_{\tau=k}$ (Hansen et al, 2004). The value of a $q^i_{\tau=k}$ depends on the probability distribution over states when it is started (at stage $t = h - k$) as well as the probability with which the other agents $j \neq i$ select their sub-tree policies. Therefore, a $q^i_{\tau=k}$ is dominated if it is not maximizing at any point in the *multi-agent belief space*: the simplex over $S \times Q^{m,-i}_{\tau=k}$. It is possible to test for

[8] Since the $q^i_{\tau=k}$ are essentially full policies for the horizon-k problem, their number is doubly exponentially in k.

dominance by linear programming. Removal of a dominated sub-tree policy $q^i_{\tau=k}$ of an agent i may cause a $q^j_{\tau=k}$ of another agent j to become dominated. Therefore DP iterates over agents until no further pruning is possible, a procedure known as *iterated elimination of dominated policies* (Osborne and Rubinstein, 1994).

In practice, the pruning step in DP often is not able to sufficiently reduce the maintained sets to make the approach tractable for larger problems. *Bounded DP (BDP)* can be used to compute a bounded approximation (Amato et al, 2007). It performs more aggressive ϵ-pruning: a $q^i_{\tau=k}$ that is maximizing in some region of the multi-agent belief space, but improves the value in this region by at most ϵ, is also pruned. Because iterated elimination using ϵ-pruning can still lead to an unbounded reduction in value, BDP performs one iteration of ϵ-pruning, followed by iterated elimination using normal pruning.

Even when many sub-tree policies can be pruned, DP can run into problems during the exhaustive backup. *Incremental policy generation (IPG)* is a technique to mitigate this problem by performing a one-step state reachability analysis (Amato et al, 2009). During the back up of sub-trees for an agent i, IPG analyzes the set of states $S_{\langle a^i,o^i\rangle}$ that have non-zero probability after each $\langle a^i,o^i\rangle$-pair (a particular observation may exclude many states). Subsequently, in constructing the set $Q^{e,i}_{\tau=k+1}$, only sub-tree policies that are non-dominated for $S_{\langle a^i,o^i\rangle}$ are selected for action a^i and observation o^i. This can lead to much smaller sets of sub-tree policies.

An additional difficulty in DP is that, in order to perform pruning, all the $V(s,\boldsymbol{q}_{\tau=k})$ values need to be computed and stored, which takes $|Q^{e,\dagger}_{\tau=k}|^n \times |S|$ real numbers. As such, DP runs out of memory well before it runs out of time. In order to address this problem Boularias and Chaib-draa (2008) represent these values more compactly by making use of a *sequence form* (Koller et al, 1994) representation. A disadvantage is that this approach can lead to keeping dominated policies, however. As such there is a trade-off between space required to store the values for all sub-tree policies and the number of sub-tree policies.

4.5.2 Point-Based DP

DP only removes $q^i_{\tau=k}$ that are not maximizing at *any point* in the multi-agent belief space. *Point-based DP (PBDP)* (Szer and Charpillet, 2006) proposes to improve pruning of the set $Q^{e,i}_{\tau=k}$ by considering only a subset $\mathcal{B}^i \subset \mathcal{P}(S \times Q^{-i}_{\tau=k})$ of *reachable* multi-agent belief points. Only those $q^i_{\tau=k}$ that maximize the value at some $b^i \in \mathcal{B}^i$ are kept. The definition of reachable is slightly involved.

Definition 9. A multi-agent belief point b^i_t is *reachable* if there exists a probability distribution $\Pr(s_t,\bar{\theta}^{-i}_t|I,\boldsymbol{\varphi}_t)$ (for any deterministic $\boldsymbol{\varphi}_t$) and an induced mapping $\Gamma^{-i}_t = \langle \Gamma^j_t \rangle_{j\neq i}$ with $\Gamma^j_t : \bar{\Theta}^j_t \to Q^j_{\tau=k}$ that result in b^i_t.

That is, a belief point b^i is reachable if there is a past joint policy that will result in the appropriate distribution over states and AOHs of other agents such that, when combined with a mapping of those AOHs to sub-tree policies, b^i is the resulting distribution over states and sub-tree policies.

PBDP can be understood in the light of (10). Suppose that the range of the Γ_t^i are restricted to the sets generated by exhaustive backup: $\Gamma_t^i : \bar{\Theta}_t^i \rightarrow Q_{\tau=k}^{e,i}$. Solving (10) for a past joint policy $\boldsymbol{\varphi}_t$ will result in Γ_t^* which will specify, for all agents, all the useful sub-tree policies $q_{\tau=k}^i \in Q_{\tau=k}^{e,i}$ *given* $\boldsymbol{\varphi}_t$. Solving (10) *for all* $\boldsymbol{\varphi}_t$ will result in the set of all potentially useful $q_{\tau=k}^i \in Q_{\tau=k}^{e,i}$.

