A Decision-Theoretic Approach to Collaboration: Principal Description Methods and Efficient Heuristic Approximations

Frans A. Oliehoek and Arnoud Visser

Abstract. This chapter gives an overview of the state of the art in decision-theoretic models to describe cooperation between multiple agents in a dynamic environment. Making (near-) optimal decisions in such settings gets harder when the number of agents grows or the uncertainty about the environment increases. It is essential to have compact models, because otherwise just representing the decision problem becomes intractable. Several such model descriptions and approximate solution methods, studied in the Interactive Collaborative Information Systems project, are presented and illustrated in the context of crisis management.

1 Introduction

Making decisions is hard. Even though we humans make thousands of decisions a day, some decisions, especially important ones, are difficult to make. This is even more true for decision making in complex dynamic environments. This chapter focuses on such complex decision problems and particularly on situations where there are multiple decision makers or *agents* that have to cooperate.

When compared to computer systems, humans perform extremely well in making most decisions. Still, there is a growing need for the development of intelligent decision support systems and implementing cognitive abilities in autonomous artificial agents, because human decision making has its limitations. For instance, human situation awareness is characterized by structural biases and humans are

Frans A. Oliehoek
Intelligent System Laboratory Amsterdam, Science Park 107, NL 1098 XG Amsterdam, The Netherlands
e-mail: F.A.Oliehoek@uva.nl

Arnoud Visser
Intelligent System Laboratory Amsterdam, Science Park 107, NL 1098 XG Amsterdam, The Netherlands
e-mail: A.Visser@uva.nl

R. Babuska & F.C.A. Groen (Eds.): Interactive Collaborative Information Systems, SCI 281, pp. 87–124.
springerlink.com

conservative estimators as compared to applying for example Bayesian statistics in handling uncertainties [14]. As such, human decision making may be substantially improved when assisted by intelligent decision support systems [21], providing an important social and economic incentive to develop such systems with intelligent behavior.

The Interactive Collaborative Information Systems (ICIS) project focuses on the development of intelligent techniques and methods that can be applied to decision support systems. It particularly aims to improve overall quality of information processing and decision making under conditions of stress, risk and uncertainty. For systems to be effective under such circumstances, several requirements must be met, including are the capabilities to:

1. Reach a set of (predetermined) goals by influencing the environment in which the system operates.
2. Cope with changes in the dynamic environment.
3. Support reasoning with uncertainty, reasoning with risks and reasoning under a lack of knowledge, necessary because of the nondeterministic nature of the real world.

The systems under concern are connected to the real world and can observe and influence this world. Also, such systems need to make use of their experience of previous interactions with the world. That is, such a system needs capabilities of tackling the problem of *sequential decision making*, making a series of decisions over time. This chapter is concerned with methods to realize such capabilities. In particular, it focuses on decision-theoretic methods for dynamic planning, collaboration and optimization.

Following the ICIS project, the work in this chapter is illustrated by the application to crisis management situations. These results, however, can be used in other domains of application that can be characterized by decision making in complex, dynamic and nondeterministic environments.

An example of a challenging environment is the RoboCup Rescue League [40]. Inspired on the earthquake in Kobe in 1995 [75], the agent competition RoboCup Rescue simulates an earthquake in an urban environment (see Fig. 1). In this scenario, buildings collapse causes roads to get blocked and people to get trapped in the debris, damages to gas pipes cause fires to break out all over the city and parts of the communication infrastructure fails.

In this chaotic setting, teams of firefighters, police officers and ambulances have to make decisions locally, based on only limited information. The objective of these emergency response services is to minimize the casualties and structural damage. To this end, it is important to make effective plans to deal with the situation, but this is difficult due to the uncertainties. For instance, it is hard to estimate how fast the fire will spread or how many firefighting units one should allocate to a particular fire site, and how long it will take them to control the fire. Moreover, it is also important to try to improve the situational awareness, and these two goals may compete with each other. For instance, it may be necessary to trade off human capacity between fighting fire at a particular site and reconnaissance. Not performing such information-gaining

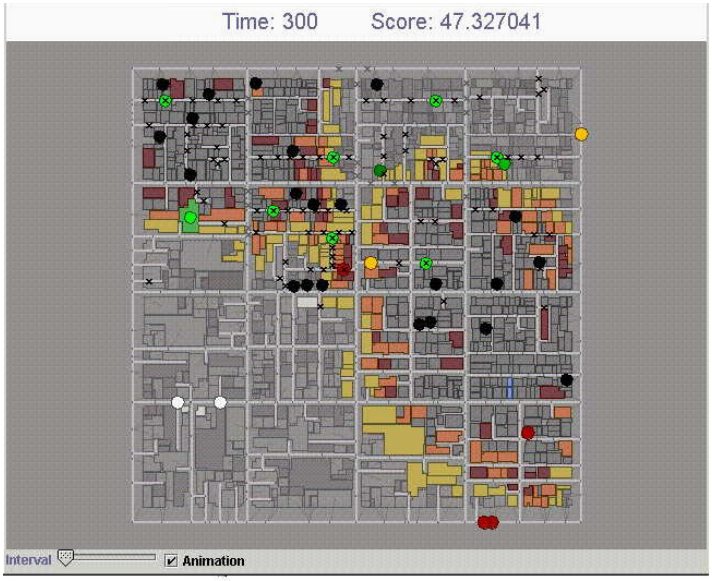


Fig. 1 Robocup Rescue simulates an earthquake in an urban environment. This is a top view of a city center, with gray buildings and white roads. Circles indicate agents; a black circle for instance indicates an agent which died due to a fire or a collapse.

activities allows allocating more agents to deal with the current situation, but may impose severe risks, e.g., a yet unknown fire source may spread out of control.

1.1 Forms of Uncertainty

The example of RoboCup Rescue illustrates how various forms of uncertainty complicate effectively resolving situations. Here, we formalize these different types of uncertainty as considered in this chapter.

The first type of uncertainty we consider is *outcome uncertainty*: the outcome or effects of actions may be uncertain. In particular, we will assume that the possible outcomes of an action are known, but that each of those outcomes is realized with some probability. This means that the state transitions are stochastic.

In the real world an agent might not be able to determine what the state of the environment exactly is, which means that there is *state uncertainty*. In such cases, we say that the environment is *partially observable*. Partial observability results from noisy and/or limited sensors. Because of *sensor noise* an agent can receive faulty or inaccurate observations. When sensors are limited the agent is unable to observe the differences between states that cannot be detected by the sensor, e.g., the presence or absence of an object outside a laser range-finder's field of view. When the same sensor reading might require different action choices, this phenomenon is referred to as *perceptual aliasing*.

Another complicating factor is the presence of multiple agents that each make decisions that influence the environment. Such an environment in which multiple agents operate is called a *multiagent system (MAS)* [70, 72, 78, 79]. The difficulty in MASs is that each agent can be uncertain regarding the actions of other agents. This is obvious in MASs with self-interested agents, where agents may be unwilling to share information. In a cooperative setting, the agents have the same goal and therefore are willing to share information and coordinate their actions. Still, it is non-trivial how such coordination should be performed. Especially when communication capabilities are limited or absent, *how* the agents should coordinate their actions becomes problematic. This problem is magnified in partially observable environments: as the agents are not assumed to observe the state—each agent only knows its own observations received and actions taken—there is no common signal they can condition their actions on. Note that this problem is in addition to the problem of partial observability, and not instead of it; even if the agents could freely and instantaneously communicate their individual observations, the joint observations would not disambiguate the true state.

1.2 Decision-Theoretic Approach to MASs

Many approaches to multiagent systems share the drawback of not having a measure of quality of the generated plans. To overcome this problem, we build upon the field of decision theory (DT).

Decision theory describes how a decision maker, or agent, should make a decision given its preferences over alternatives. Let us for the moment assume that an agent has preferences over states s of the environment. If the agents' preferences satisfy the axioms of utility theory, they can be conveniently described by a utility function u that maps each state to a real number $u(s)$ [61]. This number indicates how the agent values that state: if A is preferred to B, then $u(A) > u(B)$.

If the agent has a model of how its actions a influence the environment, i.e. when the probabilities $\Pr(s|a)$ are available, the agent can compute the *expected* utility of each action as

$$u(a) \equiv E[u(s)|a] = \sum_s u(s) \Pr(s|a). \tag{1}$$

A rational agent should select the action that maximizes this expected utility.

Decision-theoretic planning (DTP) extends DT to sequential decision problems and has roots in control theory and operations research. In control theory, one or more controllers control a stochastic system with a specific output as goal. Operations research considers tasks related to scheduling, logistics and work flow and tries to optimize the concerning systems. Many decision-theoretic planning problems can be formalized as *Markov decision processes* (MDPs). In the last decades, the MDP framework has gained in popularity in the AI community as a model for planning under uncertainty [8, 36].

The MDP is a framework for sequential decision making at predetermined points in time, i.e., it is a discrete-time model. The extension of the MDP to continuous

time is called a *semi Markov decision process (SMDP)* [56]. Also in control theory much research has considered continuous time settings [66]. In order to solve such continuous time settings, however, time is discretized (again leading to a regular MDP formulation), or special assumptions, such as linear dynamics and quadratic costs, are required [5]. As such, most approaches to DTP for multiagent systems have focused on extensions of the MDP, a notable exception is presented by Van der Broek et al. [9].

MDPs can be used to formalize a discrete-time planning task of a single agent in a stochastically changing environment, on the condition that the agent can observe the state of the environment. The environment is observed at discrete *time steps*, also referred to as *stages* [8]. The number of time steps the agent will interact with the environment is called the *horizon* of the decision problem, and will be denoted by h. Every time step the state changes stochastically, but the agent chooses an action that selects a particular transition function: i.e., taking an action from a particular state at time step t induces a probability distribution over states at time step $t + 1$. The probabilities of state transitions are specified by the model. The goal of planning for such an MDP is to find a *policy* that is optimal with respect to the desired behavior. This desired behavior is specified by the reward model: for each action taken from each possible state of the environment, a reward is issued. The agent has to maximize the expected long-term reward signal.

When the agent knows the probabilities of the state transitions, i.e., when it knows the model, it can contemplate the expected transitions over time and compute a plan that is most likely to reach a specific goal state, minimizes the expected costs or maximizes the expected reward. This stands in contrast to reinforcement learning (RL) [71], where the agent does not have a model of the environment, but has to learn good behavior by repeatedly interacting with the environment. Reinforcement learning can be seen as the combined task of learning the model of the environment *and* planning, although in practice often it is not necessary to explicitly recover the environment model.

In order to deal with the introduced sensor uncertainty, a *partially observable Markov decision process (POMDP)* extends the MDP model by incorporating observations and their probability of occurrence conditional on the state of the environment [39]. A POMDP, however, only considers one agent in an environment. We consider the setting where there are multiple agents that each may influence the state of this environment, as illustrated in Fig. 2. To incorporate the influence of multiple agents, the models need to be adapted to consider the joint effect of the individual actions, i.e., transition probabilities depend on *joint actions* and similarly for observations.

The effects of uncertainty with respect to other agents may be mitigated through communication. Under the stringent assumptions of instantaneous, cost- and noise-free communication, these effects can be discarded altogether, and the problem reduces to a POMDP [57]. However, in general, these assumptions are too strong and deciding *when* to communicate *what* becomes part of the problem.

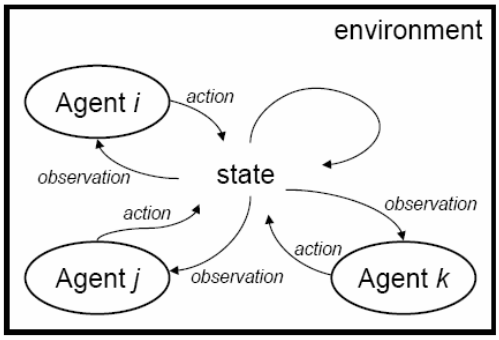


Fig. 2 A schematic representation of multiple agents in a dynamic environment. The state is influence by the combined actions of all agents.

There are two perspectives on MASs. One option is to consider each agent separately, and have each such agent maintain an explicit model of the other agents, we refer to this as the *subjective perspective* of a MAS. This is the approach as chosen in the *recursive modeling method (RMM)* [29] and the *Interactive POMDP (I-POMDP)* framework [28]. On the other hand, the *decentralized partially observable Markov decision process (Dec-POMDP)* [4] is a generalization to multiple agents of a POMDP and can be used to model a team of cooperative agents that are situated in a stochastic, partially observable environment. It presents an *objective perspective* of the MAS, in which we try to find plans for all agents at the same time.

