Value-Based Planning for Teams of Agents in Stochastic Partially Observable Environments

Frans A. Oliehoek

Lay out: Typeset by the author himself using LaTeX 2$_\varepsilon$.
Cover design: René Staelenberg, Amsterdam.

ISBN 9789056296100
e-ISBN 9789048512300
NUR 983

Value-Based Planning for Teams of Agents in Stochastic Partially Observable Environments

Academisch Proefschrift

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. D.C. van den Boom
ten overstaan van een door het college voor promoties
ingestelde commissie,
in het openbaar te verdedigen in de Agnietenkapel
op vrijdag 12 februari 2010, te 14:00 uur

door

Frans Adriaan Oliehoek

geboren te Utrecht

Promotiecommissie

Promotor:	Prof. dr. ir. F. C. A. Groen
Co-promotor:	Dr. N. Vlassis
Overige leden:	Prof. dr. R. Babuska
	Prof. dr. H. J. van den Herik
	Prof. dr. L. P. Kaelbling
	Prof. dr. H. J. Kappen
	Dr. M. A. Wiering

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

This research has been performed at the IAS group of the university of Amsterdam and is part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant nr: BSIK03024.

Contents

Chapter 1

Introduction

Making decisions is hard. Even though we humans make thousands of decisions a day, most of which we do not even explicitly think about, some decisions are difficult to make. Especially when there is uncertainty in the information on which the decision is based and when the impact of a decision is great, we start to doubt. This is even more true for decision making in complex dynamic environments, in which the consequences of decisions are difficult to anticipate. This thesis focuses on such complex decision problems and particularly on situations where there are multiple decision makers, or *agents*.

When compared to computers 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 conservative estimators as compared to applying for example Bayesian statistics in handling uncertainties (Dawes, 1988). As such, human decision making may be substantially improved when assisted by intelligent decision support systems (Druzdzel and Flynn, 2003), providing an important social and economic incentive to develop such systems with intelligent behavior.

The Interactive Collaborative Information Systems (ICIS) project, which has supported the research reported in this thesis, focuses on the development of intelligent techniques and methods that can be applied in decision support systems. It particularly aims to improve the overall quality of information processing and decision making under conditions of stress, risk and uncertainty. For systems to be effective under such circumstances, they need to have the capabilities to:

1. Operate in a highly dynamic environment and to cope with the changes in the environment.

2. Support reasoning with uncertainty, reasoning with risks and reasoning in the absence of knowledge, necessary because of the chaotic nature of the real world. In particular, the system should be able to reach a set of (determined)

goals by influencing the environment in which the system operates.

That is, such a system needs the capability to tackle the problem of *sequential decision making*, making a series of decisions over time. This thesis describes methods to realize such capabilities. In particular, it focuses on decision-theoretic methods to allow for exact or approximate multiagent planning, collaboration and optimization.

Following the ICIS project, the work in this thesis 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 and dynamic environments.

1.1 An Example of a Challenging Environment

An example of a challenging environment is considered in the RoboCup Rescue competition (Kitano, Tadokoro, Noda, Matsubara, Takahashi, Shinjoh, and Shimada, 1999). Motivated by the earthquake in Kobe in 1995 (Tierney and Goltz, 1997), RoboCup Rescue simulates an earthquake in an urban environment. In this scenario, buildings collapse causing roads to get blocked and people to get trapped in the debris. Damage 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 activities allows allocating more agents to deal with the current situation, but may impose severe risks, e.g., a yet unknown fire source may grow out of hand completely.

1.2 Forms of Uncertainty

The example of RoboCup Rescue illustrates how various forms of uncertainty complicate effectively resolving situations. Here we discuss these different types of uncertainty.

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 way the state of the environment changes is stochastic.

State Uncertainty. In the real world an agent might not be able to determine what the state of the environment exactly is. In such cases, we say that the environment is *partially observable.* Partial observability results from noisy and/or limited sensors that may be spatially distributed. Because of *sensor noise* an agent can receive faulty or inaccurate sensor readings, or *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. Due to such sensor limitations, the same sensor reading might require different action choices, a phenomenon referred to as *perceptual aliasing.* In order to mitigate these problems, an agent may use the history of actions it took and the observations it received to get a better estimate of the state of the environment.

Uncertainty with respect to Other Agents. Another complicating factor is the presence of multiple agents that each make decisions that influence the environment. Such an environment together with the multiple agents that operate in it is called a *multiagent system (MAS).* The difficulty in MASs is that each agent can be uncertain regarding the other agents' actions. This is apparent in self-interested and especially adversarial settings, such as games, where agents may not share information or try to mislead other agents. In such settings each agent should try to accurately predict the behavior of the others in order to maximize its payoff.

In the cooperative setting, the agents have the same goal and therefore are willing to coordinate. Still it is non-trivial how such coordination should be performed (Boutilier, 1996). Especially when communication capabilities are limited or absent, the question of how the agents should coordinate their actions is problematic. This problem is magnified in partially observable environments: as the agents are not assumed to observe the complete state of the environment—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 a substitute for it; even if the agents could freely and instantaneously communicate their individual observations, the joint observations would in general not disambiguate the true state of the environment.

1.3 Multiagent Systems

The field of MASs is a broad interdisciplinary field with relations to distributed and concurrent systems, artificial intelligence (AI), economics, logic, philosophy, ecology and social sciences (Wooldridge, 2002). The sub-field of AI that deals with principles and design of MASs is also referred to as 'distributed AI'. Research on MASs is motivated by the fact that it can potentially provide (Vlassis, 2007; Sycara, 1998):

- Speedup and efficiency, due to the asynchronous and parallel computation.
- Robustness and reliability: the whole system can undergo a 'graceful degradation' when one or more agents fail.

- Scalability and flexibility, by adding additional agents as required.
- Lower cost, assuming the agents cost much less than a centralized system.
- Lower development cost and reusability, since it is easier to develop and maintain a modular system.

There are many different aspects of multiagent systems, concerning the type of agents, their capabilities and their environment. For instance, one can consider a homogeneous MAS where all agents are identical, or heterogeneous MAS where the design and capabilities of each agent can be different. The level of cooperation may differ: are the agents cooperative, self-interested or adversarial? The environment can be dynamic or static. These are just a few of many possible parameters one can identify. As such, the total number of settings of these parameters and therefore the number of types of MASs is huge, which has led to many different approaches. Only a subset of these methods are relevant for problems of sequential decision making. The remainder of this section gives a concise overview of such approaches, for a more comprehensive overview of the field of MASs the reader is referred to the texts by Huhns (1987); Singh (1994); Sycara (1998); Weiss (1999); Stone and Veloso (2000); Yokoo (2001); Wooldridge (2002); Xiang (2002); Bordini, Dastani, Dix, and El Fallah Segrouchni (2005); Shoham and Leyton-Brown (2007); Vlassis (2007); Buşoniu, Babuška, and De Schutter (2008).

Many approaches to MASs are based on game theory, which therefore is an essential tool in the field. Game theory studies the interaction of agents formalized in models of games. We also use several game-theoretic models which will be treated in some detail in Section 2.

Within AI much research has focused on planning. The single-agent 'classical planning problem' is to transform the state of the environment, which is typically described in logic, into a goal state through a sequence of actions, the plan. These methods have been extended to the multiagent setting, resulting in a combination of planning and coordination, e.g. distributed problem solving (DPS) (Durfee, 2001). However, like classical planning itself, these extensions, typically fail to address stochastic or partially observable environments (de Weerdt, ter Mors, and Witteveen, 2005).

On the other hand, for the single-agent case, classical planning has been extended to deal with stochasticity and partial observability in *conformant* and *contingent* planners. The latter category is equally expressive as the single-agent decision-theoretic models considered in this thesis (Majercik and Littman, 2003; Poupart, 2005). As such we expect that an extension of those planners to multiple agents may be equally expressive to the models we consider.