Given a $\boldsymbol{\varphi}_t$ and a Γ_t^{-i}, (10) can be rewritten as a maximization from the perspective of agent i to compute its best response:[9]

$$BR^i(\bar{\theta}_t^i,\boldsymbol{\varphi}_t,\Gamma_t^{-i}) = \arg\max_{q_{\tau=k}^i} \sum_{\bar{\theta}_t^{-i}} \sum_{s_t} \Pr(s_t,\bar{\theta}_t^{-i}|\bar{\theta}_t^i,I,\boldsymbol{\varphi}_t) V(s_t, \langle \Gamma_t^{-i}(\bar{\theta}_t^{-i}), q_{\tau=k}^i \rangle). \tag{16}$$

That is, given $\boldsymbol{\varphi}_t$ and Γ_t^{-i}, each $\bar{\theta}_t^i$ generates a multi-agent belief point, for which (16) performs the maximization. The set $Q_{\tau=k}^{m,i} := \{BR^i(\bar{\theta}_t^i,\boldsymbol{\varphi}_t,\Gamma_t^{-i})\}$ of best responses for all $\boldsymbol{\varphi}_t$, Γ_t^{-i} and $\bar{\theta}_t^i$, contains all non-dominated sub-tree policies, thus yielding an exact pruning step.

PBDP uses the initial belief to overcome the need to test for dominance over the entire multi-agent belief space. It can also result in more pruning, since it avoids maintaining sub-tree policies that are maximizing in a part of this space that cannot be reached. Still, the operation described above is intractable because the number of $(\bar{\theta}_t^i,\boldsymbol{\varphi}_t,\Gamma_t^{-i})$ is doubly exponential in t and because the maintained sets $Q_{\tau=k}^{m,i}$ can still grow doubly exponentially.

4.5.3 Memory-Bounded DP

Memory-bounded DP (MBDP) (Seuken and Zilberstein, 2007b) is an approximate method that addresses these complexity issues by making two approximations. This first approximation is the assumption that the joint sub-tree policies that are maximizing for a joint belief are likely to specify good candidate individual sub-tree policies. I.e., instead of performing (16) to compute candidate sub-tree policies MBDP performs:

$$\forall_{\bar{\boldsymbol{\theta}}_t} \quad \boldsymbol{q}_{\tau=k}^{\bar{\boldsymbol{\theta}}_t} = \arg\max_{\boldsymbol{q}_{\tau=k} \in \boldsymbol{Q}_{\tau=k}^e} V(I,\bar{\boldsymbol{\theta}}_t,\boldsymbol{q}_{\tau=k}), \tag{17}$$

where $\boldsymbol{Q}_{\tau=k}^e \triangleq Q_{\tau=k}^{e,1} \times \cdots \times Q_{\tau=k}^{e,n}$ is the set of $\boldsymbol{q}_{\tau=k}$ induced by the sets exhaustively backed-up sub-trees $Q_{\tau=k}^{e,i}$. If a $q_{\tau=k}^i$ is not part of any $\boldsymbol{q}_{\tau=k}^{\bar{\boldsymbol{\theta}}_t}$, it is assumed to be dominated. Note that $V(I,\bar{\boldsymbol{\theta}}_t,\boldsymbol{q}_{\tau=k})$, defined by (7), only depends on $\bar{\boldsymbol{\theta}}_t$ through the joint beliefs $\boldsymbol{b}$ it induces, so (17) only has to be

[9] The summation over states comes from substituting (7) for $V(I,\bar{\boldsymbol{\theta}}_t,\boldsymbol{q}_{\tau=k})$).

evaluated for distinct $\boldsymbol{b}$. Also note that this maximization is surprising: it was explained in Section 4.3.1 that performing this maximization for a particular node of the AOH tree is not possible. The difference here is that MBDP will not use the found $\boldsymbol{q}_{\tau=k}^{\bar{\boldsymbol{\theta}}_t}$ as the joint sub-tree policy for $\bar{\boldsymbol{\theta}}_t$ (which might result in a centralized joint policy), but rather uses it to construct sets of *individual* candidate $q_{\tau=k}^i$.

The second approximation is that MBDP maintains sets $Q_{\tau=k}^{m,i}$ of a fixed size, M, which has two main consequences. First, the size of the candidate sets $Q_{\tau=k}^{e,i}$ formed by exhaustive backup is $O(|A^\dagger|M^{|O^\dagger|})$, which clearly does not depend on the horizon. Second, (17) does not have to be evaluated for *all* distinct $\boldsymbol{b}$; rather MBDP *samples* M joint belief points $\boldsymbol{b}$ on which (17) is evaluated.[10] To perform this sampling, MBDP uses heuristic policies.