Some other models that we do not consider in details stem from game theory. In particular, extensive games [52] can model problems of sequential decision making in uncertain environment. However, the game trees needed to model complex environments are extremely large. The more recently introduced MAIDs [44] and NIDs [25] can be more compact than extensive games, but this is especially the case with respect to the structure of the variables of influence, not with respect to decisions made over multiple stages.

1.3 Overview

This chapter gives an overview of some different decision-theoretic models for multiagent planning. First, in Section 2 we treat objective models, in particular the Dec-POMDP. Next, Section 3 presents the interactive POMDP, the most well-known subjective model. As illustrated in Section 4, there still is a gap between the state of the art in DTP and many real-world applications. Our research tries to close this gap from two sides. On the one hand, we try to scale up the problems decision-theoretic methods can handle as explained in the remainder of Section 4. On the other hand, Section 5 shows how we employ efficient heuristic methods to large realistic tasks. Finally, Section 6 concludes.

2 The Objective Approach: Dec-POMDPs

This section introduces the objective approach to planning for MASs. In particular it focuses on the decentralized POMDP [4], which is a model for objective sequential decision making for a team of cooperative agents. It is roughly equivalent to the *multiagent team decision problem (MTDP)* [57].

2.1 Decentralized POMDPs

Here, we formally define the Dec-POMDP model and its components.

Definition 1. A *decentralized partially observable Markov decision process (Dec-POMDP)* is defined as a tuple $\langle n,S,A,P_T,R,O,P_O,h,b^0 \rangle$ where:

- n is the number of agents.
- S is a finite set of states.
- A is the set of joint actions.
- P_T is the transition function.
- R is the immediate reward function.
- O is the set of joint observations.
- P_O is the observation function.
- h is the horizon of the problem.
- $b^0 \in \mathscr{P}(S)$ is the initial state distribution at time $t = 0$.[1]

The Dec-POMDP model extends single-agent (PO)MDP models by considering *joint* actions and observations. In particular, $A = \times_i A_i$ 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 $\mathbf{a} = \langle a_1,...,a_n \rangle$ is taken. 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, $O = \times_i O_i$ 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 $\mathbf{o} = \langle o_1,...,o_n \rangle$, from which each agent i only observes its own component o_i, as illustrated in Fig. 3.

Actions and observations are the interface between the agents and their environment. The Dec-POMDP framework describes this environment by its *states* and *transitions*. A Dec-POMDP specifies an environment model simply as the set of possible states $S = \{s_1,...,s_{|S|}\}$ together with the probabilities of state transitions. In particular, the transition from some state to another depends stochastically on the past states and actions. This probabilistic dependence models *outcome uncertainty*: the fact that the outcome of an action cannot be predicted with full certainty as discussed in Section 1.1.

An important characteristic of Dec-POMDPs is that the states possess the *Markov property*. That is, the probability of a particular next state depends on the current state and joint action, but not on the whole history:

[1] $\mathscr{P}(\cdot)$ denotes the set of probability distributions over $(\cdot)$.

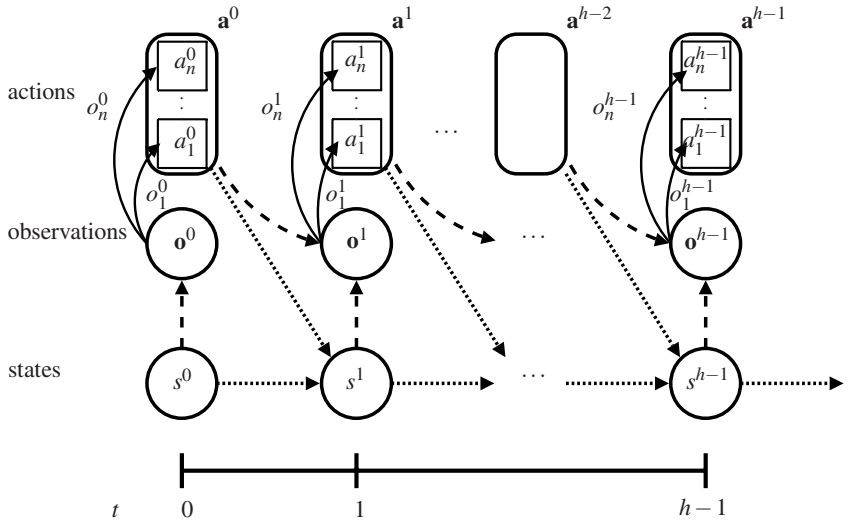


Fig. 3 An illustration of the dynamics of a Dec-POMDP. At every stage the environment is in a particular state. This state emits a joint observation, of which each agent observes its individual observation. Then each agent selects an action forming the joint action.

$$\Pr(s^{t+1}|s^t,\mathbf{a}^t,s^{t-1},\mathbf{a}^{t-1},\ldots,s^0,\mathbf{a}^0) = \Pr(s^{t+1}|s^t,\mathbf{a}^t). \tag{2}$$

Also, we will assume that the transition probabilities are *stationary*, meaning that they are independent of the stage t.

In a way similar to how the transition model P_T describes the stochastic influence of actions on the environment, the observation model describes how the state of the environment is perceived by the agents. Formally, P_O is a mapping from joint actions and successor states to probability distributions over joint observations: $P_O : A \times S \to \mathscr{P}(O)$, i.e., it specifies

$$\Pr(\mathbf{o}^t|\mathbf{a}^{t-1},s^t). \tag{3}$$

The Dec-POMDP is truly decentralized in the sense that 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 handle communication: communication can be modeled implicitly through the regular actions and observations as will be illustrated in Section 4.1.

The reward function $R : S \times A \to \mathbb{R}$ is used to specify the goal of the agents. In particular, a desirable sequence of joint actions should correspond to a high 'long-term' reward, formalized as the *return*.

Definition 2. Let the *return* or *cumulative reward* of a Dec-POMDP be defined as total of the rewards received during an execution:

$$\sum_{t=0}^{h} R(s^t, \mathbf{a}^t) \tag{4}$$

where $R(s^t, \mathbf{a}^t)$ is the reward received at time step t.

We consider as optimality criterion the *expected cumulative reward*, where the expectation refers to the expectation over sequences of states and executed joint actions. The planning problem is to find a tuple of policies, called a *joint policy* that maximizes the expected cumulative reward.

Note that, in contrast to reinforcement learning settings [71], in a Dec-POMDP, the agents are assumed not to observe the immediate rewards. Observing the immediate rewards could convey information regarding the true state which is not present in the received observations. This is undesirable as all information available to the agents should be modeled in the observations. When planning for Dec-POMDPs the only thing that matters is the *expectation* of the cumulative future reward, not the actual reward obtained.

The assumption in this text is that planning takes place in an off-line phase, after which the plans are executed in the on-line phase. In the decentralized setting, however, this statement can use some clarification. The on-line execution phase is completely decentralized: each agent only knows the joint policy as found in the planning phase and its individual history of actions and observations. The planning phase, however, is centralized: a single centralized computer computes a joint plan and consequently distributes these plans to the agents, who then merely execute the plans on-line.

Example 1 (The FIREFIGHTING *problem).* As an example, we consider a benchmark problem in which a team of n fire fighters have to extinguish fires in a row of N_H houses. Each house H is characterized by an integer parameter fl_H, or fire level. It indicates to what degree a house is burning, and it can have n_f different values, $0 \leq fl_H < n_f$. Its minimum value is 0, indicating the house is not burning.

At every time step, the agents receive a reward of $-fl_H$ for each house and each agent can choose to move to any of the houses to fight fires at that location. That is, each agent has actions $H_1 \ldots H_{N_H}$ If a house is burning ($fl_H > 0$) and no fire fighting agent is present, its fire level will increase by one point with probability 0.8 if any of its neighboring houses are burning, and with probability 0.4 if none of its neighbors are on fire. A house that is not burning can only catch fire with probability 0.8 if one of its neighbors is on fire. When two agents are in the same house, they will extinguish any present fire completely, setting the house's fire level to 0. A single agent present at a house will lower the fire level by one point with probability 1 if no neighbors are burning, and with probability 0.6 otherwise. Each agent can only observe whether there are flames (F) or not (N) at its location. Flames are observed with probability 0.2 if $fl_H = 0$, with probability 0.5 if $fl_H = 1$, and with probability 0.8 otherwise. Initially, the agents start outside any of the houses, and the fire level fl_H of each house is drawn from a uniform distribution.

In case there is only one agent, the Dec-POMDP reduces to a standard POMDP. In a POMDP, the agent still cannot observe the state, so it is not possible to specify

a policy as a mapping from states to actions as is done in an MDP. However, it turns out that maintaining a probability distribution over states, called *belief*, $b \in \mathscr{P}(S)$, is a sufficient statistic:

$$\forall_{s^t} \quad b^t(s^t) \equiv \Pr(s^t|o^t,a^{t-1},o^{t-1},\ldots,a^0,o^0). \tag{5}$$

As a result, a single agent in a partially observable environment can specify its policy as a series of mappings from the set of beliefs to actions.

Unfortunately, in the general Dec-POMDP case no such simplifications are possible. Even though the transition and observation model can be used to compute a *joint* belief, this computation requires knowledge of the joint actions and observations. During execution, the agents have no access to this information and thus can not compute such a joint belief.

2.2 Histories and Policies

The goal of planning in a Dec-POMDP is to find a (near-) optimal tuple of policies, and these policies specify for each agent how to act in a specific situation. A Dec-POMDP specifies h time steps or stages $t = 0,\ldots,h-1$. At each of these stages, there is a state s^t, joint observation $\mathbf{o}^t$ and joint action $\mathbf{a}^t$. Therefore, when the agents have to select their k-th actions (at $t = k-1$), the history has the following form:

$$\left(s^0,\mathbf{o}^0,\mathbf{a}^0,s^1,\mathbf{o}^1,\mathbf{a}^1,\ldots,s^{k-1},\mathbf{o}^{k-1}\right). \tag{6}$$

Here s^0 is the initial state, drawn according to the initial state distribution b^0. The initial joint observation $\mathbf{o}^0$ is assumed to be the empty joint observation: $\mathbf{o}^0 = \mathbf{o}_\emptyset$.

From this history of the process, the states remain unobserved and agent i can only observe its own actions and observations. Therefore, an agent will have to base its decision regarding which action to select on the sequence of actions and observations observed up to that point.

Definition 3. We define *the action-observation history (AOH) for agent i*, $\vec{\theta}_i$, as the sequence of actions taken by and observations received by agent i. At a specific time step t, this is

$$\vec{\theta}_i^t = \left(o_i^0,a_i^0,o_i^1\ldots,a_i^{t-1},o_i^t\right). \tag{7}$$

The *joint action-observation history*, $\vec{\boldsymbol{\theta}}$, is the action-observation history for all agents: $\vec{\boldsymbol{\theta}}^t = \langle\vec{\theta}_1^t,\ldots,\vec{\theta}_n^t\rangle$. Agent is set of possible AOHs at time t is $\vec{\Theta}_i^t$ and the set of all its AOHs is $\vec{\Theta}_i = \cup_{t=0}^{h-1}\vec{\Theta}_i^t$.[2] Finally, the set of all possible *joint* AOHs is given by $\vec{\Theta} = \cup_{t=0}^{h-1}(\vec{\Theta}_1^t \times \ldots \times \vec{\Theta}_n^t)$. At $t = 0$, the action-observation history is empty, denoted by $\vec{\boldsymbol{\theta}}^0 = \vec{\boldsymbol{\theta}}_\emptyset$.

[2] 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.

Definition 4. The *observation history (OH) for agent i*, $\vec{o}_i$, is the sequence of observations an agent has received. At a specific time step t, this is

$$\vec{o}_i^t = \left(o_i^0, o_i^1, \ldots, o_i^t\right). \tag{8}$$

The *joint observation history*, $\vec{\mathbf{o}}$, is the OH for all agents: $\vec{\mathbf{o}}^t = \langle \vec{o}_1^t, \ldots, \vec{o}_n^t \rangle$. Similar to the notation for AOHs, the set of OHs for agent i at time t is denoted $\vec{O}_i^t$. We also use $\vec{O}_i$ and $\vec{O}$ and the empty observation history is denoted $\vec{\mathbf{o}}_\emptyset$.