Another branch of work is that of teamwork theory, which provides a more implementation-directed perspective to control and coordination of MASs. This line of work is largely based on the theory of *practical reasoning* (Bratman, 1987) as observed with humans. In particular, the *belief-desire-intention (BDI)* model of practical reasoning (Georgeff, Pell, Pollack, Tambe, and Wooldridge, 1999) has inspired many teamwork theories, such as *joint intentions* (Cohen and Levesque, 1990, 1991a,b) and *shared plans* (Grosz and Sidner, 1990; Grosz and Kraus, 1996),

and architectures that implement these theories (Jennings, 1995; Tambe, 1997; Stone and Veloso, 1999; Pynadath and Tambe, 2003). Typically, these BDI-based approaches to teamwork use precompiled plans, which have to be specified by a programmer. Also, in the basis these approaches are based on behavior observed by humans. As such, it is not likely that such a system will complement human performance in decision making under uncertainty. Another disadvantage in this line of work is that there are no clear quantitative measures for their performance, making it hard to judge their quality (Pynadath and Tambe, 2002b; Nair and Tambe, 2005). Therefore, we will not consider this line of work any further. For more information we refer to the texts by Wooldridge and Jennings (1995); Wooldridge (2002); Bordini et al. (2005).

1.4 Decision-Theoretic Planning

Many of the mentioned multiagent approaches 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-theoretic planning (DTP), which compactly specifies sequential decision problems and provides tools to tackle them.

DTP has roots in control theory and in operations research (OR). In control theory, one or more controllers control a deterministic or stochastic system by manipulating the inputs to a system (i.e., selecting actions) to obtain the desired effect on the output of the system. The focus in control theory lies on physical problems with continuous state variables and actions such as the control of a robot arm. In this particular example, the state of the system may be characterized by the angles of the joints and the angle velocities and the desired output may be a particular position of the arm. Operations research, on the other hand, considers tasks related to scheduling, logistics and work flow and tries to optimize the concerning systems. The nature of such problems is often different from the problems in control theory, as they typically involve discrete components. As a result, the OR community has typically considered different models and methods.

Nevertheless there also is an overlap between OR and control theory. Many decision-theoretic planning problems can be formalized as *Markov decision processes* (MDPs) (Puterman, 1994), which have been frequently employed in both control theory as well as operations research. In the last decades, the MDP framework has also gained in popularity in the AI community as a model for planning under uncertainty (Boutilier, Dean, and Hanks, 1999; Guestrin, Koller, Parr, and Venkataraman, 2003). DTP is the intersection of control theory, operations research and AI that studies MDPs and related models.

1.4.1 Planning with Outcome Uncertainty

The MDP is a framework for sequential decision making of a single agent 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).* Also in control theory much research has considered continuous time settings (Sontag, 1998). In order to solve such continuous time settings, however, time is discretized

or special assumptions, such as linear dynamics and quadratic costs, are required (Bertsekas, 2005). As such, most approaches to DTP for multiagent systems have focused on extensions of the MDP, a notable exception is presented by Wiegerinck, van den Broek, and Kappen (2007) and Van den Broek, Wiegerinck, and Kappen (2008), who consider the control of a MAS in continuous space and time.

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. Every time step the state changes stochastically, but the agent chooses an action that selects a particular transition function. Taking an action from a particular state at time step t induces a probability distribution over states at time step $t + 1$. 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, the agent's objective, can be formulated in several ways. The first type of objective of an agent is reaching a specific goal state, for example in a maze in which the agent's goal is to reach the exit. A different formulation is given by associating a certain cost with the execution of a particular action in a particular state, in which case the goal will be to minimize the expected total cost. Alternatively, one can associate rewards with actions performed in a certain state, the goal being to maximize the total reward.

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. Such a planning approach stands in contrast to reinforcement learning (RL) (Sutton and Barto, 1998; Buşoniu et al., 2008), 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 it often is not necessary to explicitly recover the environment model. This thesis focuses only on planning, but considers two factors that complicate computing successful plans: the inability of the agent to observe the state of the environment as well as the presence of multiple agents.

1.4.2 Dealing with State Uncertainty

As mentioned in Section 1.2, noisy and limited sensors may prevent the agent from observing the state of the environment, because the observations are inaccurate and perceptual aliasing may occur. In order to represent such state 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 (Kaelbling, Littman, and Cassandra, 1998; Cassandra, 1998). In a POMDP, an agent no longer knows the state of the world, but rather has to maintain a *belief* over states. That is, it can use the history of observations to estimate the probability of each state and use this information to decide upon an action.

In control theory, the (continuous) observations, also referred to as measurements, are typically described as a function of the state. Sensor noise is modeled by

adding a random disturbance term to this function and is dealt with by introducing a state estimator component, e.g., by Kalman filtering (Kalman, 1960). Perceptual aliasing arises when a state component cannot be measured. For instance it may not be possible to directly measure angular velocity of a robot arm, in this case it may be possible to use a so-called observer to estimate this velocity from its positions over time.

Although the treatment of state uncertainty in control theory involves terminology and techniques different from those in used in POMDPs, the basic idea in both is the same: use information gathered from the history of observations in order to improve decisions. There also is one fundamental difference, however. Control theory typically separates the estimation from the control component. E.g., the estimator returns a particular value for the angles and angle velocities of the robot arm and these values are used to select actions as if there was no uncertainty. In contrast, POMDPs allow the agent to explicitly reason over the belief (the probability of all possible states) and what the best action is given that belief. The up-shot of this is that agents using POMDP techniques can select actions that will provide information about the state.

1.4.3 Multiple Agents

Although POMDPs provide principled treatment of state uncertainty, they only consider a single agent. In order to deal with the effects of uncertainty with respect to other agents, this thesis will consider an extension of the POMDP framework. We also note that this type of uncertainty may be mitigated through communication. Under the stringent assumptions of instantaneous, cost- and noise free communication, they can be discarded altogether, and the problem reduces to a POMDP (Pynadath and Tambe, 2002b). However, in general these assumptions are too strong and deciding *when* to communicate *what* becomes part of the problem. Chapter 3 considers various assumptions with respect to communication delays in MASs. The main focus of this thesis, however is the truly decentralized, non-communicative setting. As it turns out, the framework we consider for these non-communicative MASs can also model communication with a particular cost that is subject to minimization (Pynadath and Tambe, 2002b; Goldman and Zilberstein, 2004) and the non-communicative setting can be interpreted as the special case with infinite cost.

In a MAS each agent can be considered separately. In this case, which we refer to as the *subjective perspective* of a MAS, each such agent maintains an explicit model of the other agents. This is the approach as chosen in the *recursive modeling method (RMM)* (Gmytrasiewicz and Durfee, 1995; Gmytrasiewicz, Noh, and Kellogg, 1998) and the *interactive POMDP (I-POMDP)* framework (Gmytrasiewicz and Doshi, 2005). A difficulty in these approaches, however, is that the other agents also model the considered agent, leading to an infinite recursion of beliefs regarding the behavior of agents. Moreover, the number of possible models of other agents is infinite. Even though there might be solutions to these problems (Rathnasabapathy, Doshi, and Gmytrasiewicz, 2006; Zettlemoyer, Milch, and Kaelbling, 2009), we feel that this approach is more appropriate for systems with self-interested agents.

When planning for a team of cooperative agents, we feel that it is better to adopt an *objective perspective* of the MAS, in which we try to find plans for all agents at the same time. Such a model is the *decentralized POMDP (Dec-POMDP)* introduced by Bernstein, Zilberstein, and Immerman (2000). The Dec-POMDP framework is the generalization to multiple agents of the POMDP and can be used to model a team of cooperative agents that are situated in a stochastic, partially observable environment. Also, it will be the framework adopted in this thesis.