In order to perform the maximization in (17), MBDP loops over the $|\boldsymbol{Q}_{\tau=k}^e| = O(|A^\dagger|^n M^{n|O^\dagger|})$ joint sub-tree policies for each of the sampled belief points. To reduce the burden of this complexity, many papers have proposed new methods for performing this point-based backup operation (Seuken and Zilberstein, 2007a; Carlin and Zilberstein, 2008; Dibangoye et al, 2009; Amato et al, 2009; Wu et al, 2010a). This backup corresponds to solving a CBG for each joint action (Kumar and Zilberstein, 2010b; Oliehoek et al, 2010).

Finally, sample-based extensions have been proposed (Wu et al, 2010c,b). These use sampling to evaluate the quantities $V(s,\boldsymbol{q}_{\tau=k})$ and use particle representations for the sampled joint beliefs.

4.6 Other Finite-Horizon Methods

There are a few other approaches for finite-horizon Dec-POMDPs, which we will only briefly describe here. Aras et al (2007) proposed a mixed integer linear programming formulation for the optimal solution of finite-horizon Dec-POMDPs, based on representing the set of possible policies for each agent in *sequence form* (Koller and Pfeffer, 1997). In this representation, a policy for an agent i is represented as a subset of the set of *sequences* (roughly corresponding to action-observation histories) for the agent. As such the problem can be interpreted as a combinatorial optimization problem and solved with a mixed integer linear program.

The fact that solving a Dec-POMDP can be approached as a combinatorial optimization problem was also recognized by approaches based on cross-entropy optimization (Oliehoek et al, 2008a) and genetic algorithms (Eker and Akın, 2008).

[10] If evaluation of (17) leads to duplicate $q_{\tau=k}^i$ more samples may be necessary.

5 Further Topics

This section provides pointers to some further topics in Dec-POMDPs.

5.1 Generalization and Special Cases

The generalization of the Dec-POMDP is the *partially observable stochastic game (POSG)*. It has the same components as a Dec-POMDP, except that it specifies a collection of reward functions: one for each agent. A POSG assumes self-interested agents that maximize their individual expected cumulative reward. The consequence of this is that there is no longer a simple concept of optimal joint policy. Rather the joint policy should be a Nash Equilibrium (NE), and preferably a Pareto optimal NE. However, there is no clear way to identify the best one. Moreover, such an NE is only guaranteed to exist in randomized policies (for a finite POSG), which means that it is no longer possible to perform brute-force policy evaluation. Also, search methods based on alternating maximization are no longer guaranteed to converge for POSGs. The (not point-based) dynamic programming method, discussed in Section 4.5.1, applies to POSGs since it finds the set of non-dominated policies for each agent.

Because of the negative complexity results for Dec-POMDPs, much research has focused on special cases to which pointers are given below. For a more comprehensive overview the reader is referred to the texts by Pynadath and Tambe (2002); Goldman and Zilberstein (2004); Seuken and Zilberstein (2008).

Some of the special cases are formed by different degrees of observability. These range from *fully-* or *individually observable* as in a multi-agent MDP (Boutilier, 1996) to *non-observable*. In the non-observable case agents use open-loop policies and solving it is easier from a complexity point of view (NP-complete, Pynadath and Tambe 2002). Between these two extremes there are partially observable problems. One more special case has been identified, namely the *jointly observable* case, where not the individual, but the joint observation identifies the true state. A jointly observable Dec-POMDP is referred to as a *Dec-MDP*, which is a non-trivial sub-class of the Dec-POMDP for which the NEXP-completeness result holds (Bernstein et al, 2002).

Other research has tried to exploit structure in states, transitions and reward. For instance, many approaches are based on special cases of *factored Dec-POMDPs*. A factored Dec-POMDP (Oliehoek et al, 2008c) is a Dec-POMDP in which the state space is factored, i.e., a state $s = \langle x_1, \ldots, x_k \rangle$ is specified as an assignment to a number of state variables or *factors*. For factored Dec-POMDPs the transition and reward models can often be specified much more compactly by making use of Bayesian networks and additive reward decomposition (the total reward is the sum of a number of 'smaller'

reward functions, specified over a subset of agents). Many special cases have tried to exploit independence between agents by partitioning the set of state factors into individual states s_i for each agent.