Definition 5. The *action history (AH) for agent i*, $\vec{a}_i$, is the sequence of actions an agent has performed. At a specific time step t, we write

$$\vec{a}_i^t = \left(a_i^0, a_i^1, \ldots, a_i^{t-1}\right). \tag{9}$$

Notation for joint action histories and sets are analogous to those for observation histories. Finally we note that, clearly, a (joint) action-observation history consists of a (joint) action- and a (joint) observation history: $\vec{\boldsymbol{\theta}}^t = \langle \vec{\mathbf{o}}^t, \vec{\mathbf{a}}^t \rangle$.

The action-observation history of an agent specifies all the information the agent has when it has to decide upon an action. For the moment we assume that an individual policy π_i for agent i is a deterministic mapping from action-observation histories to actions. However, the number of such histories grows exponentially with the horizon of the problem: e.g., at the last time step $h-1$, there are $(|A_i|\,|O_i|)^{h-1}$ action-observation histories for agent i. The number of policies for agent i is exponential in this number, and thus doubly exponential in the horizon h.

It is possible to reduce the number of policies under consideration by realizing that a lot of policies specify the same behavior. This is illustrated by the left side of Fig. 4, which clearly shows that under a *deterministic* policy only a subset of possible action-observation histories are reached. Policies that only differ with respect to an action-observation history that is not reached in the first place, manifest the same behavior. The consequence is that to specify a deterministic policy, the observation history suffices: when an agent takes its action deterministically, he will be able to infer what action he took from only the observation history as illustrated by the right side of Fig. 4.

Definition 6. A *pure* or *deterministic policy*, π_i, for agent i in a Dec-POMDP is a mapping from observation histories to actions, $\pi_i : \vec{O}_i \to A_i$. The set of pure policies of agent i is denoted Π_i.

Note that also for pure policies we write $\pi_i(\vec{\theta}_i)$. In this case we mean the action that π_i specifies for the observation history contained in $\vec{\theta}_i$. We use $\pi = \langle \pi_1, \ldots, \pi_n \rangle$ to denote a *joint policy* Also we use $\pi_{\neq i} = \langle \pi_1, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_n \rangle$, to denote a profile of policies for all agents but i.

Apart from pure policies, it is also possible to have the agents execute *randomized policies*, i.e., policies that do not always specify the same action for the same situation, but in which there is an element of chance that decides which action is performed. We will not consider such policies here, since in a Dec-POMDP, there always is an optimal pure joint policy [48].

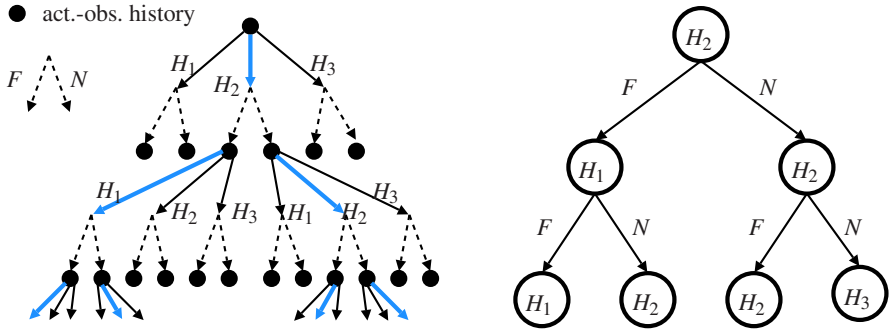


Fig. 4 Left: a tree of action-observation histories $\vec{\theta}_i$ for one of the agents in the $\langle N_H = 3, n_f = 3\rangle$ FIREFIGHTING problem. An arbitrary deterministic policy π_i is highlighted. Clearly shown is that π_i only reaches a subset of histories $\vec{\theta}_i$. ($\vec{\theta}_i$ that are not reached are not further expanded.) Right: The same policy can be shown in a simplified policy tree.

A common way to represent the temporal structure in a policy is to split it in *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 the case of a deterministic policy, the form of the decision rule for stage t is a mapping from length-t observation histories to actions $\delta_i^t : \vec{O}_i^t \to A_i$. A joint decision rule $\delta^t = \langle \delta_1^t, \ldots, \delta_n^t \rangle$ specifies a decision rule for each agent.

We will also consider policies that are partially specified with respect to time. Formally, $\varphi^t = (\delta^0, \ldots, \delta^{t-1})$ denotes the past joint policy at stage t, which is a partial joint policy π specified for stages 0,...,$t-1$.

Clearly, policies differ in how much reward they can expect to accumulate. We consider the expected cumulative reward of a joint policy, also referred to as its *value*.

Definition 7. The *value* $V(\pi)$ of a joint policy π is defined as

$$V(\pi) \equiv E\left[\sum_{t=0}^{h-1} R(s^t, \mathbf{a}^t) \middle| \pi, b^0\right], \tag{10}$$

where the expectation is over states and observations.

In particular, we can calculate this expectation recursively using

$$V_\pi^t(s^t, \vec{\mathbf{o}}^t) = R\left(s^t, \pi(\vec{\mathbf{o}}^t)\right) + \sum_{s^{t+1} \in S} \sum_{\mathbf{o}^{t+1} \in O} \Pr(s^{t+1}, \mathbf{o}^{t+1} | s^t, \pi(\vec{\mathbf{o}}^t)) V_\pi^{t+1}(s^{t+1}, \vec{\mathbf{o}}^{t+1}). \tag{11}$$

The value of joint policy π is then given by

$$V(\pi) = \sum_{s^0 \in S} V_\pi(s^0, \vec{\boldsymbol{\theta}}_\emptyset) b^0(s^0). \tag{12}$$

Example 2 (Optimal policies for the FIREFIGHTING *problem).* As an example, Fig. 5 shows an optimal joint policy for horizon 3 of the $\langle N_H = 3, n_f = 3\rangle$ FIREFIGHTING problem. One agent initially moves to the middle house to fight fires there, which helps prevent fire from spreading to its two neighbors. The other agent moves to house 3, and stays there if it observes fire, and moves to house 1 if it does not observe flames. As well as being optimal, such a joint policy makes sense intuitively speaking.

no observation → go house 3	no observation → go house 2
flames → go house 3	flames → go house 2
no flames → go house 1	no flames → go house 2
flames, flames → go house 1	flames, flames → go house 1
flames, no flames → go house 1	flames, no flames → go house 1
no flames, flames → go house 2	no flames, flames → go house 1
no flames, no flames → go house 2	no flames, no flames → go house 1

Fig. 5 An optimal joint policy for FIREFIGHTING $\langle N_H = 3, n_f = 3\rangle$, horizon 3. On the left the policy for the first agent, on the right the second agent's policy.

2.3 Solving Dec-POMDPs

The fact that the number of pure joint policies is doubly exponential in the horizon h provides some intuition about how hard the problem is. This intuition is supported by the result that the problem of finding the optimal solution for a finite-horizon Dec-POMDP with $n \geq 2$ is NEXP-complete [4]. Moreover, Dec-POMDPs cannot be approximated efficiently: even finding an ε-approximate solution is NEXP-hard [59]. Also, the problem of solving over an infinite horizon is undecidable, which is a direct result of the undecidability of (single-agent) POMDPs over an infinite horizon [45].

In the rest of this section, we provide background on the two main approaches of solving finite-horizon Dec-POMDPs: the forward view and the backward view. For a more complete overview of finite-horizon solution methods we refer to [48] and for an overview including infinite-horizon approaches to [64].

2.3.1 The Forward View: Heuristic Search

The forward view of Dec-POMDPs is given by the combination of multiagent A* [73] and the method proposed by Emery-Montemerlo [22] and the works derived from those methods. In particular, both approaches can be unified in a generalization of MAA*, creating a new view that may be called the forward perspective to solving Dec-POMDPs [48].

Multiagent A* (MAA*)

Szer et al. introduced a heuristically guided policy search method called *multiagent A* (MAA*)* [73]. It performs a guided A*-like search over partially specified joint

policies, pruning joint policies that are guaranteed to be worse than the best (fully specified) joint policy found so far by an admissible heuristic.

In particular MAA* considers joint policies that are partially specified with respect to time: a partial joint policy $\varphi^t = (\delta^0, \delta^1, \ldots, \delta^{t-1})$ specifies the joint decision rules for the first t stages. For such a partial joint policy φ^t a heuristic value $\widehat{V}(\varphi^t)$ is calculated by taking $V^{0\ldots t-1}(\varphi^t)$, the actual expected reward φ^t achieves over the first t stages, and adding $\widehat{V}^{t\ldots h-1}$, a heuristic value for the remaining $h-t$ stages. Clearly, when $\widehat{V}^{t\ldots h-1}$ is an *admissible heuristic*—a guaranteed overestimation—so is $\widehat{V}(\varphi^t)$.

MAA* starts by placing the completely unspecified joint policy φ^0 in an open list. Then, it proceeds by selecting partial joint policies $\varphi^t = (\delta^0, \delta^1, \ldots, \delta^{t-1})$ from the list and 'expanding' them: generating all $\varphi^{t+1} = (\delta^0, \delta^1, \ldots, \delta^{t-1}, \delta^t)$ by appending all possible joint decision rules δ^t for next time step (t). The left side of Fig. 6 illustrates the expansion process. After expansion, all created children are heuristically evaluated and placed in the open list. Any partial joint policies φ^{t+1} with $\widehat{V}(\varphi^{t+1})$ less than the expected value $V(\pi)$ of some earlier found (fully specified) joint policy π, can be pruned. The search ends when the list becomes empty, at which point an optimal fully specified joint policy has been found.

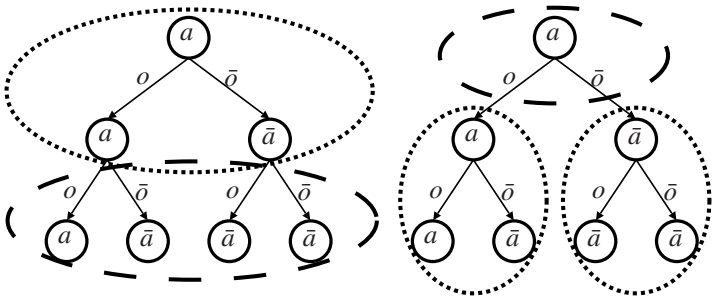


Fig. 6 Difference between policy construction in MAA* (left) and dynamic programming (right) for an agent with actions a,$\bar{a}$ and observations o,$\bar{o}$. The dashed components are newly generated, dotted components result from the previous iteration. MAA* 'expands' a partial policy from the leaves, while dynamic programming backs up a set of 'sub-tree policies' forming new ones.

Dec-POMDPs as series of Bayesian games

The MAA* algorithm can be generalized by interpreting a Dec-POMDP as a series of Bayesian games. A Bayesian game (BG) [52] is an extension of a normal form game in which the agents can hold some private information which is expressed by their *type*. BGs can be used to approximate Dec-POMDPs [22]. In this method, agents construct and solve a BG for each stage of the process in an on-line fashion. Such modeling is exact when using an optimal payoff function for the BGs [48]. Modeling Dec-POMDPs through BGs is appealing because it decomposes the recursive Dec-POMDP problem into a conceptually simpler problem for each stage.

It allows for efficient approximations, since local optima for BGs can be computed efficiently using solution methods from game theory. Also, it opens a new branch of related work and techniques that may be used, some of which are discussed in Section 4.

The crucial difficulty in making a decision at some stage t in a Dec-POMDP is that the agents lack a common signal on which to condition their actions and must rely instead on their individual (action-)observation histories. Given b^0 and φ^t, the joint policy followed for stages $0 \ldots t-1$, this situation can be modeled as a BG with identical payoffs. Such a game $BG(b^0,\varphi^t)$ consists of the set of agents $\{1 \ldots n\}$, their joint actions A, the set of their joint types Θ, a probability distribution over these joint types $\Pr(\cdot)$ and a payoff function u that maps a joint type and action to a real number $u(\boldsymbol{\theta},\mathbf{a})$. A joint type $\boldsymbol{\theta} \in \Theta$ specifies a type for each agent $\boldsymbol{\theta} = \langle \theta_1, \ldots, \theta_n \rangle$. Since the type of an agent represents the private information it holds, types are AOHs $\theta_i \equiv \vec{\theta}_i^t$ and joint types correspond to joint AOHs $\boldsymbol{\theta} \equiv \vec{\boldsymbol{\theta}}^t$. Given φ^t and b^0, the probability distribution over joint AOHs is welldefined and the payoff function is given by $u(\boldsymbol{\theta},\mathbf{a}) \equiv Q^*(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$, the optimal Q-value function of the Dec-POMDP. Although Q^* is intractable to compute, heuristic Q-value functions $\widehat{Q}$ can be computed, for instance using the underlying MDP's value function.