1.4.4 DTP and Game Theory

Up to this point, the overview of decision-theoretic planning has focused on single-agent models (MDPs, POMDPs) and their generalization to multiple agents (the Dec-POMDP). Game theory has not yet been mentioned because it usually is not considered to fall in the DTP category. However, it can definitely be seen as a branch of decision theory. In the author's opinion, game-theoretic models, especially those that consider acting over multiple time steps, should also be considered part of the field of DTP. In Chapter 2 we will treat background for both game-theoretic models and Dec-POMDPs under the term 'decision-theoretic planning for teams of agents'. In contrast to mainstream DTP, game theory has always considered multiple agents, and as a consequence several ideas and concepts from game theory are now being applied in decentralized decision-theoretic planning. In moving from the single-agent case to the multiagent case ideas and concepts from game theory automatically come in to play. Still there have been problems in applying game theory in the design of agents, as Russell and Norvig (2003) mention

> "there have been some barriers that have prevented game theory from being widely used in agent design. [...] there currently is no good way to combine game theoretic and POMDP control strategies. Because of these and other problems, game theory has been used primarily to *analyze* environments that are at equilibrium, rather than to *control* agents within an environment."

This means that there is a gap in the theory between game theory and DTP methods. There has been work to extend game-theoretic models to continuous actions and time, thereby reducing the gap with control theory (Basar and Olsder, 1995). Still the gap with DTP methods considered in AI and OR remains. This thesis makes a contribution to further closing this gap. Specifically, it will apply one-shot game-theoretic models to represent time-steps in a sequential decision problem.

1.5 The Focus of this Thesis

This thesis focuses on decision-theoretic planning for teams of cooperative agents that are situated in a stochastic, partially observable environment. It adopts the *decentralized partially observable Markov decision process (Dec-POMDP)* as its central framework because it is an expressive model that allows for the principled treatment of the uncertainty faced by such teams.

Unfortunately, the power of the Dec-POMDP model comes at a price: Bernstein et al. show that optimally solving a Dec-POMDP is provably intractable (NEXP-complete). As such optimal solutions will only be possible for very small problems and efficient approximation algorithms are an important point of research. However, since any advancements in optimal solution methods are likely to transfer also to approximate ones, this thesis considers both exact and approximate methods. Although recent approximate methods successfully scale with respect to the horizon of the problem (Seuken and Zilberstein, 2007a), optimal algorithms have remained behind. Moreover scalability with respect to other problem parameters such as the number of agents has remained poor also for approximate methods.

> This thesis aims at advancing *decision-theoretic planning* for *cooperative multiagent systems* in *stochastic, partially observable environments* that do *not allow cost- and noise-free instantaneous communication,* as formalized by the *Dec-POMDP* framework. It aims to do so by
>
> 1. extending the theory of Dec-POMDPs and establishing further connections with the area of game theory,
> 2. proposing optimal and approximate solution methods that scale better with respect to the number of agents, and
> 3. proposing optimal solution methods that scale better with respect to the horizon.

The assumption in this thesis is that planning takes place in an off-line phase, after which the plans are executed in an on-line phase. In the decentralized setting, however, this statement deserves some clarification. In the on-line phase the computed plan is executed in a completely decentralized way: each agent knows only the joint policy as found in the planning phase and its individual history of actions and observations. The planning phase, however, can be viewed in two ways. First, we can think of a *centralized* computer that computes the joint plan and consequently distributes these plans to the agents, who then merely execute the plans on-line. In this view the agents are reactive and all the intelligence comes from the centralized computer. The second view is that each agent runs the same planning algorithm in parallel and therefore each agent computes the same joint plan from which it executes its individual component. In this view the planning phase is decentralized too and the agents are decision theoretic. This thesis will not consider decentralization of computation of plans in order the provide an increase in efficiency, i.e., methods that break up the planning problem in sub-problems which are then distributed and processed in parallel. Although such methods are of great interest, the centralized-planning case assumed in this thesis presents enough challenges by itself.

1.6 Applications

Because of the complexity of multiagent decision making under uncertainty, research has been forced to restrict itself to toy problems, as there are no real-life

applications small enough to tackle. Nevertheless, there are some application areas that seem to come within reach. This section gives an overview starting with those application areas and moving to areas that still present more challenges.

It may seem rather obvious that areas that are (almost) within reach of DTP methods are those areas that are somewhat more abstract and thus simpler. An example of such a task is that of load balancing among queues, as presented by Cogill, Rotkowitz, Roy, and Lall (2004). Here, each agent represents a processing unit with a queue and can only observe its own queue size and that of immediate neighbors. The agents have to decide whether to accept new jobs or pass them to another queue. Such a restricted problem can be found in many settings, for instance industrial plants or a cluster of web-servers, and as such DTP (and in particular Dec-POMDP) methods may actually be close to being employed. Another example of an abstract domain is that of communication networks. For instance, Peshkin (2001) treated a packet routing application in which agents are routers and have to minimize the average transfer time of packets. They are connected to immediate neighbors and have to decide at each time step to which neighbor to send each packet. Other approaches to communication networks using decentralized, stochastic, partially observable systems are given by Ooi and Wornell (1996), Tao, Baxter, and Weaver (2001) and Altman (2002). Dec-POMDPs are particularly relevant for communication networks, because the agents (e.g., routers) typically cannot communicate, since the cost of sending meta-level communication occupying the communication channels is prohibitively large. Even though such communication networks typically are too complex to fully model with DTP methods, policies found by such methods on smaller, more abstract instances, may be extrapolated by human experts to improve current policies. Also it may be possible to use machine-learning techniques to aid such extrapolations. In particular, the field of *inductive transfer* or *transfer learning* deals with such problems. In Chapter 5 we will also introduce a technique, *transfer planning*, that uses the solution of a smaller Dec-POMDP problem to aid in the solution of a larger one.

Much research in AI currently focuses on single-agent POMDP methods for active perception (Spaan, 2008; Grappiolo, Whiteson, Pavlin, and Bakker, 2009), which roughly can be explained as the problem where you need to take particular actions (active) to gain more information (perception). When scaling such scenarios to multiple agents, we immediately end up in Dec-POMDP-like settings. An application domain that receives much attention in the Dec-POMDP community is that of sensor networks (typically used for surveillance) (Lesser, Ortiz Jr., and Tambe, 2003). Because there typically is much structure and relatively little interaction, researchers have been able to scale to larger demonstration networks (Nair, Varakantham, Tambe, and Yokoo, 2005; Varakantham, Marecki, Yabu, Tambe, and Yokoo, 2007; Kumar and Zilberstein, 2009).

In the long term, Dec-POMDPs and related models may be important for all (cooperative) robotics (Arai, Pagello, and Parker, 2002). Robots have to navigate in the real world which is always partially observable due to sensor noise and perceptual aliasing. Also, in most of these domains full communication is either not possible (e.g., too little bandwidth to transmit video streams from many cameras) or consumes resources (e.g., battery power) and thus has a particular cost.

Therefore models such as Dec-POMDPs, which do consider partially observable environments are relevant for essentially all teams of embodied agents. However, the size and continuous nature of this physical world also presents great challenges for DTP methods and therefore large scale deployment of such methods in this area may be far away. Still, there are examples of research that has been able to find small more abstract problems within this area and apply DTP methods there. For instance, Becker, Zilberstein, Lesser, and Goldman (2004b) consider multi-robot space exploration, where the agents are Mars rovers and the decision problem is how to proceed their mission: whether to collect particular samples at specific sites or not. The rewards of particular samples can be sub- or super-additive, making this task non-trivial. Another example is given by Emery-Montemerlo (2005), who considered multi-robot navigation in which a team of agents with noisy sensors has to act to find/capture a goal. They actually managed to run their Dec-POMDP algorithm on robots. Also within robotic soccer as applied in RoboCup (Kitano, Asada, Kuniyoshi, Noda, and Osawa, 1997), some researchers have been able to apply decision-theoretic methods (Kok and Vlassis, 2005).

The application area of decision support systems for complex real-world settings, such as crisis management, is actually closely related to the field of cooperative robotics described above. Also in this setting, it is typically necessary to deal with the real world which often is continuous and highly uncertain. As such, the widespread application of DTP methods in this area may also require quite a few more years of research. Nevertheless, also in this field there have been some advances. For instance, within the setting of RoboCup Rescue (Kitano et al., 1999) as described in Section 1.1, researchers have been able to model small subproblems using Dec-POMDPs (Nair, Tambe, and Marsella, 2002, 2003a,b; Oliehoek and Visser, 2006; Paquet, Tobin, and Chaib-draa, 2005).