One such example is the *transition- and observation-independent (TOI) Dec-MDP* (Becker et al, 2004b; Wu and Durfee, 2006) that assumes each agent i has its own MDP with local states s_i and transitions, but that these MDPs are coupled through certain *events* in the reward function: some combinations of joint actions and joint states will cause extra reward (or penalty). This work introduced the idea that in order to compute a best response against a policy π^j, an agent i may not need to reason about all the details of π^j, but can use a more abstract representation of the influence of π^j on itself. This core idea was also used in *event-driven (ED) Dec-MDPs* (Becker et al, 2004a) that model settings in which the rewards are independent, but there are certain events that cause transition dependencies. Mostafa and Lesser (2009) introduced the EDI-CR, a type of Dec-POMDP that generalizes TOI and ED-Dec-MDPs. Recently the idea of abstraction has been further explored by Witwicki and Durfee (2010), resulting in a more general formulation of influence-based policy abstraction for a more general sub-class of the factored Dec-POMDP called *temporally decoupled Dec-POMDP (TD-POMDP)* that also generalizes TOI- and ED-Dec-MDPs (Witwicki, 2011). While much more general than TOI Dec-MDPs (e.g., the local states of agents can overlap) the TD-POMDP is still restrictive as it does not allow multiple agents to have direct influence on the same state factor.

Finally, there has been a body of work on *networked distributed POMDPs (ND-POMDPs)* (Nair et al, 2005; Kim et al, 2006; Varakantham et al, 2007; Marecki et al, 2008; Kumar and Zilberstein, 2009; Varakantham et al, 2009). ND-POMDPs can be understood as factored Dec-POMDPs with TOI and additively factored reward functions. For this model, it was shown that the value function $V(\boldsymbol{\pi})$ can be additively factored as well. As a consequence, it is possible to apply many ideas from distributed constraint optimization in order to optimize the value more efficiently. As such ND-POMDPs have been shown to scale to moderate numbers (up to 20) of agents. These results were extended to general factored Dec-POMDPs by Oliehoek et al (2008c). In that case, the amount of independence depends on the stage of the process; earlier stages are typically fully coupled limiting exact solutions to small horizons and few (three) agents. Approximate solutions, however, were shown to scale to hundreds of agents (Oliehoek, 2010).

5.2 Infinite-Horizon Dec-POMDPs

The main focus of this chapter has been on finding solution methods for finite-horizon Dec-POMDPs. There also has been quite a bit of work on infinite-horizon Dec-POMDPs, some of which is summarized here.

The infinite-horizon case is substantially different from the finite-horizon case. For instance, the infinite-horizon problem is undecidable (Bernstein et al, 2002), which is a direct result of the undecidability of (single-agent) POMDPs over an infinite horizon (Madani et al, 1999). This can be understood by thinking about the representations of policies; in the infinite-horizon case the policy trees themselves should be infinite and clearly there is no way to represent that in a finite amount of memory.

As a result, research on infinite-horizon Dec-POMDPs has focused on approximate methods that use finite policy representations. A common choice is to use finite state controllers (FSCs). A side-effect of limiting the amount of memory for the policy is that in many cases it can be beneficial to allow stochastic policies (Singh et al, 1994). Most research in this line of work has proposed methods that incrementally improve the quality of the controller. For instance, Bernstein et al (2009) propose a policy iteration algorithm that computes an ϵ-optimal solution by iteratively performing backup operations on the FSCs. These backups, however, grow the size of the controller exponentially. While value-preserving transformations may reduce the size of the controller, the controllers can still grow unboundedly.

One idea to overcome this problem is *bounded policy iteration (BPI)* for Dec-POMDPs (Bernstein et al, 2005). BPI keeps the number of nodes of the FSCs fixed by applying bounded backups. BPI converges to a local optimum *given a particular controller size*. Amato et al (2010) also consider finding an optimal joint policy given a particular controller size, but instead propose a non-linear programming (NLP) formulation. While this formulation characterizes the true optimum, solving the NLP exactly is intractable. However, approximate NLP solvers have shown good results in practice.

Finally, a recent development has been to address infinite-horizon Dec-POMDPs via the planning-as-inference paradigm (Kumar and Zilberstein, 2010a). Pajarinen and Peltonen (2011) extended this approach to factored Dec-POMDPs.

5.3 Reinforcement Learning

A next related issue is the more general setting of multi-agent reinforcement learning (MARL). That is, this chapter has focused on the task of planning given a model. In a MARL setting however, the agents do not have access to such a model. Rather, the model will have to be learned on-line (model-based MARL) or the agents will have to use model-free methods. While there is a great deal of work on MARL in general (Buşoniu et al, 2008), MARL in Dec-POMDP-like settings has received little attention.