We can solve such a BG by computing the expected heuristic value $\widehat{V}$ for all joint BG-policies $\boldsymbol{\beta}^t = \langle \beta_1, \ldots, \beta_n \rangle$, where an individual BG-policy maps types to actions $\beta_i(\vec{\theta}_i^t) = a_i^t$. This valuation is given by

$$\widehat{V}(\boldsymbol{\beta}^t) = \sum_{\vec{\boldsymbol{\theta}}^t} \Pr(\vec{\boldsymbol{\theta}}^t | \varphi^t, b^0) \widehat{Q}(\vec{\boldsymbol{\theta}}^t, \boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t)), \tag{13}$$

where $\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t) = \langle \beta_i(\vec{\theta}_i^t) \rangle_{i=1\ldots n}$ denotes the joint action that results from application of the individual BG-policies to the individual AOHs $\vec{\theta}_i^t$ specified by $\vec{\boldsymbol{\theta}}^t$. The solution $\boldsymbol{\beta}^{t,*}$ is the joint BG-policy with the highest expected value. Note that if φ^t is deterministic, the probability of $\vec{\boldsymbol{\theta}}^t = \langle \vec{\mathbf{a}}^t, \vec{\mathbf{o}}^t \rangle$ is non-zero for only one $\vec{\mathbf{a}}^t$ per $\vec{\mathbf{o}}^t$ ($\vec{\mathbf{a}}^t$ can be reconstructed from $\vec{\mathbf{o}}^t, \varphi^t$). Therefore, in effect the BG-policies reduce to decision rules: mappings from observation histories to actions $\beta_i(\vec{o}_i^t) = a_i^t$.

Generalized MAA*

The modeling of a stage of a Dec-POMDP as a BG as outlined above can be applied in a heuristic policy search scheme called Generalized MAA* (GMAA*) [48], which generalizes MAA* and the BG-method of [22]. Algorithm 3 shows GMAA*, which maintains an open list P of partial joint policies φ^t and their heuristic values $\widehat{V}(\varphi^t)$. Every iteration the highest ranked φ^t is selected and expanded, i.e., the Bayesian game $BG(\varphi^t, b^0)$ is constructed and all joint BG-policies $\boldsymbol{\beta}^t$ are evaluated. Consequently, these joint BG-policies are used to construct a new set of partial policies

$$\Phi_{\texttt{Next}} := \{\varphi^{t+1} = (\varphi^t, \boldsymbol{\beta}^t)\} \tag{14}$$

Algorithm 3 GMAA*

$\underline{v}^\star \leftarrow -\infty$
$P \leftarrow \{\varphi^0 = ()\}$
repeat
  $\varphi^t \leftarrow$ SelectHighestRankedPartialJPol(P)
  $\Phi_{\text{new}} \leftarrow$ ConstructAndSolveBG(φ^t, b^0)
  if Φ_{new} contains full policies $\Pi_{\text{new}} \subseteq \Phi_{\text{new}}$ **then**
    $\pi' \leftarrow \arg\max_{\pi \in \Pi_{\text{new}}} V(\pi)$
    if $V(\pi') > \underline{v}^\star$ **then**
      $\underline{v}^\star \leftarrow V(\pi')$ {found new lower bound}
      $\pi^\star \leftarrow \pi'$
      $P \leftarrow \{\varphi \in P \mid \widehat{V}(\varphi) > \underline{v}^\star\}$ {prune P}
    end if
    $\Phi_{\text{new}} \leftarrow \Phi_{\text{new}} \setminus \Pi_{\text{new}}$ {remove full policies}
  end if
  $P \leftarrow (P \setminus \varphi^t) \cup \{\varphi \in \Phi_{\text{new}} \mid \widehat{V}(\varphi) > \underline{v}^\star\}$ {remove processed/add new partial policies}
until P is empty

and their heuristic values. When the heuristic values are an upper bound to the true values, any lower bounds $\underline{v}^\star$ (i.e., full joint policies) that are found can be used to prune P. When P becomes empty, the optimal policy has been found.

GMAA* as outlined here is MAA* reformulated to work on BGs. The BG-method of [22] is similar, but does not backtrack, i.e., rather than constructing all new partial policies $\forall_{\boldsymbol{\beta}^t}\ \varphi^{t+1} = (\varphi^t, \boldsymbol{\beta}^t)$ only the best-ranked partial policy $(\varphi^t, \boldsymbol{\beta}^{t,*})$ is constructed and the open list P will never contain more than 1 partial policy. A generalization k-GMAA* constructs the k best-ranked partial policies, allowing to trade off computation time and solution quality.

2.3.2 The Backward View: Dynamic Programming

GMAA* incrementally builds policies from the first stage $t = 0$ to the last $t = h-1$. Dynamic programming (DP) for Dec-POMDPs [37] constructs policies the other way around: starting with a set of '1-step policies' (actions) that can be executed at the last stage, they construct a set of 2-step policies to be executed at $h-2$, etc.

It should be stressed that the policies maintained are quite different from those used by GMAA*. In particular, a partial policy in GMAA* has the form $\varphi^t = (\delta^0, \delta^1, \ldots, \delta^{t-1})$. The policies maintained by DP do not have such a correspondence to decision rules. We define the *time-to-go* τ at stage t as

$$\tau = h - t. \tag{15}$$

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 maintained for agent i is

denoted $Q_i^{\tau=k}$. Dynamic programming for Dec-POMDPs is based on backup operations: constructing $Q_i^{\tau=k+1}$ from $Q_i^{\tau=k}$. For instance, the right side of Fig. 6 shows how $q_i^{\tau=3}$, a 3-steps-to-go sub-tree policy, is constructed from two $q_i^{\tau=2} \in Q_i^{\tau=2}$. Also illustrated is the difference between this process and MAA* expansion (on the left side).

Dynamic programming consecutively constructs $Q_i^{\tau=1}, Q_i^{\tau=2}, \ldots, Q_i^{\tau=h}$ for all agents i. However, the size of the set $Q_i^{\tau=k+1}$ is given by

$$|Q_i^{\tau=k+1}| = |A_i| \, |Q_i^{\tau=k}|^{|O_i|}, \tag{16}$$

and as a result the sizes of the maintained sets grow doubly exponential with k. To counter this source of intractability, Hansen et al. [37] propose to eliminate dominated sub-tree policies. The expected reward of a particular sub-tree policy $q_i^{\tau=k}$ depends on the probability over states when $q_i^{\tau=k}$ 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 $q_j^{\tau=k} \in Q_j^{\tau=k}$. If we let $q_{\neq i}^{\tau=k}$ denote a sub-tree profile for all agents but i, and $Q_{\neq i}^{\tau=k}$ the set of such profiles, we can say that $q_i^{\tau=k}$ is dominated if it is not maximizing at any point in the *multiagent belief* space: the simplex over $S \times Q_{\neq i}^{\tau=k}$. Hansen et al. test for dominance over the entire multiagent belief space by linear programming. Removal of a dominated sub-tree policy $q_i^{\tau=k}$ of an agent i may cause a sub-tree policy $q_j^{\tau=k}$ of an other agent j to become dominated. Therefore, they propose to iterate over agents until no further pruning is possible, a procedure known as *iterated elimination of dominated policies* [52].

Finally, when the last backup step is completed the optimal policy can be found by evaluating all joint policies $\pi \in Q_1^{\tau=h} \times \cdots \times Q_n^{\tau=h}$ for the initial belief b^0.

2.4 Special Cases and Generalization

Because of the negative complexity results for Dec-POMDPs, much research has focused on special cases of Dec-POMDPs. This section reviews some of these. For a more comprehensive overview of special cases, the reader is referred to [32, 57, 64].

2.4.1 Factored Dec-POMDPs

A *factored* Dec-POMDP has a state space $S = X_1 \times \ldots \times X_{|X|}$ that is spanned by $X = \{X_1, \ldots, X_{|X|}\}$ a set of state variables, or *factors*. A state corresponds to an assignment of values for all factors $s = \langle x_1, \ldots, x_{|X|} \rangle$. In a factored Dec-POMDP, the transition and observation model can be compactly represented by exploiting conditional independence between variables. In particular, the transition and observation model can be represented by a dynamic Bayesian network (DBN) [8].

Even though factored Dec-POMDPs can be represented more compactly, solving them is still hard. However, by adding additional assumptions to the factored Dec-POMDP model, more efficient solutions are sometimes possible [2, 32, 64].

2.4.2 Degrees of Observability

The literature has identified different categories of observability [32, 57]. When the observation function is such that the individual observation for each of the agents will always uniquely identify the true state, the problem is considered *fully-* or *individually observable*. In such a case, a Dec-POMDP effectively reduces to a *multi-agent Markov decision process (MMDP)* [7].

In this setting a (joint) action can be selected based on the state without considering the history, because the state is Markovian. Moreover, because each agent can observe the state, there is an effective way to coordinate. One can think of the situation as a regular MDP with a 'puppeteer' agent that selects joint actions. For this 'underlying MDP' an optimal solution π^* can be found efficiently (P-complete [53]) with standard dynamic programming techniques [56]. Such a solution $\pi^* = (\delta^0, \ldots, \delta^{h-1})$ specifies a mapping from states to joint actions for each stage $\forall_t\ \delta^t : S \to A$ and can be split into individual policies $\pi_i = \left(\delta_i^0, \ldots, \delta_i^{h-1}\right)$ with $\forall_t\ \delta_i^t : S \to A_i$ for all agents.

The other extreme is when the problem is *non-observable*, meaning that none of the agents observes any useful information. This is modeled by the fact that agents always receive a null-observation, $\forall_i\ O_i = \{o_{i,\emptyset}\}$. Under non-observability agents can only employ an open-loop plan. A result of this is that non-observable setting is easier from a complexity point of view (NP-complete [57]).

Between these two extremes, there are partially observable problems. One more special case has been identified, namely, the case where not the individual, but the joint observation identifies the true state. This case is referred to as *jointly-* or *collectively observable*.

Definition 8. A jointly observable Dec-POMDP is referred to as a *Dec-MDP*.

Even though all observation together identify the state in a Dec-MDP, each agent still has a partial view. As such Dec-MDPs are a non-trivial sub-class of Dec-POMDPs for which the NEXP-completeness result holds [3].

2.4.3 Explicit Communication

The Dec-POMDP model does not explicitly define communication. Of course, it is possible to include communication actions and observations in the regular set of actions and observations, so a Dec-POMDP can handle communication implicitly. The Dec-POMDP, however, has been extended to incorporate explicitly communication actions and observations. The resulting model the Dec-POMDP-Com [31, 32] additionally includes a set of messages Σ that can be sent by each agent and a cost function C_Σ that specifies the cost of sending each message. The MTDP has a similar extension, called the Com-MTDP [57], which is equivalent to the Dec-POMDP-Com.

Although the Dec-POMDP-Com model itself could allow different communication models, studies so far have considered noise-free instantaneous broadcast communication. That is, at a stage in the process each agent broadcasts its message and receives the messages sent by all other agents instantaneously and without errors.

In the most general case, 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 [31]

That is, in this perspective the semantics of the communication actions become part of the optimization problem. This problem is considered in [31, 67, 80].

One can also consider the case where messages have fixed semantics. In such a case the agents need a mechanism to process these semantics (i.e., to allow the messages to affect their beliefs). 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 that under cost-free communication, a joint communication policy that shares the local observation at each stage is optimal [57].

Models with explicit communication seem more general than the models without, but (in the absence of special mechanisms to interpret the messages) it is possible to transform the former to the latter. That is, a Dec-POMDP-Com can be transformed to a Dec-POMDP [32, 64].

2.4.4 Generalization: Partially Observable Stochastic Games (POSGs)

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 not a single reward function, but a collection of reward functions, one for each agent. This means that a POSG assumes self-interested agents that each want to maximize their individual expected cumulative reward.

The consequence of this is that there is no longer an optimal joint policy, simply because 'optimality' no longer defined. Rather the joint policy to suggest should be a (Bayesian) Nash Equilibrium, and preferably a Pareto optimal NE. However, there is no clear way to identify the 'best' one. Moreover, such a Pareto optimal 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.