1.7 Organization of Thesis and Publications

This thesis is based on several publications. This section explains the organization of the thesis, at the same time explaining on which publications the different parts are based. Chapter 2, gives an extensive introduction to decision-theoretic planning methods for MASs. In the first part, the necessary game-theoretic background is provided, covering both one-shot and sequential decision models. The second part gives a formal introduction to the Dec-POMDP model and solution methods.

Chapter 3 describes optimal *value functions* for Dec-POMDPs and is largely based on the work by Oliehoek, Spaan, and Vlassis (2008b) to which we refer as OSV08 here. For single-agent MDPs, many solution methods are based on value functions that indicate what the expected total reward is when starting to act from a particular state of the environment. For Dec-POMDPs, however, such value functions had previously not been investigated. Our investigation of such value functions is facilitated by interpreting the Dec-POMDP as a series of one-shot Bayesian games (BGs) of which the payoff function corresponds to a so-called action-value or *Q-value function*. The chapter provides optimal value functions for Dec-POMDP with different assumptions with respect to communication. First,

it considers the setting without communication, then it considers zero, one and k-steps delayed communication. The chapter expands on OSV08 in giving a more cohesive overview of the value functions of these settings and how they relate.

Next, Chapter 4, which is also based on OSV08, introduces a generalized value-based policy search framework for Dec-POMDPs that works by iteratively solving BGs that represent different stages. Because the optimal value-function identified in Chapter 3 is typically infeasible to compute, approximate Q_{MDP}, Q_{POMDP} and Q_{BG} value functions are introduced that correspond to appropriately chosen delays as discussed in Chapter 3. Of these, Q_{MDP} and Q_{POMDP} are well-known; Q_{BG} was proposed by Oliehoek and Vlassis (2007b). Next, the chapter introduces the generalized MAA* (GMAA*) policy search method and explains how it generalizes the existing methods of Szer and Charpillet (2005) and Emery-Montemerlo, Gordon, Schneider, and Thrun (2004). GMAA* was first proposed by Oliehoek and Vlassis (2007a).

Chapter 5 exploits structure present in many real-world domains to scale up with respect to the number of agents. Previous research achieved this by imposing assumptions of complete independence between agents on the Dec-POMDP model. However, such assumptions are too restrictive for many interesting problems. Still, in many such problems the agents may be *nearly* independent: for instance each agent may only need to interact a few neighbors. In particular, Chapter 5 introduces the framework of *factored* Dec-POMDPs with *additive rewards* and proposes to model such factored Dec-POMDPs as series of *collaborative graphical Bayesian games (CGBGs)*. Such CGBGs have a compact representation and we discuss how they can be computed efficiently through approximate inference. The chapter also introduces *factored* Q-value functions that can be used as the payoff functions for the CGBGs. The CGBGs can be optimally solved by non-serial dynamic programming or approximated efficiently by introducing a factor-graph description and running a message passing algorithm on this graph. Given the complete specification of the CGBGs and a method to solve them, it is possible to apply GMAA* policy search methods. An empirical evaluation reveals that the proposed methods scale to problems with up to 1,000 agents, as opposed to the maximum of 5 agents in previous literature. This chapter is based on Oliehoek, Spaan, Whiteson, and Vlassis (2008c) and an article under submission (Oliehoek, Whiteson, and Spaan, 2010).

The last chapter that describes research is Chapter 6. This chapter considers techniques to provide scaling with respect to the horizon of the decision problem. It is based on Oliehoek, Whiteson, and Spaan (2009) who propose to perform lossless clustering of the histories of agents (i.e., types) within the BGs. Such clustering of types was also proposed by Emery-Montemerlo, Gordon, Schneider, and Thrun (2005), but only in an approximate setting. This chapter, on the other hand, establishes criteria under which the optimal value function is guaranteed to be equal and therefore histories can be clustered without loss in value in these settings. By applying this technique within GMAA*, we have been able to compute optimal solutions for significantly longer horizons than previously possible, although the amount of scale-up is problem dependent.

Finally, Chapter 7 concludes and discusses the presented research and the most

important directions of future work.

1.7.1 Other Research

Not all research that was performed fits in the scope of this thesis. This section describes the work that did not make it into the thesis. To start, apart from the identification of the optimal value functions under different communication settings, most other work on communication in Dec-POMDPs has not been included. This line of work includes:

- Certain theoretical properties of the Q_{BG} value function, especially the proof that such value functions are piecewise-linear and convex over the space of (joint) beliefs and that for an infinite horizon they can be approximated with arbitrary accuracy (Oliehoek, Vlassis, and Spaan, 2007c).
- Based on these properties, Oliehoek, Spaan, and Vlassis (2007b) propose finite and infinite-horizon value iteration algorithms for the computation of value functions of one-step delayed communication Dec-POMDPs and propose efficient approximations by extending Perseus, an approximate POMDP solver (Spaan and Vlassis, 2005).
- Exact value functions and their efficient computation for settings with a stochastic delay of 0 or 1 time step (Spaan, Oliehoek, and Vlassis, 2008). The work also applies such value functions as an approximation of settings with longer stochastic delays.

Also, we developed a policy search method based on cross-entropy optimization (Oliehoek, Kooij, and Vlassis, 2007a; Oliehoek, Kooi, and Vlassis, 2008a). The resulting method DICE was the first method to be demonstrated on regular Dec-POMDPs with more than 2 agents. Although this line of research addresses the plain Dec-POMDP setting, it does not fit nicely in the framework of value-based planning described in this thesis. The software toolbox co-developed with Matthijs Spaan (Spaan and Oliehoek, 2008) is not described in this thesis and the explicit link between Dec-POMDPs and extensive-form games was described in a technical report (Oliehoek and Vlassis, 2006), but has not been reproduced. Finally, older work resulting from the author's master's thesis was also omitted. This research focused on applying POMDPs in competitive two-player (poker) games (Oliehoek, Spaan, and Vlassis, 2005a) and on coevolutionary methods to efficiently compute approximate Nash equilibria for such games (Oliehoek, Vlassis, and de Jong, 2005b; Oliehoek, de Jong, and Vlassis, 2006).

Chapter 2

Decision-Theoretic Planning for Teams of Agents

The introduction explained that this thesis focuses on decision-theoretic planning (DTP) for teams of cooperative agents, a problem of sequential decision making. The goal is to find a plan that specifies what decisions the agents should take over a predetermined number of time steps. Such teams of cooperative agents are a special case of a multiagent system (MAS). Therefore, this chapter provides background of DTP methods for multiagent systems. First, however, we take a step back and consider making a one-shot decision for a single agent.

Decision theory describes how a decision maker, or agent, should make a decision given its preferences over as set of alternatives. Let us for the moment assume that an agent has preferences over some abstract set of *outcomes* $\omega \in \Omega$. If the agent's preferences satisfy the axioms of utility theory, they can be conveniently represented by a utility function u that maps each outcome to a real number $u(\omega)$ (Russell and Norvig, 2003; DeGroot, 2004). This number indicates how the agent values that outcome: if A is preferred to B, than $u(A) > u(B)$.

Now consider that the agent has to make a one-shot decision that results in a particular outcome. In particular, the agent can select an action from some specified set $a \in \mathcal{A}$. If the agent has a model of how its actions influence the outcomes, i.e. when the probabilities $\Pr(\omega|a)$ are available, the agent can compute the expected utility of each action, which is defined as

$$u(a) \equiv E\left[u(\omega)|a\right] = \sum_{\omega \in \Omega} u(\omega) \Pr(\omega|a). \tag{2.0.1}$$

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

This chapter extends the preceding explanation of decision making and introduces some different models in which rational agents make decisions by maximizing their expected utility. In particular, we are interested in rational agents in multiagent systems. In a MAS, the outcomes may be influenced by multiple agents and therefore the utility function for a particular agent will in general depend on

the actions of multiple agents. As such, making decisions in MASs becomes significantly more complex. Section 2.1 first introduces game-theoretic models for such settings and subsequently treats models that extend multiagent decision making to multiple time steps. Section 2.2 then introduces the Dec-POMDP, the model central in this thesis, the remainder of the chapter provides background on the Dec-POMDP and existing solution methods.