Probably one of the main reasons for this gap in literature is that it is hard to properly define the setup of the RL problem in these partially observable environments with multiple agents. For instance, it is not clear when or how

the agents will the observe rewards.[11] Moreover, even when the agents can observe the state, convergence of MARL is not well-understood: from the perspective of one agent, the environment has become non-stationary since the other agent is also learning, which means that convergence guarantees for single-agent RL no longer hold. Claus and Boutilier (1998) argue that, in a cooperative setting, independent Q-learners are guaranteed to converge to a local optimum, but not the optimal solution. Nevertheless, this method has on occasion been reported to be successful in practice (e.g., Crites and Barto, 1998) and theoretical understanding of convergence of individual learners is progressing (e.g., Tuyls et al, 2006; Kaisers and Tuyls, 2010; Wunder et al, 2010). There are coupled learning methods (e.g., Q-learning using the joint action space) that will converge to an optimal solution (Vlassis, 2007). However, all forms of coupled learning are precluded in the true Dec-POMDP setting: such algorithms require either full observation of the state and actions of other agents, or communication of all the state information.

Concluding this section we will provide pointers to a few notable approaches to RL in Dec-POMDP-like settings. Peshkin et al (2000) introduced *decentralized gradient ascent policy search (DGAPS),* a method for MARL in partially observable settings based on gradient descent. DGAPS represents individual policies using FSCs and assumes that agents observe the global rewards. Based in this, it is possible for each agent to independently update its policy in the direction of the gradient with respect to the return, resulting in a locally optimal joint policy. This approach was extended to learn policies for self-configurable modular robots (Varshavskaya et al, 2008). Chang et al (2004) also consider decentralized RL assuming that the global rewards are available to the agents. In their approach, these global rewards are interpreted as individual rewards, corrupted by noise due to the influence of other agents. Each agent explicitly tries to estimate the individual reward using Kalman filtering and performs independent Q-learning using the filtered individual rewards. The method by Wu et al (2010b) is closely related to RL since it does not need a model as input. It does, however, needs access to a simulator which can be initialized to specific states. Moreover, the algorithm itself is centralized, as such it is not directly suitable for on-line RL.

Finally, there are MARL methods for partially observed decentralized settings that require only limited amounts of communication. For instance, Boyan and Littman (1993) considered decentralized RL for a packet routing problem. Their approach, Q-routing, performs a type of Q-learning where there is only limited local communication: neighboring nodes communicate the expected future waiting time for a packet. Q-routing was extended to mobile wireless networks by Chang and Ho (2004). A similar problem, distributed task allocation, is considered by Abdallah and Lesser (2007). In this problem there also is a network, but now agents do not send communication packets, but rather tasks to neighbors. Again, communication is only local.

[11] Even in a POMDP, the agent is not assumed to have access to the immediate rewards, since they can convey hidden information about the states.

Finally, in some RL methods for multi-agent MDPs (i.e., coupled methods) it is possible to have agents observe a subset of state factors if they have the ability to communicate *locally* (Guestrin et al, 2002; Kok and Vlassis, 2006).

5.4 Communication

The Dec-POMDP has been extended to explicitly incorporate communication actions, and observations. The resulting model, the Dec-POMDP-Com (Goldman and Zilberstein, 2003, 2004) includes a set of messages that can be sent by each agent and a cost function that specifies the cost of sending each message. The goal in a Dec-POMDP-Com is to:

> "find a joint policy that maximizes the expected total reward over the finite horizon. Solving for this policy embeds the *optimal meaning* of the messages chosen to be communicated" — Goldman and Zilberstein (2003)

That is, in this perspective the semantics of the communication actions become part of the optimization problem (Xuan et al, 2001; Goldman and Zilberstein, 2003; Spaan et al, 2006; Goldman et al, 2007).

One can also consider the case where messages have fixed semantics. In such a case the agents need a mechanism to process these semantics. For instance, when the agents share their local observations, each agent maintains a joint belief and performs an update of this joint belief, rather than maintaining the list of observations. It was shown by Pynadath and Tambe (2002) that under cost-free communication, a joint communication policy that shares the local observations at each stage is optimal. Much research has investigated sharing local observations in models similar to the Dec-POMDP-Com (Ooi and Wornell, 1996; Pynadath and Tambe, 2002; Nair et al, 2004; Becker et al, 2005; Roth et al, 2005b,a; Spaan et al, 2006; Oliehoek et al, 2007; Roth et al, 2007; Goldman and Zilberstein, 2008; Wu et al, 2011).

A final note is that, although models with explicit communication seem more general than the models without, it is possible to transform the former to the latter. That is, a Dec-POMDP-Com can be transformed to a Dec-POMDP (Goldman and Zilberstein, 2004; Seuken and Zilberstein, 2008).