3 The Subjective Approach

The Dec-POMDP presents an objective perspective in which the goal is to find a plan, a (near-) optimal joint policy, for all agents simultaneously. In the subjective perspective, we reason for one particular protagonist agent. That is, this agent tries to take the best actions by predicting the other agents in the environment.

This perspective has been computationally formalized in the *recursive modeling method (RMM)* [29, 30]. The limitation of the RMM framework, however, is that the agents interact in the setting of repeated strategic games, and such games do not take into account the environment, or observations of agents. As such it is difficult

to model many realistic settings, where agents have different information regarding the environment.

The *interactive POMDP (I-POMDP)* is a different framework that overcomes these limitations [28]. It is closely related to the POMDP in how it models the environment. The I-POMDP focuses on the decision making process of a single self-interested agent, situated in an environment with other agents. Gradually, dependent on the so-called strategy level, the influence of other agents is incorporated into the decisions of the protagonistic agent. Because of this, the framework also admits non-cooperative and competitive MASs.

3.1 Interactive POMDPs

When planning from the perspective of a protagonist agent in a POMDP-like environment that includes other agents, an obvious first approach is to simply ignore the other agents. The problem with this approach is that the result of its action and observations are influenced by other agents.

It is possible to treat this influence as noise, but that this approach has two main drawbacks: first, this approximation decreases the value of the optimal policy. Second, as these approximations are incorporated in the transition- and observation model, they are the same for each time-step. During execution, however, the other agent's policy might be non-stationary, thus the probability of another agent performing some action might depend on the timestep. This non-stationarity is actually guaranteed to take place when the other agent learns of its experience, and especially when it tries to learn and predict the behavior of the protagonist agent.

Therefore, instead of approximating other agents' influence through noise, Gmytrasiewicz and Doshi [28] propose to predict the behavior of other agents. To do this, they maintain an *interactive belief* over worldstates and models of other agents. A model m_i for an agent i from the set of possible models M_i describes the internal state and probabilities of future actions of agent i.

Definition 9. Formally, each model $m_i \in M_i$ for agent i is defined as a triple $m_i = \langle \vec{o}_i, \pi_i, P_{O_i} \rangle$, where $\vec{o}_i$ is the history of observations of agent i, π_i is the policy (referred to as 'function') of agent i and P_{O_i} is agent i's observation function.

Furthermore, Gmytrasiewicz and Doshi divide the set of models into two classes: *sub-intentional* and *intentional* models. The former being simpler, the latter being more complex by attributing beliefs, preferences and rationality to the agent. The intentional model that is considered in I-POMDPs is an agent's *type*:

Definition 10. A *type* θ_i of an agent i that participates in a 'POMDP-like' environment is a tuple $\langle b_i, \hat{\theta}_i \rangle$, where $b_i \in \mathscr{P}(S)$ is the belief over states of agent i and $\hat{\theta}_i = \langle A_i, P_{T_i}, R_i, O_i, P_{O_i}, OC_i \rangle$ is called agent i's *frame*. This frame in turn consists of $A_i, P_{T_i}, R_i, O_i, P_{O_i}$ the actions, transition- and reward function, observations and observation function for agent i, and OC_i, the optimality criterion of agent i: usually the cumulative (discounted) reward. A type is an element of the space of intentional models: $\theta_i \in M_i^{intentional}$.

Given these definitions, we can give the definition of an I-POMDP as follows:

Definition 11. Formally, an *interactive-POMDP (I-POMDP)* of agent i is a tuple $\langle IS_i, A, P_{T_i}, R_i, O_i, P_{O_i} \rangle$, where:

- IS_i is the set of *interactive states*, defined as $IS_i \equiv S \times (\times_{j \neq i} M_j)$.
- A is the set of joint actions.
- $P_{T_i}, R_i, O_i, P_{O_i}$ are the transition- and reward function, observations and observation function for agent i. These are defined over joint actions, i.e. $P(s'|s,\mathbf{a})$ and $R(s,\mathbf{a})$, but over individual observations, i.e. $P(o_i|\mathbf{a},s')$.

An I-POMDP is a POMDP defined over interactive states, which include models of the other agents. This means that an interactive state fully specifies the future behavior of the other agents. As such, it is possible to transform the POMDP belief update to the I-POMDP setting [28]. Since it is rather complex, we do not repeat this transformed belief update here, but just mention that to predict the actions of the other agents, it uses probabilities $\forall_j \, P(a_j|\theta_j)$ given by the model m_j.

When considering intentional models (types), the formal definition of I-POMDPs as above leads to an infinite hierarchy of beliefs, because an I-POMDP for agent i defines its belief over models and thus types of other agents, which in turn define a belief over the type of agent i, etc. To overcome this problem one can define *finitely nested I-POMDPs* [28]. Here, a 0-th level belief for agent i, $b_{i,0}$, is a belief over world-states S. An k-th level belief $b_{i,k}$ is defined over world-states and models consisting of types that admit beliefs of (up to) level $k-1$. The actual number of levels that the finitely nested I-POMDP admits is called the *strategy level.*

Example 3 (The FIREFIGHTING *problem as an I-POMDP).* Here we illustrate how the FIREFIGHTING problem may be represented as a finitely nested I-POMDP. To ease the following discussion we assume that there are two agents: i and j.

Suppose we define two sub-intentional models for agent j: one model m_{H_1} specifies that agent j will always perform action H_1, the other m_{rand} assumes that agent j will act random; i.e.,will select its actions with uniform probability. Together these models form the set $M_{j,0}$ of 0th-level models[3] of agent j and this set can be used to construct a 1st-level I-POMDP model for agent i. In this I-POMDP $IP_{i,1}$,

- the interactive state space $IS_{i,1} = S \times M_{j,0}$ is the product of the set of world states S, which is defined in the same way as in the Dec-POMDP case (i.e., the state specifies the fire levels of each of the houses), and the possible models,
- the set of joint actions is equal to the Dec-POMDP case: it is the cross product of the sets of individual actions $\{H_1 \ldots H_{N_H}\}$,
- the transition, observation and reward function are also defined in the same way as in the Dec-POMDP case.

Because each of the possible models $m \in M_{j,0}$ fully specifies the behavior of agent j, i.e., it specifies $\Pr(a_j|m)$, the interactive state possesses the Markov property and $IP_{i,1}$ can be solved.

[3] Clearly, this is a very simple example. A more realistic scenario might include more sub-intentional models.

Now in turn $IP_{i,1}$ can be interpreted as a model for agent i. In particular, given an interactive belief for $b_{i,1} \in IS_{i,1}$ the behavior of agent i is completely specified. This means that we can use it as a model $m_{i,1} = \langle b_{i,1}, IP_{i,1} \rangle$ for agent i in the interactive 2nd level state space $IS_{j,2} = S \times M_{i,1}$ for agent j, etc.

A final note in this example is that by updating the belief over interactive states in the belief update not only the belief over states is updated, but also the belief over the models of the other agent. For instance, consider a FIREFIGHTING problem where the observation spaces are augmented to include the observation of the other agent if they happen to fight fire at the same house. In this case, when agent i observes agent j at a house different than house 1, it learns that the model m_{H_1} is incorrect and will update its beliefs to reflect this, thereby improving future predictions of agent j.

A graphical model formulation of I-POMDPs called *interactive dynamic influence diagram (I-DID)* has also been proposed [20]. This model allows for a more compact representation of sequential decision problems by making explicit the structure of state variables and their influences. Like the I-POMDP itself, it gives a subjective view from a particular agent.

Such a subjective approach is very appropriate for systems with self-interested agents, but can also be applied to cooperative agents. The subjective perspective also allows for flexibility, for instance in systems where agents may come and go. Also, the subjective approach can be used to allow efficient approximations, such as demonstrated in Section 5.2.

3.2 Solving I-POMDPs

As mentioned in Section 3.1, an I-POMDP actually considers reasoning from one agent's perspective, which is why it is very similar to single agent planning.

For a finitely nested I-POMDP, the reasoning the agent performs in fact is of the form "what would the other agents do if they think that I think that...". A finitely nested I-POMDP can be solved bottom up [28]: An I-POMDP of strategic level 0 for agent i is a regular POMDP that can be solved. This gives a level 0 policy. Now, assuming that agent i can only have certain particular beliefs (over states, since it is a level-0 model), this induces a probability distribution over actions, which can be used in the level 1 I-POMDP of agent j, effectively transforming the latter to a POMDP as well. The solution (a level 1 I-POMDP policy π_j) can then be used for the level 2 I-POMDP for the agent i, etc.

The problem in the above procedure lies in the assumption that the modeled agent's belief must be a member of some particular finite set of beliefs, instead of the infinite set of beliefs. Even though one can argue that both agents have the same initial belief b^0 and that therefore the number of (0-th level) beliefs over states is finite, the number of possible models of other agents is infinite. Effectively this means that an arbitrary discretization of the interactive belief space has to be made. To overcome this problem, it is possible to group models in behavioral equivalence classes [60]. This induces a partition of the infinite interactive state space and thus leads to a finite representation of the interactive state space that allows for exact solutions.

3.3 The Complexity of Solving I-POMDPs

For finitely nested I-POMDPs, similar results hold as for POMDPs: value iteration is guaranteed to converge and the value function is piecewise linear and convex [28]. The authors also state that solving a finitely nested I-POMDP of strategy level l is PSPACE-hard, because it is equivalent to solving $O(M^l)$ POMDPs, *assuming that the number of models considered at each level is bounded by a number M*. However, the space of models considered can be very large or even infinite. Moreover, when updating the belief of another agent's model, we have to consider what observations the other agent may have received and introduce a new resulting model for each of them. This means that the number of considered models grows exponentially with the planning horizon.

4 Application of Decision-Theoretic Models and the Need to Scale up

In this section, we give an example of the application of decision-theoretic models to a realistic crisis management task, namely, RoboCup Rescue. The resulting model is huge and current methods are still far away from solving such a model. We argue that this is typical for the application of decision-theoretic tools to real-world problems and that it motivates the search for methods that scale better.

Consequently, we address some different methodologies that contribute to realizing better scaling for MASs under outcome and state uncertainty. Most of these methods have been proposed for Dec-POMDPs. Most work on I-POMDPs has focused on making the I-POMDP belief update more tractable [17, 18], clustering models [60, 81], or extending single-agent POMDP techniques [19].

4.1 An Example: RoboCup Rescue as a Dec-POMDP

Here, we present an example that shows how firefighting in RoboCup Rescue can be modeled as a factored Dec-POMDP and that the resulting model is intractable.[4] It would also be possible to model it as an I-POMDP, but similar scaling problems would result.

4.1.1 State Description

Here, we describe a formal model of the state space for the task of fire fighting in RoboCup Rescue. It is based on the dynamic factors in the RoboCup Rescue world, restricted to the factors relevant for firefighting. Static factors, i.e. factors that do not change throughout the simulation, will not have to be incorporated in the state space. In RoboCup Rescue, the world is given by a map, consisting of buildings,

[4] This is a simplified summary of the description in [50].

roads and nodes, which act as the glue between roads and buildings. A typical map consists of approximately 1000 buildings and the same number of roads and nodes.

This world is populated by civilians, rescue agents (platoons) and the immobile rescue centers. The rescue agents (both mobile and center agents) can in turn be divided into fire, police and ambulance agents. The center agents function as communication centers. We only consider (about 15) mobile firefighting agents here.

The mobile agents can be located on roads, nodes or in buildings. So the number of valid positions on a map is the sum of these elements (i.e., typically around 3000). Also these mobile agents have a particular health, expressed in health points (HPs). Fire-brigade agents have a particular amount of water left.

The earthquake that takes place at the beginning of the simulation has a large effect on the state: buildings collapse and catch fire, these collapses can cause roads to get blocked (in either direction) and civilians to get trapped in debris. Starting from this initial state, fires will spread if left unattended. The fire simulator is based on heat energy as primary concept. Fire propagation's main component is heat radiation, which is simulated by dividing the open (non-building) area of the map in (approx. 20000) cells for which the air temperature is computed. Other factors that determine the spread of fire are properties of buildings. The dynamic factors are the heat of a building, whether it is burning, how much fuel is left and how much water is put in by extinguish actions. Static properties like the size and composition of a particular building (wood, steel or reinforced concrete) and how close it is to surrounding buildings also influence the spreading of fire. However, because these properties are static, they do not need to be incorporated in the state description. Instead the probability of a particular building i catching fire given that a neighboring building j is on fire is modeled through the transition model.