We note that although the focus of this thesis is on sequential decision making formalized as Dec-POMDPs, the game-theoretic background is essential. In particular, we will model Dec-POMDPs by series of one-shot decision problems, Bayesian games, from game theory.

2.1 Game-Theoretic Models

This section describes game-theoretic models for multiagent decision making. The main difference between these models and the Dec-POMDP model is that game-theoretic models concentrate on the interaction between agents, referred to as *players,* and not so much on the environment. This is is especially true for one-shot games as will be treated in Subsection 2.1.1. Sequential decision making is discussed in Subsection 2.1.2, which introduces extensive games. These games can model a dynamic environment, although also here the main focus is on the interaction. More recently, methods have been proposed to exploit structure within games. These methods will receive more attention in Chapter 5. Here, we just briefly mention graphical games that exploit structure in the interactions between agents in Subsection 2.1.1.3 and models that exploit structure of the environment in Subsection 2.1.2.3.

2.1.1 One-Shot Decisions

One-shot games are games that are played only once. That is, there is one and only one interaction between the agents, that is formalized by the game. This subsection introduces two types of one-shot games, strategic and Bayesian games, and the (Bayesian) Nash equilibrium solution concept.

2.1.1.1 Strategic Games and Nash Equilibria

The simplest form of game is the *strategic* or *normal-form game*. A strategic game consists of a set of agents or players, each of which has a set of actions (or strategies). The combination of selected actions specifies a particular *outcome*: a payoff profile specifying the payoff for each agent. When a strategic game consists of two agents, it can be visualized as a matrix as shown in Figure 2.1. The first game shown is called 'Chicken' and involves two teenagers who are driving head on. Both have the option to drive on or chicken out. Each teenager's payoff is maximal (+2) when he drives on and his opponent chickens out. However, if both drive on, a collision follows giving both a payoff of −1. The second game is the meeting location problem. Both agents want to meet in location A or B. They have no preference over which location, as long as both pick the same location. This game

	D	C
D	$-1,-1$	$+2,0$
C	$0,+2$	$+1,+1$

(a) Chicken. Both players have the option to (D)rive on or (C)hicken out.

	A	B
A	$+2$	0
B	0	$+2$

(b) The meeting location problem has identical payoffs, so each entry contains just one number.

Figure 2.1: Two strategic games: Chicken and the meeting location problem.

is fully cooperative, which is modeled by the fact that the agents receive identical payoffs.

Definition 2.1 (Strategic game)**.** Formally, a *strategic game* is a tuple $\langle \mathcal{D}, \mathcal{A}, u \rangle$, where

- $\mathcal{D} = \{1, \ldots, n\}$ is the set of n agents (decision makers),
- $\mathcal{A} = \times_{i \in \mathcal{D}} \mathcal{A}_i$ is the set of joint actions $\mathbf{a} = \langle a_1, \ldots, a_n \rangle$, and
- $u = \langle u_1, \ldots, u_n \rangle$ with $u_i : \mathcal{A} \to \mathbb{R}$ is the payoff function of agent i.

When the set of actions $\mathcal{A}_i$ for each agent i is finite, the strategic game is called *finite.*

Game theory takes an objective perspective and tries to specify for each agent how to play. That is, a game-theoretic solution should suggest a policy for each agent. In a strategic game we write π_i to denote a policy for agent i and π for a joint policy. A policy for agent i is simply one of its actions $\pi_i = a_i \in \mathcal{A}_i$ (i.e., a *pure* policy), or a probability distribution over its actions $\pi_i \in \mathcal{P}(\mathcal{A}_i)$ (i.e., a *mixed* policy).[1] Also, the policy suggested to each agent should be rational given the policies suggested to the other agent; it would be undesirable to suggest a particular policy to an agent, if it can get a better payoff by switching to another policy. Rather, the suggested policies should form an equilibrium, meaning that it is not profitable for an agent to unilaterally deviate from its suggested policy. This notion is formalized by the concept of Nash equilibrium.

Definition 2.2 (Nash equilibrium)**.** A pure policy profile $\pi = \langle \pi_1, \ldots, \pi_i, \ldots, \pi_n \rangle$ specifying a pure policy for each agent is a *Nash Equilibrium (NE)* if and only if

$$u_i(\langle \pi_1, \ldots, \pi_i, \ldots, \pi_n \rangle) \geq u_i(\langle \pi_1, \ldots, \pi_i', \ldots, \pi_n \rangle), \quad \forall_{i:1 \leq i \leq n}, \forall_{\pi_i' \in \mathcal{A}_i}. \tag{2.1.1}$$

This definition can be easily extended to incorporate mixed policies by defining

$$u_i(\langle \pi_1, \ldots, \pi_n \rangle) \equiv \sum_{\langle a_1, \ldots, a_n \rangle} u_i(\langle a_1, \ldots, a_n \rangle) \prod_{j=1}^{n} \Pr(a_j | \pi_j). \tag{2.1.2}$$

Nash (1950) proved that when allowing mixed policies, every (finite) strategic game contains at least one NE, making it a proper solution for a game. However, it is

[1] We write $\mathcal{P}(\cdot)$ to denote the set of probability distributions over $(\cdot)$.

unclear how such an NE should be found. In particular, there may be multiple NEs in a game, making it unclear which one to select. In order to make some discrimination between Nash equilibria, we can consider NEs such that there is no other NE that is better for everyone. This intuition is formalized by the concept of Pareto optimality.

Definition 2.3 (Pareto optimality)**.** A joint action $\mathbf{a}$ is *Pareto optimal* if there is no other joint action $\mathbf{a}'$ that specifies at least the same payoff for every agent and a higher payoff for at least one agent. I.e., there exists no $\mathbf{a}'$ such that

$$\forall_i \; u_i(\mathbf{a}') \geq u_i(\mathbf{a}) \quad \wedge \quad \exists_i \; u_i(\mathbf{a}') > u_i(\mathbf{a}). \tag{2.1.3}$$

If there does exist an $\mathbf{a}'$ such that (2.1.3) holds, then $\mathbf{a}$ is said to be *Pareto dominated* by $\mathbf{a}'$.

This means that if a joint action $\mathbf{a}$ is Pareto optimal, a different joint action $\mathbf{a}'$ that is better for one agent, must be worse for another agent. For instance, in the Chicken problem of Figure 2.1a $\langle$D,D$\rangle$ is not Pareto optimal, because it is Pareto dominated by all other joint actions. The other joint actions, however, are all Pareto optimal.

We now extend this definition to (possibly mixed) Nash equilibria.

Definition 2.4 (Pareto optimal NE)**.** A Nash Equilibrium π is referred to as *Pareto Optimal (PO)* if and only if there is no other π' such that π' is a NE *and* π' Pareto dominates π.

In the case when multiple Pareto optimal Nash equilibria exist, the agents can agree beforehand on a particular ordering, to ensure the same NE is chosen.

2.1.1.2 Bayesian Games

A strategic game of imperfect information or *Bayesian game* (Osborne and Rubinstein, 1994) is an augmented strategic game in which the players hold some private information. This private information defines the *type* of the agent, i.e., a particular type $\theta_i \in \Theta_i$ of an agent i corresponds to that agent knowing some particular information. The payoff the agents receive now no longer only depends on their actions, but also on their private information. Formally, a BG is defined as follows:

Definition 2.5. A *Bayesian game (BG)* is a tuple $\langle \mathcal{D}, \mathcal{A}, \mathbf{\Theta}, \Pr(\mathbf{\Theta}), \langle u_1, \ldots u_n \rangle \rangle$, where

- $\mathcal{D}$ is the set of n agents,
- $\mathcal{A}$ is the set of joint actions $\mathbf{a} = \langle a_1, \ldots, a_n \rangle$,
- $\mathbf{\Theta} = \times_i \Theta_i$ is the set of joint types $\boldsymbol{\theta} = \langle \theta_1, \ldots, \theta_n \rangle$,
- over which a probability function $\Pr(\mathbf{\Theta})$ is specified, and
- $u_i : \mathbf{\Theta} \times \mathcal{A} \to \mathbb{R}$ is the payoff function of agent i.