Acknowledgments

I would like to thank Leslie Kaelbling and Shimon Whiteson for the valuable feedback they provided and the reviewers for their insightful comments. Research supported by AFOSR MURI project #FA9550-09-1-0538.

References

Abdallah S, Lesser V (2007) Multiagent reinforcement learning and self-organization in a network of agents. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 172–179

Amato C, Carlin A, Zilberstein S (2007) Bounded dynamic programming for decentralized POMDPs. In: Proc. of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)

Amato C, Dibangoye JS, Zilberstein S (2009) Incremental policy generation for finite-horizon DEC-POMDPs. In: Proc. of the International Conference on Automated Planning and Scheduling, pp 2–9

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

Aras R, Dutech A, Charpillet F (2007) Mixed integer linear programming for exact finite-horizon planning in decentralized POMDPs. In: Proc. of the International Conference on Automated Planning and Scheduling

Becker R, Zilberstein S, Lesser V (2004a) Decentralized Markov decision processes with event-driven interactions. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 302–309

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

Becker R, Lesser V, Zilberstein S (2005) Analyzing myopic approaches for multi-agent communication. In: Proc. of the International Conference on Intelligent Agent Technology, pp 550–557

Bernstein DS, Givan R, Immerman N, Zilberstein S (2002) The complexity of decentralized control of Markov decision processes. Mathematics of Operations Research 27(4):819–840

Bernstein DS, Hansen EA, Zilberstein S (2005) Bounded policy iteration for decentralized POMDPs. In: Proc. of the International Joint Conference on Artificial Intelligence, pp 1287–1292

Bernstein DS, Amato C, Hansen EA, Zilberstein S (2009) Policy iteration for decentralized control of Markov decision processes. Journal of Artificial Intelligence Research 34:89–132

Boularias A, Chaib-draa B (2008) Exact dynamic programming for decentralized POMDPs with lossless policy compression. In: Proc. of the International Conference on Automated Planning and Scheduling

Boutilier C (1996) Planning, learning and coordination in multiagent decision processes. In: Proc. of the 6th Conference on Theoretical Aspects of Rationality and Knowledge, pp 195–210

Boyan JA, Littman ML (1993) Packet routing in dynamically changing networks: A reinforcement learning approach. In: Advances in Neural Information Processing Systems 6, pp 671–678

Buşoniu L, Babuška R, De Schutter B (2008) A comprehensive survey of multi-agent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 38(2):156–172

Carlin A, Zilberstein S (2008) Value-based observation compression for DEC-POMDPs. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 501–508

Chang YH, Ho T (2004) Mobilized ad-hoc networks: A reinforcement learning approach. In: Proceedings of the First International Conference on Autonomic Computing, pp 240–247

Chang YH, Ho T, Kaelbling LP (2004) All learning is local: Multi-agent learning in global reward games. In: Advances in Neural Information Processing Systems 16

Claus C, Boutilier C (1998) The dynamics of reinforcement learning in cooperative multiagent systems. In: Proc. of the National Conference on Artificial Intelligence, pp 746–752

Cogill R, Rotkowitz M, Roy BV, Lall S (2004) An approximate dynamic programming approach to decentralized control of stochastic systems. In: Proc. of the 2004 Allerton Conference on Communication, Control, and Computing

Crites RH, Barto AG (1998) Elevator group control using multiple reinforcement learning agents. Machine Learning 33(2-3):235–262

Dibangoye JS, Mouaddib AI, Chai-draa B (2009) Point-based incremental pruning heuristic for solving finite-horizon DEC-POMDPs. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 569–576

Eker B, Akın HL (2008) Using evolution strategies to solve DEC-POMDP problems. Soft Computing - A Fusion of Foundations, Methodologies and Applications

Emery-Montemerlo R, Gordon G, Schneider J, Thrun S (2004) Approximate solutions for partially observable stochastic games with common payoffs. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 136–143

Emery-Montemerlo R, Gordon G, Schneider J, Thrun S (2005) Game theoretic control for robot teams. In: Proc. of the IEEE International Conference on Robotics and Automation, pp 1175–1181

Goldman CV, Zilberstein S (2003) Optimizing information exchange in cooperative multiagent systems. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 137–144

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

Goldman CV, Zilberstein S (2008) Communication-based decomposition mechanisms for decentralized MDPs. Journal of Artificial Intelligence Research 32:169–202

Goldman CV, Allen M, Zilberstein S (2007) Learning to communicate in a decentralized environment. Autonomous Agents and Multi-Agent Systems 15(1):47–90

Guestrin C, Lagoudakis M, Parr R (2002) Coordinated reinforcement learning. In: Proc. of the International Conference on Machine Learning, pp 227–234