Table 1 summarizes the state factors. Note that, as this example focuses on the firefighting aspect, certain factors (e.g. the position, health and other properties of other agents) are ignored.

4.1.2 Actions

The actions for an agent i in RoboCup Rescue can be divided into domain level actions A_i^d and communication actions A_i^c. A mobile agent can perform both a domain level action as communication within one timestep (e.g. a fire-brigade agent can move/extinguish and communicate). This means that the set of actions A_i for a mobile agent i is the Cartesian product of all domain level and communication actions $A_i = A_i^d \times A_i^c$. In this section, we will discuss the domain level actions. Section 4.1.4 will deal with communication actions.

All mobile agents can perform the *move* action. The argument of this *move* action is a path along which the agent should move. Clearly, the *move* actions are dependent on the current position of the agent. Also, there is a maximum distance that an agent can travel in one timestep (333m). This means that two paths that deviate only after this point lead to the same action. Fire-brigades have two specialized actions: *extinguish* and *refill*. The *extinguish* action specifies a building and the amount of water (in liters) to direct to that building.The *refill* action restores the

Table 1 State variables or *factors* for a particular RoboCup Rescue world

factor	values
for all fire agents in the world (±15)	
current position	valid positions
health	0–9999HPs
amount of water	0–15000l
for all roads (±1000)	
blocked? (2 directions)	blocked/free
for all buildings (±1000)	
heat energy	0–10^6 GJ
state	(not) burning
fuel left	percent(%)
amount of water	0–150000l
for all air cells (±20000)	
temperature	20-10000°C

water supply of the brigade and can only be performed at 'refuges'; these are special buildings where agents can find shelter.

4.1.3 Observations

As with actions we specify the set of observations for agent i as the Cartesian product of domain and communication observations $O_i = O_i^d \times O_i^c$. Here we treat the domain observations, communication observations are treated in section 4.1.4.

At each timestep, only objects within a range of 10 m are seen, except for fiercely burning buildings, which can be observed from a larger distance. When an agent executed a *move* action, only observations of the new position are received (i.e. no observations are made 'en route'). On average 4–6 static objects (building and roads) can be visually observed during a timestep [54].

Observing an object means that the agent receives the object ID, its type and properties. For a road this property is whether or not it is blocked (in both ways), for a building, the so-called fieriness, is observed. This fieriness factor is a direct function of the amount of fuel and water left in the building and determines the part of the area counted as damaged.

4.1.4 Communication

Communication consists of both an action (by the sender) and an observation (for the receiver). In RoboCup Rescue there are two forms of communication actions: *say* and *tell*. The *say* messages are directly transferred (i.e., shouted), the latter transmitted by radio. Both types of communication are broadcast: *say* messages can be picked up by agents of all types within 30 m, *tell* messages can be received by all agents of the same type regardless of the distance. The restrictions that are posed

on communication vary per competition. We assume that platoon agents can send 4 *tell* messages and one *say* message and that all agents can receive all messages. Restrictions on the number of received messages can also be incorporated [50].

In a Dec-POMDP, we can model communication by introducing communication actions and observations. The basic idea is that for each joint communication action $\mathbf{a}^c$ one joint communication observation $\mathbf{o}^c$ can be introduced that for each agent contains the messages sent by the other agents. Restrictions with respect to communication distance can be modeled by making communication dependent on the (next) state s'. That is, it is possible to specify a communication model of the form $\Pr(\mathbf{o}^c|\mathbf{a}^c,s')$.

The complete observation model is then given as the product of this communication model and the regular, domain observation model:

$$\Pr(\langle\mathbf{o}^d,\mathbf{o}^c\rangle|\langle\mathbf{a}^d,\mathbf{a}^c\rangle,s') = \Pr(\mathbf{o}^c|\mathbf{a}^c,s')\cdot\Pr(\mathbf{o}^d|\mathbf{a}^d,s'). \tag{17}$$

In a pure planning framework, messages have no a priori semantics. Instead the planning process should embed the 'optimal meaning' in each communication action as explained in Section 2.4.3 In RoboCup Rescue all messages are 256 bytes. When an agent can send 4 *tell* and 1 *say* message, it has $8\cdot256\cdot5 = 10240$ bits to encode its communication action and thus that $|A^c_i| = 2^{10240}$. This means that the number of joint communication actions $|A^c| = 2^{10240n}$. Clearly this way of treating communication is intractable.

4.1.5 Transition, Observation and Reward Model

The transition model of a factored Dec-POMDP can be compactly described by a two-stage DBN. Because the state description is the same as used by the simulator components, these structures and probabilities for this DBN can be found by analyzing the code of the simulation system. For the (domain) observation model we can make a similar argument.

The reward function is easily derived from the scoring function. A typical scoring function is

$$Score(s) = (P + S/S_0)\cdot\sqrt{B/B_0}, \tag{18}$$

where P is the number of living agents, S_0 is the total sum of health points (HPs) at start of the simulation, S is the remaining sum of HPs, B_0 is the total area of houses and B is the area of houses that remained undamaged.

This gives us the reward function of the Dec-POMDP in the following way:

$$R(s,\mathbf{a},s') = R(s,s') = Score(s') - Score(s). \tag{19}$$

The horizon is finite in the RoboCup Rescue competition (300 time-steps). However, in the real-life setting we will typically want to plan for a varying horizon (until all fire is extinguished and all trapped people are either rescued or dead). This can be accomplished by treating the problem as one of infinite horizons.

4.1.6 Complexity

Here, we will give a brief argument of the complexity of the Dec-POMDP representation. By ignoring some factors of the statespace, we effectively performed a first abstraction to reduce the state space. However, the state space as presented in Table 1 is still huge. When there are $n = 15$ fire-brigade agents and 3000 valid positions this already leads to 3000^{15} different configurations. When considering only the first four factors, we already get a state space of

$$\begin{aligned} |nr\_pos|^{15} \cdot |HPs|^{15} \cdot |water|^{15} \cdot 2^{|2 \cdot nr\_roads|} &= \\ 3000^{15} \cdot 10000^{15} \cdot 15000^{15} \cdot 2^{2000} &\approx \\ 10^{52} \cdot 10^{60} \cdot 10^{62} \cdot 10^{602} &= 10^{776} \end{aligned}$$

and this is not even including the state of each of the 1000 buildings. We already saw that the number of joint communication actions is prohibitively large and the same holds for domain actions and observations. Therefore, this is clearly an intractable problem.

In the above, we modeled RoboCup Rescue as a Dec-POMDP and concluded that the resulting model is intractable to solve. Even though this is just an example, it is exemplary for the current state of affairs: when modeling a realistic problem using decision-theoretic tools, the result is typically an intractable problem. On the one hand, this motivates the need for decision-theoretic methods that scale better. On the other hand, some research tries to directly find heuristic solutions that perform well, thereby trying to closer the gap between principled solutions and real-life applications from both sides. In Section 5, we will provide some examples of our work in the latter category. The rest of this section focuses on ways to scale up decision-theoretic approaches. In particular, we give an overview of hierarchical decompositions, exploiting independence between agents and compressions of policies. We focus on aspects of MASs, but stress that scaling up methods for single-agent (PO)MDPs is still a very important area of research [36].

4.2 Aggregation and Hierarchical Decompositions

Some common sense reveals that the intractability of the flat description of the previous section is no surprise: In the real world, a fireman who is extinguishing some building typically will not care about properties of a building beyond the zone of potential fire spreading. Nor will he consider what the exact position of a colleague at the other side of town is. Effectively, in order for a firefighter to perform his job he needs to focus on those state variables that are relevant for his current task. States that do not differ on relevant variables can be grouped or *aggregated*. In such a way state aggregations reduce the size of the state space [15, 24, 27, 55].

Aggregation is closely related to hierarchical methods [1, 16, 68]. That is, these above insights may be used to provide a hierarchical decomposition of tasks, bringing leverage to the decision process, by constraining the number of possible policies. An example how such a decomposition might look for the RoboCup Rescue

Dec-POMDP is given in [50]: At the lowest level each group of burning buildings (fire zone) may be considered separately, coordination between different fire zones is performed at a higher district level and the highest level considers the entire city. So far, most other hierarchical approaches have focussed on single-agent and fully observable settings, typically using the framework of SMDPs. An important direction of future research is the examination of more principled hierarchical methods for partially observable MASs.

4.3 Modeling and Exploiting Independence between Agents

In a complex MAS, not all agents may need to consider each other's actions. A hierarchical decomposition may make this assumption explicit by having some agents participate in different parts of the decomposition (e.g. different fire zones in the example above). On a smaller scale, however, there may also be independence between agents. For instance consider a slightly modified version of FIREFIGHTING, called FIREFIGHTINGGRAPH, illustrated in Fig. 7. This version of the problem restricts each agent to only go to any of two fixed houses that are assigned to it. In such a setting it is conceivable that agent 1 only has to directly coordinate with agent 2, but not with agent 3.

In the last decade, much research has been devoted to exploiting such independence between agents. However, many of these approaches make very restrictive assumptions. For instance, models with more assumptions on observability and/or communication have been considered [35, 42]. These approaches try to exploit independence between agents, but they either require the agents to observe the state of the system (as in a multiagent MDP) or to observe a subset of state variables (as in a factored Dec-MDP) and communicate at every stage. For the particular case of transition and observation independent Dec-POMDPs [2], it has been suggested to exploit independence [47, 76]. However, the assumption of transition and

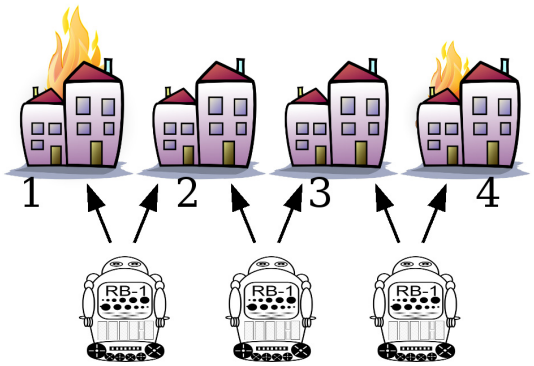


Fig. 7 An illustration of the FIREFIGHTINGGRAPH problem. Each agent is restricted to go fight fire at the houses on its left and right sides.

observation independence (TOI) severely limits the applicability of these models. In particular, TOI implies that the agents have disjoint sets of individual states and observations, and that one agent cannot influence the state transition or the observation of another agent. In practice, this means that many interesting tasks, such as two robots carrying an object, cannot be modeled.

A more general analysis of locality of interaction in factored Dec-POMDPs shows that the last stage in the process contains the highest degree of independence but, when moving back in time (towards the initial stage $t = 0$), the scope of dependence grows [49]. Still, the authors are able to exploit independence between agents in the optimal solution of factored Dec-POMDPs, because most of the independence is located where it is most needed: the overwhelming majority of the computational effort of solving a Dec-POMDP is spent in the last stage [73]. The method of [49] is an extension of GMAA*, which replaces the BGs by *collaborative graphical BGs (CGBGs)*, which are less computationally expensive to solve in later stages where independence is high. Even though they can only exploit independence in the last stage to guarantee optimality (GMAA* needs to return *all* partial policies for intermediate stages to guarantee optimality), an experimental evaluation shows a speedup of two orders of magnitude.

4.4 Compression of the Considered Policy Space

A third category of techniques to facilitate better scaling tries to compress directly the space of policies that is considered. The first example of this type of approach is presented in [22] that proposes to prune low-probability joint types (i.e., joint action-observation histories with a low probability) from the BGs constructed for each stage. Subsequently, the smaller BGs are easier to solve, since there are a lot less joint BG-policies (i.e., decision rules for the stage for which the BG is constructed) to consider. As such, the search space of joint policies is effectively trimmed by pruning in the space of histories. This approach is refined by clustering histories with similar (heuristic) payoff profiles, rather than pruning [23].

Even though these pruning and clustering techniques proved to be quite effective, they are a form of lossy compression of the BG and no quality guarantees are available. Recently, a criterion has been identified that *guarantees* that two individual histories have the same optimal value, allowing *lossless clustering* and therefore faster optimal solutions of Dec-POMDPs [51]. The authors experimentally demonstrate that in several well-known test problems, the proposed method allows for the optimal solution of significantly longer horizons.