In a normal form game the agents select an action. Now, in a BG the agents can condition their action on their private information. This means that in BGs the agents use a different type of policies. For a BG, we denote a joint policy $\boldsymbol{\beta} = \langle \beta_1,...,\beta_n \rangle$, where β_i is the individual policy of agent i. Deterministic (pure) individual policies are mappings from types to actions $\beta_i : \Theta_i \rightarrow \mathcal{A}_i$, while randomized policies map each type θ_i to a probability distribution over actions $\mathcal{P}(\mathcal{A}_i)$. In the context of BGs, an NE is usually referred to as a *Bayesian Nash equilibrium (BNE),* but the definition is similar.

It is possible to convert a BG G to a strategic game G'. The actions a'_i of an agent i in G' correspond to all the pure BG-policies β_i it has in G. The payoff $u'_i(\mathbf{a}')$ for agent i of joint action $\mathbf{a}'$, which corresponds to a joint BG policy $\boldsymbol{\beta}$ in G, is given by taking the expectation over joint types

$$u'_i(\mathbf{a}') \equiv E_{\boldsymbol{\theta}}[u_i(\boldsymbol{\theta}, \langle \beta_1(\theta_1),...,\beta_n(\theta_n) \rangle)].$$

As such, the result of Nash (1950) transfers to (finite) BGs: there is guaranteed to be a BNE in mixed policies.

In the special case of identical payoffs for the agents, a solution of a BG exists in pure policies. It is given by the following theorem:

Theorem 2.1. *For a BG with identical payoffs* $\forall_{i,j} \forall_{\boldsymbol{\theta}} \forall_{\mathbf{a}} \; u_i(\boldsymbol{\theta},\mathbf{a}) = u_j(\boldsymbol{\theta},\mathbf{a})$ *and we write simply* $u(\boldsymbol{\theta},\mathbf{a})$*. Its solution is given by:*

$$\boldsymbol{\beta}^* = \arg\max_{\boldsymbol{\beta}} \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \Pr(\boldsymbol{\theta}) u(\boldsymbol{\theta}, \boldsymbol{\beta}(\boldsymbol{\theta})), \tag{2.1.4}$$

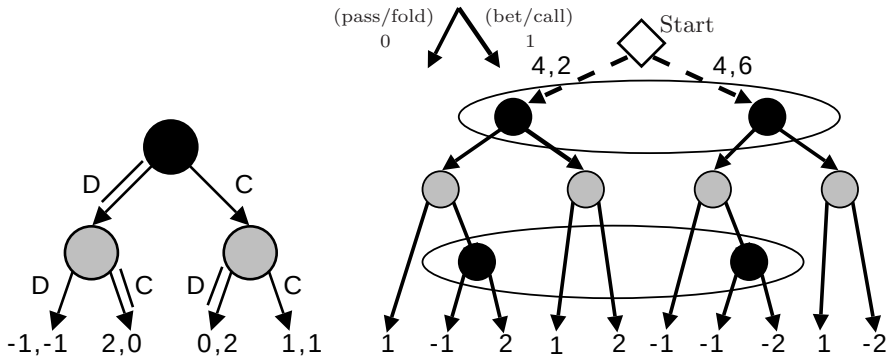
where $\boldsymbol{\beta}(\boldsymbol{\theta}) = \langle \beta_1(\theta_1),...,\beta_n(\theta_n) \rangle$ *is the joint action specified by* $\boldsymbol{\beta}$ *for joint type* $\boldsymbol{\theta}$*. This solution constitutes a Pareto optimal Bayes-Nash equilibrium.*

Sketch of Proof. The proof consists of two parts: the first shows that $\boldsymbol{\beta}^*$ is a Nash equilibrium by rewriting (2.1.4) from the perspective of an arbitrary agent i, the second shows it is Pareto optimal by contradiction. The full proof is listed in Appendix D. □

2.1.1.3 Graphical Models for Games

Strategic and Bayesian games can be represented by multi-dimensional matrices, where the number of dimensions is equal to the number of agents. This means that the size of these games grows exponentially in the number of agents. If the interaction is dense, i.e. each agent affects the payoff of all the other agents, there is no way around this. However, in many real-world domains agents have only limited interaction with other agents. E.g., an agent may need to interact only with the subset of agents that is in its physical neighborhood.

Graph-based models, or *graphical models*, have received much attention in AI because of their ability to compactly represent domains that exhibit some structure (Pearl, 1988; Kschischang, Frey, and Loeliger, 2001; Murphy, 2002; Bishop, 2006, ch.8). In a similar way, *graphical games (GGs)* (Kearns, Littman, and Singh, 2001; Kearns, 2007) and *graphical Bayesian games (GBGs)* (Singh, Soni, and Wellman, 2004b; Soni, Singh, and Wellman, 2007) compactly represent games with multiple agents. We will return to the topic of graphical models and games in Chapter 5.



(a) A sequential version of 'Chicken' where agent 1 gets to decide first. The rational decisions are indicated by double lines.

(b) A partial game-tree of 8-card poker. Shown are situations in which (4,2) and (4,6) are dealt. Gambler's decision nodes are black, dealer's are gray. The diamond represent the chance move at start. The payoffs are given for the gambler.

Figure 2.2: Two extensive games.

2.1.2 Sequential Decisions

The game-theoretic models reviewed so far consider one-shot interactions. The same models can be used when such a one-shot interaction takes place repeatedly. This is the problem as considered in *repeated games* (Osborne and Rubinstein, 1994). Although repeated games are an instance of sequential decision making, the game is exactly the same each time. As such it is not possible to model any temporal context such as a dynamic environment within this paradigm. In the rest of this section we will consider *extensive games,* which do allow different interactions over time.

2.1.2.1 Extensive Games

Extensive games (Kuhn, 1950, 1953) are specified by a tree, referred to as a *game tree.* In this tree, there are two types of nodes: decision nodes and outcome nodes. Decision nodes are decision points each associated with one of the players, which means that the decisions are made sequentially.[1] An action of an agent corresponds to selecting a child node in the tree. Outcome nodes are the leaves of the tree and specify a payoff for each of the agents. Figure 2.2a shows a sequential version of 'Chicken'. In this version, agent 1 may decide first and this choice is observed by agent 2.

In an extensive game, the rational choice at each decision node can be determined from the leaves to the root. This procedure first described by Zermelo (1913) is usually referred to as *backwards induction.* Figure 2.2a indicates the resulting

[1] It is possible to model simultaneous decisions by hiding the decision of the first player in an extensive game of imperfect information.

policies, which is a Nash equilibrium, by double lines: agent 1 selects (D)rive on, while agent 2 (C)hickens out. Note that even in parts of the trees that are not reached by the NE rational actions are specified: i.e., would agent 1 select C, then the policy specifies that agent 2 selects D. An NE that possesses this property is called a *sub-game perfect Nash equilibrium* (Osborne and Rubinstein, 1994).

2.1.2.2 Extensive Games with Imperfect Information

Extensive games have been extended to deal with partial information and chance. This is explained using a small game from literature (Koller and Pfeffer, 1997) called 8-card poker.