Hansen EA, Bernstein DS, Zilberstein S (2004) Dynamic programming for partially observable stochastic games. In: Proc. of the National Conference on Artificial Intelligence, pp 709–715

Kaisers M, Tuyls K (2010) Frequency adjusted multi-agent Q-learning. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 309–316

Kim Y, Nair R, Varakantham P, Tambe M, Yokoo M (2006) Exploiting locality of interaction in networked distributed POMDPs. In: Proc. of the AAAI Spring Symposium on Distributed Plan and Schedule Management

Kok JR, Vlassis N (2006) Collaborative multiagent reinforcement learning by payoff propagation. Journal of Machine Learning Research 7:1789–1828

Koller D, Pfeffer A (1997) Representations and solutions for game-theoretic problems. Artificial Intelligence 94(1-2):167–215

Koller D, Megiddo N, von Stengel B (1994) Fast algorithms for finding randomized strategies in game trees. In: Proc. of the 26th ACM Symposium on Theory of Computing, pp 750–759

Kumar A, Zilberstein S (2009) Constraint-based dynamic programming for decentralized POMDPs with structured interactions. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 561–568

Kumar A, Zilberstein S (2010a) Anytime planning for decentralized POMDPs using expectation maximization. In: Proc. of Uncertainty in Artificial Intelligence

Kumar A, Zilberstein S (2010b) Point-based backup for decentralized POMDPs: Complexity and new algorithms. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 1315–1322

Madani O, Hanks S, Condon A (1999) On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In: Proc. of the National Conference on Artificial Intelligence, pp 541–548

Marecki J, Gupta T, Varakantham P, Tambe M, Yokoo M (2008) Not all agents are equal: scaling up distributed POMDPs for agent networks. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 485–492

Mostafa H, Lesser V (2009) Offline planning for communication by exploiting structured interactions in decentralized MDPs. In: Proc. of 2009 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp 193–200

Nair R, Tambe M, Marsella S (2003a) Role allocation and reallocation in multiagent teams: towards a practical analysis. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 552–559

Nair R, Tambe M, Marsella S (2003b) Team formation for reformation in multiagent domains like RoboCupRescue. In: Kaminka G, Lima P, Roja R (eds) Proc. of RoboCup-2002 International Symposium, Lecture Notes in Computer Science

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

Nair R, Roth M, Yohoo M (2004) Communication for improving policy computation in distributed POMDPs. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 1098–1105

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

Oliehoek FA (2010) Value-based planning for teams of agents in stochastic partially observable environments. PhD thesis, Informatics Institute, University of Amsterdam

Oliehoek FA, Vlassis N (2007) Q-value functions for decentralized POMDPs. In: Proc. of The International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 833–840

Oliehoek FA, Spaan MTJ, Vlassis N (2007) Dec-POMDPs with delayed communication. In: AAMAS Workshop on Multi-agent Sequential Decision Making in Uncertain Domains

Oliehoek FA, Kooi JF, Vlassis N (2008a) The cross-entropy method for policy search in decentralized POMDPs. Informatica 32:341–357

Oliehoek FA, Spaan MTJ, Vlassis N (2008b) Optimal and approximate Q-value functions for decentralized POMDPs. Journal of Artificial Intelligence Research 32:289–353

Oliehoek FA, Spaan MTJ, Whiteson S, Vlassis N (2008c) Exploiting locality of interaction in factored Dec-POMDPs. In: Proc. of The International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 517–524

Oliehoek FA, Whiteson S, Spaan MTJ (2009) Lossless clustering of histories in decentralized POMDPs. In: Proc. of The International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 577–584

Oliehoek FA, Spaan MTJ, Dibangoye J, Amato C (2010) Heuristic search for identical payoff Bayesian games. In: Proc. of The International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 1115–1122

Ooi JM, Wornell GW (1996) Decentralized control of a multiple access broadcast channel: Performance bounds. In: Proc. of the 35th Conference on Decision and Control, pp 293–298

Osborne MJ, Rubinstein A (1994) A Course in Game Theory. The MIT Press

Pajarinen J, Peltonen J (2011) Efficient planning for factored infinite-horizon DEC-POMDPs. In: Proc. of the International Joint Conference on Artificial Intelligence, (to appear)

Paquet S, Tobin L, Chaib-draa B (2005) An online POMDP algorithm for complex multiagent environments. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems

Peshkin L (2001) Reinforcement learning by policy search. PhD thesis, Brown University

Peshkin L, Kim KE, Meuleau N, Kaelbling LP (2000) Learning to cooperate via policy search. In: Proc. of Uncertainty in Artificial Intelligence, pp 307–314