In the backward view of solving Dec-POMDPs similar techniques have been employed. Probably the most successful method here is memory-bounded dynamic programming (MBDP) [63]. Rather than pruning dominated sub-tree policies $q_i^{\tau=k}$, MBDP prunes all sub-tree policies except a few in each iteration. More specifically, for each agent m sub-tree policies that perform well on a set of heuristically sampled multiagent belief points are retained. Effectively this means that at every iteration the search space of policies is trimmed to a fixed size. As a result, MBDP has only

linear space and time complexity with respect to the horizon. The MBDP algorithm still depends on the exhaustive generation of the sets $Q_i^{\tau=k+1}$ which now contain $|A_i|\, m^{|O_i|}$ sub-tree policies. Moreover, in each iteration all $\left(|A_*|\, m^{|O_*|}\right)^n$ joint sub-tree policies have to be evaluated for each of the sampled belief points. To counter this growth, an extension is proposed that limits the considered observations during the backup step to the *maxObs* most likely observations [62]. MBDP with observation compression (MBDP-OC) [10] improves upon this by not pruning low-probability observations, but rather clustering observations in a manner that minimizes the expected loss.

MBDP-OC is able to bound the error introduced relative to MBDP without observation compression. The error introduced by MBDP itself, however, is unbounded. Recently, an optimal algorithm based on DP that performs a lossless compression of the sub-tree policies was introduced [6]. The idea is that sub-tree policies are represented in a smaller more compact form, similar to *sequence form* [43] and especially the so-called *tests* from the work on predictive state representations [65]. Policy compression results in significant improvement in performance compared to regular DP for Dec-POMDPs.

5 Efficient Heuristic Approaches for Teams of Agents

The decision-theoretic models covered in this chapter have some desirable properties. They are well defined and allow for quantitative comparisons of solutions. Unfortunately, the solution methods scale poorly. The previous section described some techniques that can improve scalability. Even though this is a step in the right direction, pure decision-theoretic solutions are still impractical for most real-world settings.

Our strategy has been to tackle the gap between decision-theory and real-world applications from two sides. This section describes work that approaches it from the application side using efficient heuristic methods. First, we describe a body of work that employs a limited form of decision-theoretic reasoning to allocate heuristically defined roles using variants of the Dec-POMDP. Second, we describe how a heuristic variant of the subjective approach to decision theory can be employed for the task of exploration in emergency sites.

5.1 Allocating Pre-specified Roles Sensibly

Over the last decades, many heuristic approaches to multiagent systems have been proposed and employed with considerable success in (near-) real-life settings [38, 58, 69, 74]. Of these approaches, many are based on the belief-desire-intention (BDI) paradigm [26] and extensions for MASs [11–13, 33, 34]. In such 'BDI teams' agents reason over their beliefs, desires and intentions and that of the team to select a plan, or *role* from a library of pre-compiled plans.

These heuristic approaches have been able to scale to larger problems than the decision-theoretic approaches treated in earlier sections. Still, the latter have some

desirable properties. As such, some research has tried to integrate DTP within these BDI-teams to improve performance, by applying it to substitute only a part of the reasoning. In particular, the task of role allocation has been considered.

For instance, decision-theoretic reasoning has been applied to assign roles to agents in the context of the RoboCup Soccer simulation league [41]. Their approach is based on using coordination graphs to indicate which agents should coordinate when select a role. Nair and Tambe [46] consider role allocation for TOP plans [58]. They define the role-based multiagent team decision problem (RMTDP) which is an extension of the Dec-POMDP to include role taking and role executing actions. The RMTDP is constructed and solved more efficiently by exploiting properties of the TOP plan. For instance, the features tested in the TOP plan correspond to the set of states S of the RMTDP. More important, the organization hierarchy of the TOP plan is used to calculate upper bounds for the expected reward when a role is chosen. This upper bound is used to prune less attractive role allocations. For this algorithm, experimental results on a simplified RoboCup Rescue scenario are presented.

5.2 Frontier Selection in Exploration

Multi-robot exploration is a challenging application area in which multiple robots have to explore an area as efficiently as possible. An instance of this problem can be found in the RoboCup Rescue League: inside the Virtual Robot competition multiple robots have to coordinate their actions to explore jointly an as large as possible area after a disaster. At the beginning, the robots have no knowledge of the environment, but by driving around and observing, it can be explored. Exploration can be formally defined as the task to maximize the knowledge about the environment. When the environment is modeled with an occupancy grid map, the exploration problem is the problem of maximizing the cumulative information of each grid cell.

One of the difficulties in multi-robot exploration is that information about the environment is distributed; each robot has a partial and incomplete view represented by the occupancy grid map it individually retains. Most of the knowledge about the environment is directly derived from the sensors of the robot, only occasionally augmented with some additional (older) observations from the other robots when they are relayed over a multi-hop communication network. However, communication may not be available at all times. Also, for a setting like this it is infeasible to pre-compute a plan describing what to do in all possible settings, since this would entail finding an exploration plan for every possible world. For these reasons, a subjective approach as outlined in Section 3 is more suitable for this type of problems.

In particular, the exploration problem as introduced can be formalized as an I-POMDP. In this representation, each robot maintains its private occupancy grid map. The full state vector contains both s_p the true state of the physical environment (which includes the agent positions) as well as s_i the maintained grid of each agent i. Domain-level actions of robots typically consist of low-level actuations. However, in this setting it is possible to use a higher abstraction level by defining ‘exploration frontiers’. Such frontiers are the boundaries between explored and

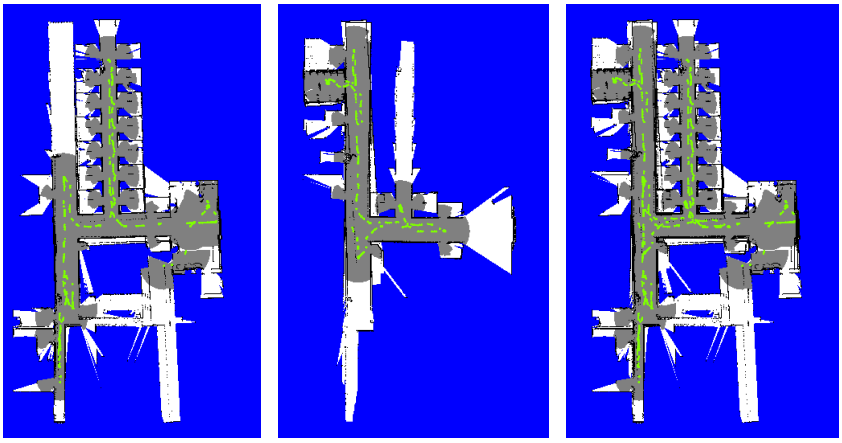
unexplored areas as specified by each individual occupancy grid. Action a_{ij} of agent i dictates that it goes to frontier f_j, the j-th frontier specified by its grid. Selecting a frontier means a decision on the action to take. Given s_p the state of the physical environment and the actions selected by the agents, a new physical state s'_p results stochastically. Each agent makes a local observation of this new state s'_p which allows them to update (their belief of) their individual maps s'_i, as well as their beliefs over the maps maintained by other agents. Any received communication may be incorporated in these beliefupdates. The reward function can be defined as the difference in cumulative information between two states s and s'.

The above defines all necessary components for an I-POMDP, except the intentional models. A sensible level-0 sub-intentional model for an agent is to go to the nearest frontier on its map m_{near}. We do not define alternative sub-intentional models, so $M_{j,0} = \{m_{near}\}$ for each agent j. As before, the interactive states of a level-1 I-POMDP for agent i can now be defined as $IS_{i,1} = S \times M_{j,0}$. Here, a $s \in S$ is a full state (specifies the physical states and the grid of each agent). Each interactive belief of such an agent i fully specifies its behavior and therefore serves as a model for a level-2 I-POMDP for agent j, etc.

In theory, for each agent i we could build such an I-POMDP model and the resulting behavior should be coordinated. Of course, such an I-POMDP representation is intractable; the number of interactive states even for an I-POMDP of level 1 is given by product of the number of physical states s_p (i.e., possible worlds) and the number of possible maps the other agent. Still the I-POMDP formulation as presented can serve as a guideline for efficient heuristic subjective methods. In particular, the number of interactive states can be greatly reduced by simply dropping the maps of other agents j from the state description for an I-POMDP of agent i. A side effect is that the sub-intentional model of an agent j is no longer welldefined: it specifies to go to the nearest frontier according to agent j's grid map, but the state no longer includes this map of agent j. Therefore, agent i assumes that agent j has the same map as itself. This approximation may work well, because the agents that need to coordinate closely are those that are physically near, which in turn means that they are likely to have similar maps.

A second issue is the infinite number of physical worlds that the robots could explore. At the end, the reasoning over hidden world states performed in an (interactive) POMDP has as its goal to derive expected future reward values for each action, which can then be used to select the best action. Now the idea is to more directly use a heuristic function to valuate each of the possible individual actions. The heuristic which used is based on the physical properties of the observation process. The difference between explored and unexplored areas is a gradual one. Some observations beyond the frontiers are already available, which can be used the estimate the future reward [77]. In particular, we use an estimate of the information gained by each possible individual action by looking at the area visible beyond the frontier.

By predicting the other agents' actions (by assuming that they go to the frontier nearest to them) each individual action a_i leads to a joint action. This joint action leads to an expected increase in information, by adding up the size of the areas beyond those frontiers. Such an approximation may work well, because in a



(a) Map generated by one robot (b) Map generated by another robot (c) combined Map

Fig. 8 Map resulting from an autonomous exploration in the RoboCup Rescue 2006 Competition

exploration setting, each non-explored world typically is equally likely, i.e., there typically is no good Bayesian prior that gives a better estimate of the value of each action than the sparse observations beyond the frontier.

Note that assuming the other agents go to the nearest frontier may result in miscoordination since in fact they will employ similar reasoning. To reduce the probability of such miscoordinations, prediction of the other agents is enhanced by assuming that the other agents also trade off information gain and travel cost. Frontiers are allocated to robots in order, where the combination which has the highest estimated information gain / travel cost balance is selected first.

The result is a heuristic exploration method inspired by the I-POMDP formulation given above. Experimental evaluations demonstrate that the method results in coordinated behavior where each robot explores a different part of the environment. An example of such cooperation is given in Fig. 8, where the exploration effort of two robots is indicated with their individual maps and the shared map. Gray areas indicate fully explored areas, and white areas indicate to be explored areas. The boundaries between white and gray areas are the frontiers. It is clear that the overlap in the exploration effort of both robots is small.

6 Conclusions

In this chapter, an overview is given of a number of decision-theoretic tools and models to allow reasoning with uncertainty, reasoning with risks and reasoning under a lack of knowledge. The decentralized POMDP framework is described as a model for objective sequential decision making for a team of cooperative agents and

compared with an alternative subjective approach; interactive POMDP. For both models we described solution methods and the known complexity results.

We also modeled a more realistic task as a Dec-POMDP and illustrated that the resulting model is prohibitively large. Therefore, we argue that there still is a big gap between decision-theoretic methods and real-life applications. In our opinion, this gap should be closed from both sides: by scaling up principled solutions and identifying efficient heuristics. In the former category we discussed some techniques that may allow for the solution of larger Dec-POMDPs. In the latter, we gave an example of how Dec-POMDPs can be applied to tackle a smaller part of the problem (role assignment), and a heuristic approach for the task of exploration. Since the approach to exploration takes a subjective perspective, it can be interpreted as an approximation to an I-POMDP. Future work should investigate whether such approximations are possible for a broader class of partially observable multiagent problems.

Acknowledgements. We wish to thank our colleagues: Julian de Hoog, Julian Kooij, Matthijs Spaan, Nikos Vlassis and Shimon Whiteson.