Example 2.1 (8-Card poker). 8-Card poker is played by two agents: a dealer and a gambler, who both own two coins. Before the game starts, each agent puts one coin in the pot, the *ante*. Then both agents are dealt one card out of a deck of eight cards (1 suit, ranks 1–8). After they have observed their card, the agents are allowed to bet their remaining coin, starting with the gambler. If the gambler bets his coin, the dealer has the option to also bet ('call' in poker terms) or pass ('fold'). If the dealer folds he loses the ante, and if he calls showdown follows. If the gambler does not bet, the dealer can choose to bet his coin. If the dealer does so, the gambler will have to decide whether to fold or call. If the game reaches the showdown (neither player bets or the bet is called), the player with the highest card wins the pot.

The game of 8-card poker can be modeled as an extensive game with partial (imperfect) information (Kuhn, 1950, 1953). To model chance, decision nodes are split in two categories: decision nodes that represent points at which agents can make a move, and chance nodes which represent stochastic transitions 'taken by nature'. In 8-card poker, the only chance node is the starting state, in which two cards are chosen at random from the 8-card deck and are dealt to the agents. In a partial information game, an agent may be uncertain about the true state of the game. In particular, an 8-card poker agent may not be able to discriminate between some nodes in the tree. The nodes that an agent cannot tell apart are grouped in *information sets*.

In Figure 2.2 a part of the game-tree of 8-card poker is drawn. At the root of the tree, the chance node 'start', a card is dealt to each agent. At each decision node the agents can choose between action 1 (*bet/call*), and action 0 (*pass/fold*). The figure shows two deals: in the first the dealer receives card 2, in the second he receives card 6. The gambler receives card 4 in both cases. Therefore the gambler cannot discriminate between the two deals. This is illustrated by the information sets indicated by ovals. The leaves of the tree represent the outcomes of the game and the corresponding payoffs. In the figure only the payoff of the gambler is shown, the payoff of the dealer is exactly the opposite, as 8-card poker is a zero-sum game.

An assumption that usually is made with the analysis of extensive form games is that of *perfect recall*. It embodies that, at a certain node or phase in the game, a player perfectly remembers the actions he took and observations he received.

2.1.2.3 Other Models of Sequential Decision Making

More recently, *influence diagrams (ID)* have been proposed for decision making (Howard and Matheson, 1984/2005). An influence diagram models a decision making situation by a graphical model that discriminates between variables (chosen by nature) and decision nodes. Additionally the model specifies one or more utility nodes, that specify the payoff for the decision maker. Koller and Milch (2003) extended IDs to include multiple agents. Their model, the *multiagent influence diagram (MAID),* can represent the same type of interactions as strategic and extensive games, but the representation can be more compact because the variables are made explicit (whereas an extensive game typically has to combine all information in its nodes). Koller and Milch show how Nash-equilibria can be computed for such MAIDs. *Networks of influence diagrams (NIDs)* were introduced by Gal and Pfeffer (2008) and extent MAIDs by allowing the agents to be uncertain over the game being played and about the models of each other. Each such model is a MAID and they are connected through a graph.

2.1.3 The Shortcoming of Game-Theoretic Models

In the preceding section we have introduced a number of models for decision making under uncertainty. However, these models all share one important drawback: they are unable to compactly specify decision problems over multiple time steps. For instance, although extensive games can model problems of sequential decision making in uncertain environments, the game trees needed to model complex environments are extremely large. MAIDs and NIDs 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. Doshi, Zeng, and Chen (2008) propose a closely related graphical model formulation of I-POMDPs called *interactive dynamic influence diagram (I-DID)*. Even though this model allows for a compact representation of sequential decision problems, like the I-POMDP, it gives a subjective view from a particular agent and as such may be more suitable for self-interested agents. Moreover I-DIDs lead to the same difficulties as I-POMDPs in that there is an infinite hierarchy of beliefs about each other.

2.2 Decentralized POMDPs

This thesis adopts the decentralized POMDP framework as its model for objective sequential decision making for a team of cooperative agents because it can be specified compactly. In the remainder of this chapter we formally define the Dec-POMDP and some other models from the family of discrete time, MDP-derived frameworks and discuss some of their properties. Intuitively, such models specify a number of agents that inhabit a particular environment, which is considered at discrete *time steps*, also referred to as *stages* (Boutilier et al., 1999) or *(decision) epochs* (Puterman, 1994). The number of time steps the agents will interact with

their environment is called the *horizon* of the decision problem, and will be denoted by h.

The family of MDP-derived frameworks considered in decision theoretic planning very neatly fits the definition of an agent (Russell and Norvig, 2003) by offering an interface of actions and observations to interact with the environment. At each stage $t = 0,1,2,\ldots,h-1$ every agent under consideration takes an action and the combination of these actions influences the environment, causing a state transition. At the next time step, each agent first receives an observation of the environment, after which it has to take an action again. The transition- and observation probabilities are specified by the transition- and observation model and determine the dynamics of the environment. Additionally there are rewards that specify what behavior is desirable. Hence, the reward model defines the agents' goal or task: the agents have to come up with a plan that maximizes the expected long term reward signal.

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

Definition 2.6 (Dec-POMDP)**.** A *decentralized partially observable Markov decision process (Dec-POMDP)* is defined as a tuple $\langle \mathcal{D},\mathcal{S},\mathcal{A},T,R,\mathcal{O},O,h,\boldsymbol{b}^0 \rangle$, where

- $\mathcal{D} = \{1,\ldots,n\}$ is the set of n agents.
- $\mathcal{S}$ is a (finite) set of states.
- $\mathcal{A}$ is the set of joint actions.
- T is the transition probability function.
- R is the immediate reward function.
- $\mathcal{O}$ is the set of joint observations.
- O is the observation probability function.
- h is the horizon of the problem as mentioned above.
- $\boldsymbol{b}^0 \in \mathcal{P}(\mathcal{S})$, is the initial state distribution at time $t = 0$.

The Dec-POMDP model extends single-agent (PO)MDP models by considering *joint* actions and observations. In particular, we define $\mathcal{A} = \times_i \mathcal{A}_i$ as the set of *joint actions*. Here, $\mathcal{A}_i$ is the set of actions available to agent i which can be different for each agent. Every time step, one joint action $\mathbf{a} = \langle a_1,\ldots,a_n \rangle$ is taken. How this joint action influences the environment is described by the transition function T as will be detailed in Subsection 2.2.1. In a Dec-POMDP, agents only know their own individual action; they do not observe each other's actions. We will assume that any action $a_i \in \mathcal{A}_i$ can be selected at any time. So the set $\mathcal{A}_i$ does not depend on the stage or state of the environment. In general, we will denote the stage using superscripts, so $\mathbf{a}^t$ denotes the joint action taken at stage t, a_i^t is the individual action of agent i taken at stage t. Also, we write $\mathbf{a}_{\neq i} = \langle a_1,\ldots,a_{i-1},a_{i+1},\ldots,a_n \rangle$ for a profile of actions for all agents but i.

Similarly to the set of joint actions, $\mathcal{O} = \times_i \mathcal{O}_i$ is the set of joint observations, where $\mathcal{O}_i$ 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 by Figure 2.3. The observation function O specifies the probabilities of joint observations as will be explained in Subsection 2.2.2. Notation with respect to time and indices for observations is analogous to the notation for actions.

The work in this thesis is based on the assumption that the state action and observation sets are finite. Infinite action- and observation sets are very difficult to deal with even in the single-agent case, and to the author's knowledge no research has been performed on this topic in the partially observable, multiagent case.

A final note is that the Dec-POMDP model is equivalent to the *multiagent team decision problem (MTDP)* that was introduced by Pynadath and Tambe (2002b). Formally, the MTDP defines an extension to a richer belief space for each agent. Would there be a compact belief representation for individual agents in partially observable MASs, this could be used as the basis of the policy in the MTDP model. So far no such compact representation has been proposed for the general setting, and the models are equivalent.

2.2.1 States and Transitions

Actions and observations are the interface between the agents and their environment. The Dec-POMDP framework describes this environment by its *states* and *transitions.* Rather than considering a complex, typically domain-dependent model of the environment that explains *how* this environment works, a descriptive stance is taken: A Dec-POMDP specifies an environment model simply as the set of possible states $\mathcal{S} = \left\{ s_1,...,s_{|\mathcal{S}|} \right\}$ of the environment, together with the probabilities of state transitions, which are dependent on executed joint actions. In particular, the transition to a state 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 1.2.