Pynadath DV, Tambe M (2002) The communicative multiagent team decision problem: Analyzing teamwork theories and models. Journal of Artificial Intelligence Research 16:389–423

Rabinovich Z, Goldman CV, Rosenschein JS (2003) The complexity of multiagent systems: the price of silence. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 1102–1103

Roth M, Simmons R, Veloso M (2005a) Decentralized communication strategies for coordinated multi-agent policies. In: Parker LE, Schneider FE, Shultz AC (eds) Multi-Robot Systems. From Swarms to Intelligent Automata, vol III, Springer, pp 93–106

Roth M, Simmons R, Veloso M (2005b) Reasoning about joint beliefs for execution-time communication decisions. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 786–793

Roth M, Simmons R, Veloso M (2007) Exploiting factored representations for decentralized execution in multi-agent teams. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 467–463

Russell S, Norvig P (2003) Artificial Intelligence: A Modern Approach, 2nd edn. Pearson Education

Seuken S, Zilberstein S (2007a) Improved memory-bounded dynamic programming for decentralized POMDPs. In: Proc. of Uncertainty in Artificial Intelligence

Seuken S, Zilberstein S (2007b) Memory-bounded dynamic programming for DEC-POMDPs. In: Proc. of the International Joint Conference on Artificial Intelligence, pp 2009–2015

Seuken S, Zilberstein S (2008) Formal models and algorithms for decentralized decision making under uncertainty. Autonomous Agents and Multi-Agent Systems 17(2):190–250

Singh SP, Jaakkola T, Jordan MI (1994) Learning without state-estimation in partially observable Markovian decision processes. In: Proc. of the International Conference on Machine Learning, Morgan Kaufmann, pp 284–292

Spaan MTJ, Gordon GJ, Vlassis N (2006) Decentralized planning under uncertainty for teams of communicating agents. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 249–256

Spaan MTJ, Oliehoek FA, Amato C (2011) Scaling up optimal heuristic search in Dec-POMDPs via incremental expansion. In: Proc. of the International Joint Conference on Artificial Intelligence, (to appear)

Szer D, Charpillet F (2006) Point-based dynamic programming for DEC-POMDPs. In: Proc. of the National Conference on Artificial Intelligence

Szer D, Charpillet F, Zilberstein S (2005) MAA*: A heuristic search algorithm for solving decentralized POMDPs. In: Proc. of Uncertainty in Artificial Intelligence, pp 576–583

Tuyls K, Hoen PJ, Vanschoenwinkel B (2006) An evolutionary dynamical analysis of multi-agent learning in iterated games. Autonomous Agents and Multi-Agent Systems 12(1):115–153

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: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems

Varakantham P, young Kwak J, Taylor ME, Marecki J, Scerri P, Tambe M (2009) Exploiting coordination locales in distributed POMDPs via social model shaping. In: Proc. of the International Conference on Automated Planning and Scheduling

Varshavskaya P, Kaelbling LP, Rus D (2008) Automated design of adaptive controllers for modular robots using reinforcement learning. International Journal of Robotics Research 27(3-4):505–526

Vlassis N (2007) A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool Publishers

Witwicki SJ (2011) Abstracting influences for efficient multiagent coordination under uncertainty. PhD thesis, University of Michigan, Ann Arbor, Michigan, USA

Witwicki SJ, Durfee EH (2010) Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In: Proc. of the International Conference on Automated Planning and Scheduling, pp 185–192

Wu F, Zilberstein S, Chen X (2010a) Point-based policy generation for decentralized POMDPs. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 1307–1314

Wu F, Zilberstein S, Chen X (2010b) Rollout sampling policy iteration for decentralized POMDPs. In: Proc. of Uncertainty in Artificial Intelligence

Wu F, Zilberstein S, Chen X (2010c) Trial-based dynamic programming for multi-agent planning. In: Proc. of the National Conference on Artificial Intelligence, pp 908–914

Wu F, Zilberstein S, Chen X (2011) Online planning for multi-agent systems with bounded communication. Artificial Intelligence 175(2):487–511

Wu J, Durfee EH (2006) Mixed-integer linear programming for transition-independent decentralized MDPs. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp 1058–1060

Wunder M, Littman ML, Babes M (2010) Classes of multiagent Q-learning dynamics with epsilon-greedy exploration. In: Proc. of the International Conference on Machine Learning, pp 1167–1174

Xuan P, Lesser V, Zilberstein S (2001) Communication decisions in multi-agent cooperation: Model and experiments. In: Proc. of the International Conference on Autonomous Agents

Zettlemoyer LS, Milch B, Kaelbling LP (2009) Multi-agent filtering with infinitely nested beliefs. In: Advances in Neural Information Processing Systems 21