References

1. Barto, A.G., Mahadevan, S.: Recent advances in hierarchical reinforcement learning. Discrete event dynamic systems: Theory and applications 13, 343–379 (2003)
2. Becker, R., Zilberstein, S., Lesser, V., Goldman, C.V.: Solving transition independent decentralized Markov decision processes. Journal of Artificial Intelligence Research 22, 423–455 (2004)
3. Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: The complexity of decentralized control of Markov decision processes. Mathematics of Operations Research 27(4), 819–840 (2002)
4. Bernstein, D.S., Zilberstein, S., Immerman, N.: The complexity of decentralized control of Markov decision processes. In: Proc. of Uncertainty in Artificial Intelligence, pp. 32–37 (2000)
5. Bertsekas, D.P.: Dynamic Programming and Optimal Control, 3rd edn., vol. I. Athena Scientific, Belmont (2005)
6. Boularias, A., Chaib-draa, B.: Exact dynamic programming for decentralized POMDPs with lossless policy compression. In: Proc. of the International Conference on Automated Planning and Scheduling (2008)
7. Boutilier, C.: Planning, learning and coordination in multiagent decision processes. In: Proc. of the 6th Conference on Theoretical Aspects of Rationality and Knowledge, pp. 195–210 (1996)
8. Boutilier, C., Dean, T., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intelligence Research 11, 1–94 (1999)
9. van den Broek, B., Wiegerinck, W., Kappen, B.: Graphical models inference in optimal control of stochastic multi-agent systems. Journal of Artificial Intelligence Research 32, 95–122 (2008)
10. Carlin, A., Zilberstein, S.: Value-based observation compression for DEC-POMDPs. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp. 501–508 (2008)

11. Cohen, P.R., Levesque, H.J.: Intention is choice with commitment. Artificial Intelligence 42(3), 213–261 (1990)
12. Cohen, P.R., Levesque, H.J.: Confirmations and joint action. In: Proc. of the International Joint Conference on Artificial Intelligence, pp. 951–957. Morgan Kaufmann, San Francisco (1991)
13. Cohen, P.R., Levesque, H.J.: Teamwork. Nous 25(4) (1991)
14. Dawes, R.M.: Rational Choice in an Uncertain World. Hartcourt Brace Jovanovich (1988)
15. Dean, T., Givan, R.: Model minimization in Markov decision processes. In: Proc. of the National Conference on Artificial Intelligence, pp. 106–111 (1997)
16. Dietterich, T.G.: Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research 13, 227–303 (2000)
17. Doshi, P.: Approximate state estimation in multiagent settings with continuous or large discrete state spaces. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, p. 13 (2007)
18. Doshi, P., Gmytrasiewicz, P.J.: A particle filtering based approach to approximating interactive POMDPs. In: Proc. of the National Conference on Artificial Intelligence, pp. 969–974 (2005)
19. Doshi, P., Perez, D.: Generalized point based value iteration for interactive POMDPs. In: Proc. of the National Conference on Artificial Intelligence, pp. 63–68 (2008)
20. Doshi, P., Zeng, Y., Chen, Q.: Graphical models for interactive POMDPs: representations and solutions. Autonomous Agents and Multi-Agent Systems 18(3), 376–416 (2008)
21. Druzdzel, M.J., Flynn, R.R.: Decision Support Systems. In: Encyclopedia of Library and Information Science. The Taylor & Francis, Inc., New York (2003)
22. Emery-Montemerlo, R., Gordon, G., Schneider, J., Thrun, S.: 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 (2004)
23. Emery-Montemerlo, R., Gordon, G., Schneider, J., Thrun, S.: Game theoretic control for robot teams. In: Proc. of the IEEE International Conference on Robotics and Automation, pp. 1175–1181 (2005)
24. Feng, Z., Hansen, E.: An approach to state aggregation for POMDPs. In: AAAI 2004 Workshop on Learning and Planning in Markov Processes – Advances and Challenges, pp. 7–12 (2004)
25. Gal, Y., Pfeffer, A.: Networks of influence diagrams: A formalism for representing agents' beliefs and decision-making processes. Journal of Artificial Intelligence Research 33, 109–147 (2008)
26. Georgeff, M.P., Pell, B., Pollack, M.E., Tambe, M., Wooldridge, M.: The belief-desire-intention model of agency. In: Rao, A.S., Singh, M.P., Müller, J.P. (eds.) ATAL 1998. LNCS (LNAI), vol. 1555, pp. 1–10. Springer, Heidelberg (1999)
27. Givan, R., Dean, T., Greig, M.: Equivalence notions and model minimization in Markov decision processes. Artificial Intelligence 147(1-2), 163–223 (2003)
28. Gmytrasiewicz, P.J., Doshi, P.: A framework for sequential planning in multi-agent settings. Journal of Artificial Intelligence Research 24, 49–79 (2005)
29. Gmytrasiewicz, P.J., Durfee, E.H.: A rigorous, operational formalization of recursive modeling. In: Proc. of the International Conference on Multiagent Systems, pp. 125–132 (1995)
30. Gmytrasiewicz, P.J., Noh, S., Kellogg, T.: Bayesian update of recursive agent models. User Modeling and User-Adapted Interaction 8(1-2), 49–69 (1998)

31. Goldman, C.V., Zilberstein, S.: Optimizing information exchange in cooperative multi-agent systems. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp. 137–144 (2003)
32. Goldman, C.V., Zilberstein, S.: Decentralized control of cooperative systems: Categorization and complexity analysis. Journal of Artificial Intelligence Research 22, 143–174 (2004)
33. Grosz, B.J., Kraus, S.: Collaborative plans for complex group action. Artificial Intelligence 86(2), 269–357 (1996)
34. Grosz, B.J., Sidner, C.: Plans for discourse. In: Intentions in Communication. MIT Press, Cambridge (1990)
35. Guestrin, C., Koller, D., Parr, R.: Multiagent planning with factored MDPs. In: Advances in Neural Information Processing Systems, vol. 14, pp. 1523–1530 (2002)
36. Guestrin, C., Koller, D., Parr, R., Venkataraman, S.: Efficient solution algorithms for factored MDPs. Journal of Artificial Intelligence Research 19, 399–468 (2003)
37. Hansen, E.A., Bernstein, D.S., Zilberstein, S.: Dynamic programming for partially observable stochastic games. In: Proc. of the National Conference on Artificial Intelligence, pp. 709–715 (2004)
38. Jennings, N.R.: Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. Artificial Intelligence 75(2), 195–240 (1995)
39. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101(1-2), 99–134 (1998)
40. Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjoh, A., Shimada, S.: RoboCup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research. In: Proc. of the International Conference on Systems, Man and Cybernetics, pp. 739–743 (1999)
41. Kok, J.R., Spaan, M.T.J., Vlassis, N.: Non-communicative multi-robot coordination in dynamic environments. Robotics and Autonomous Systems 50(2-3), 99–114 (2005)
42. Kok, J.R., Vlassis, N.: Collaborative multiagent reinforcement learning by payoff propagation. Journal of Machine Learning Research 7, 1789–1828 (2006)
43. Koller, D., Megiddo, N., von Stengel, B.: Fast algorithms for finding randomized strategies in game trees. In: Proc. of the 26th ACM Symposium on Theory of Computing, pp. 750–759 (1994)
44. Koller, D., Milch, B.: Multi-agent influence diagrams for representing and solving games. Games and Economic Behavior 45(1), 181–221 (2003)
45. Madani, O., Hanks, S., Condon, A.: 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 (1999)
46. Nair, R., Tambe, M.: Hybrid BDI-POMDP framework for multiagent teaming. Journal of Artificial Intelligence Research 23, 367–420 (2005)
47. Nair, R., Varakantham, P., Tambe, M., Yokoo, M.: Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In: Proc. of the National Conference on Artificial Intelligence, pp. 133–139 (2005)
48. Oliehoek, F.A., Spaan, M.T.J., Vlassis, N.: Optimal and approximate Q-value functions for decentralized POMDPs. Journal of Artificial Intelligence Research 32, 289–353 (2008)
49. Oliehoek, F.A., Spaan, M.T.J., Whiteson, S., Vlassis, N.: Exploiting locality of interaction in factored POMDPs. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, December 2008, pp. 517–524 (2008)

50. Oliehoek, F.A., Visser, A.: A hierarchical model for decentralized fighting of large scale urban fires. In: Proc. of the AAMAS 2006 Workshop on Hierarchical Autonomous Agents and Multi-Agent Systems (H-AAMAS), pp. 14–21 (2006)
51. Oliehoek, F.A., Whiteson, S., Spaan, M.T.J.: Lossless clustering of histories in decentralized POMDPs. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp. 577–584 (2009)
52. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. The MIT Press, Cambridge (1994)
53. Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of Markov decision processes. Mathematics of Operations Research 12(3), 441–451 (1987)
54. Post, S., Fassaert, M.: A communication and coordination model for 'RoboCupRescue' agents. Master's thesis, University of Amsterdam (2004)
55. Poupart, P.: Exploiting structure to efficiently solve large scale partially observable Markov decision processes. Ph.D. thesis, Department of Computer Science, University of Toronto (2005)
56. Puterman, M.L.: Markov Decision Processes—Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., Chichester (1994)
57. Pynadath, D.V., Tambe, M.: The communicative multiagent team decision problem: Analyzing teamwork theories and models. Journal of Artificial Intelligence Research 16, 389–423 (2002)
58. Pynadath, D.V., Tambe, M.: An automated teamwork infrastructure for heterogeneous software agents and humans. Autonomous Agents and Multi-Agent Systems 7(1-2), 71–100 (2003)
59. Rabinovich, Z., Goldman, C.V., Rosenschein, J.S.: 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 (2003)
60. Rathnasabapathy, B., Doshi, P., Gmytrasiewicz, P.: Exact solutions of interactive POMDPs using behavioral equivalence. In: Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems, pp. 1025–1032 (2006)
61. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Pearson Education, London (2003)
62. Seuken, S., Zilberstein, S.: Improved memory-bounded dynamic programming for decentralized POMDPs. In: Proc. of Uncertainty in Artificial Intelligence (2007)
63. Seuken, S., Zilberstein, S.: Memory-bounded dynamic programming for DEC-POMDPs. In: Proc. of the International Joint Conference on Artificial Intelligence, pp. 2009–2015 (2007)
64. Seuken, S., Zilberstein, S.: Formal models and algorithms for decentralized decision making under uncertainty. Autonomous Agents and Multi-Agent Systems 17(2), 190–250 (2008)
65. Singh, S., James, M.R., Rudary, M.R.: Predictive state representations: a new theory for modeling dynamical systems. In: Proc. of Uncertainty in Artificial Intelligence, pp. 512–519 (2004)
66. Sontag, E.D.: Mathematical control theory: deterministic finite dimensional systems, 2nd edn. Textbooks in Applied Mathematics. Springer, New York (1998)
67. Spaan, M.T.J., Gordon, G.J., Vlassis, N.: 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 (2006)
68. Stone, P.: Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer. MIT Press, Cambridge (2000)

69. Stone, P., Veloso, M.: Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. Artificial Intelligence 110(2), 241–273 (1999)
70. Stone, P., Veloso, M.: Multiagent systems: A survey from a machine learning perspective. Autonomous Robots 8(3), 345–383 (2000)
71. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, Cambridge (1998)
72. Sycara, K.P.: Multiagent systems. AI Magazine 19(2), 79–92 (1998)
73. Szer, D., Charpillet, F., Zilberstein, S.: MAA*: A heuristic search algorithm for solving decentralized POMDPs. In: Proc. of Uncertainty in Artificial Intelligence, pp. 576–583 (2005)
74. Tambe, M.: Towards flexible teamwork. Journal of Artificial Intelligence Research 7, 83–124 (1997)
75. Tierney, K.J., Goltz, J.D.: Emergency response: Lessons learned from the kobe earthquake. Tech. rep., Disaster Research Center (1997), http://dspace.udel.edu:8080/dspace/handle/19716/202
76. Varakantham, P., Marecki, J., Yabu, Y., Tambe, M., Yokoo, M.: 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 (2007)
77. Visser, A., Xingrui-Ji, van Ittersum, M., González Jaime, L.A., Stancu, L.A.: Beyond frontier exploration. In: Visser, U., Ribeiro, F., Ohashi, T., Dellaert, F. (eds.) RoboCup 2007: Robot Soccer World Cup XI. LNCS (LNAI), vol. 5001, pp. 113–123. Springer, Heidelberg (2008)
78. Vlassis, N.: A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence. In: Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, San Francisco (2007)
79. Wooldridge, M.: An Introduction to MultiAgent Systems. Wiley, Chichester (2002)
80. Xuan, P., Lesser, V., Zilberstein, S.: Communication decisions in multi-agent cooperation: Model and experiments. In: Proc. of the International Conference on Autonomous Agents (2001)
81. Zeng, Y., Doshi, P., Chen, Q.: Approximate solutions of interactive dynamic influence diagrams using model clustering. In: Proc. of the National Conference on Artificial Intelligence, pp. 782–787 (2007)