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:

$$\Pr(s^{t+1}|s^t,\mathbf{a}^t,s^{t-1},\mathbf{a}^{t-1},...,s^0,\mathbf{a}^0) = \Pr(s^{t+1}|s^t,\mathbf{a}^t). \tag{2.2.1}$$

Also, we will assume that the transition probabilities are *stationary*, meaning that they are independent of the stage t. In such a case the transition model T can be described by $|\mathcal{A}|$ matrices of size $|\mathcal{S}| \times |\mathcal{S}|$. In Chapter 5 we will consider *factored states*, which make the variables, or *factors,* that describe each state explicit and allow for a more compact representation of the transition model.

2.2.2 The Observation Model

In a way similar to how the transition model T describes the stochastic influence of actions on the environment, the observation model O describes how the state of the

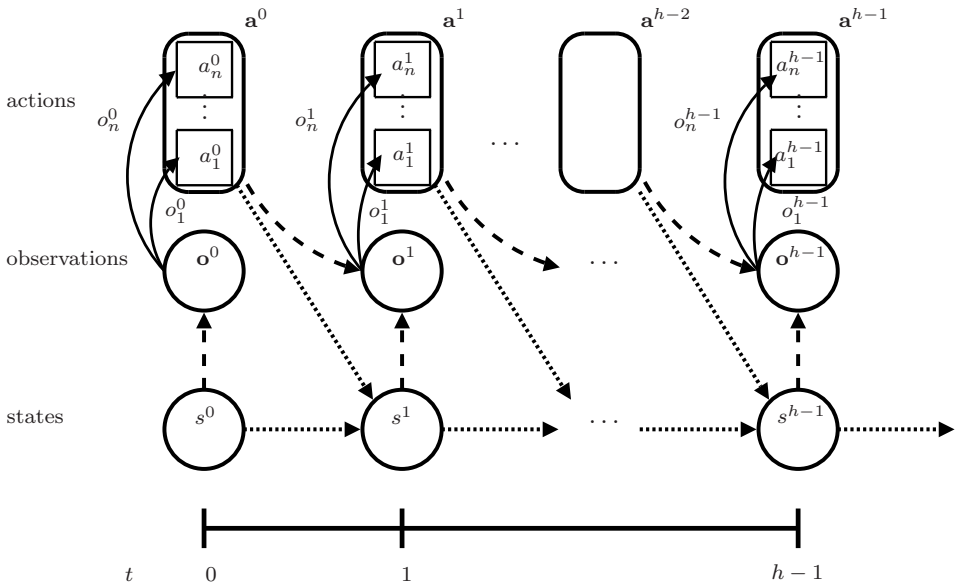


Figure 2.3: An illustration of the dynamics of a Dec-POMDP. At every stage the environment is in a particular state. This state emits a joint observation, of which each agent observes its individual observation. Then each agent selects an action forming the joint action.

environment is perceived by the agents. Formally, O is the observation function, a mapping from joint actions and resulting states to probability distributions over joint observations: $O : \mathcal{A} \times \mathcal{S} \to \mathcal{P}(\mathcal{O})$. I.e., it specifies

$$\Pr(\mathbf{o}^t | \mathbf{a}^{t-1}, s^t). \tag{2.2.2}$$

This implies that the observation model also satisfies the Markov property (there is no dependence on the history). Also the observation model is assumed stationary: there is no dependence on the stage t.

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 model settings which concern communication. For instance, if one agent has an action "write 'A' on blackboard" and the other agent has an observation "see 'A' written on blackboard", the agents certainly do have a mechanism of communication through the state of the environment. However, rather than making this communication explicit, we say that the Dec-POMDP can model communication implicitly through the actions, states and observations. Subsection 2.8.3 further elaborates on communication in Dec-POMDPs.

2.2.3 Rewards and Optimality Criteria

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

Definition 2.7. Let the *return* or *cumulative reward* of a Dec-POMDP be defined as the total sum of the rewards the team of agents receives during execution:

$$\sum_{t=0}^{h} R(s^t,\mathbf{a}^t) \tag{2.2.3}$$

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

We consider as optimality criterion the *expected cumulative reward*

$$E\Big[\sum_{t=0}^{h} R(s^t,\mathbf{a}^t)\Big], \tag{2.2.4}$$

where the expectation refers to the expectation over sequences of states and executed joint actions. The planning problem is to find a conditional plan, or *policy,* for each agent to maximize the optimality criterion. In the Dec-POMDP case this amounts to finding a tuple of policies, called a *joint policy* that maximizes the expected cumulative reward. Another frequently used optimality criterion is the discounted expected cumulative reward

$$E\Big[\sum_{t=0}^{h} \gamma^t R(s^t,\mathbf{a}^t)\Big], \tag{2.2.5}$$

where $0 \leq \gamma < 1$ is the discount factor. Discounting is typically used to keep the optimality criterion bounded in infinite horizon problems, but can also be used in finite horizon problems to give higher priority to rewards that are obtained sooner. The regular expected cumulative reward is the special case with $\gamma = 1$. Another optimality criterion we will not consider further is the average reward criterion (Puterman, 1994; Petrik and Zilberstein, 2007)

Note that, in contrast to reinforcement learning settings (Sutton and Barto, 1998), 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, which 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 which is available in the off-line planning phase, not the actual reward obtained. Indeed, it is not even assumed that the actual reward can be observed at the end of the episode.

2.3 Benchmark Problems

This section introduces two example problems: a problem in which several agents have to fight fires and Dec-Tiger, a standard Dec-POMDP benchmark.

num. agents at H	0			1			≥ 2
H burning?	no	yes	*	no	yes	yes	*
burning neighbor?	no	no	yes	*	no	yes	*
$\Pr(x'_H = 0)$	0	0	0	0	0	0	1
$\Pr(x'_H = x_H - 1)$	0	0	0	0	1	0.6	0
$\Pr(x'_H = x_H)$	1	0.6	0.2	1	0	0.4	0
$\Pr(x'_H = x_H + 1)$	0	0.4	0.8	0	0	0	0

Table 2.1: Summary of the transition model for the fire level of a house H in FireFighting. The table shows the probabilities of the 4 possible stochastic effects: the fire in the house is extinguished, the fire level decreases, remains equal or increases.

2.3.1 The FireFighting Problem

The FireFighting problem models a team of n firefighters that have to extinguish fires in a row of N_H houses. At every time step, each agent i can choose to fight fires at each house H. That is $\mathcal{A}_i = \{H_1, \ldots, H_{N_H}\}$. Each agent can only observe whether there are flames, $o_i = F$, or not, $o_i = N$, at its location. Each house H is characterized by a fire level x_H, an integer parameter between 0 and N_f the number of fire levels, where a level of 0 indicates the house is not burning. A state in FireFighting is an assignment of fire levels $s = \langle x_1, \ldots, x_{N_H} \rangle$. Initially, the fire level x_H of each house is drawn from a uniform distribution.

If a house H is burning ($x_H > 0$) and no firefighting 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 there is no agent fighting fire and at least one of its neighbors is on fire. When two agents are in the same house, they extinguish any present fire completely, setting the house's fire level to 0. A single agent present at a house lowers the fire level by one point with probability 1 if no neighbors are burning, and with probability 0.6 otherwise. The transition model is summarized by Table 2.1.

The observations of the agents depend on the fire level of the house they chose to fight fire at. Each agent will observe flames (F) with probability 0.2 if $x_H = 0$, with probability 0.5 if $x_H = 1$, and with probability 0.8 otherwise.

The agents receive a reward of $r^H(s) = -x_H$ for each house H and the total reward R received at a step is the sum of r^H. In particular, the rewards are specified by the fire levels at the next time step

$$R(s,\mathbf{a},s') = \sum_{H=1}^{N_H} r^H(s') = \sum_{H=1}^{N_H} -x'_H$$

where x'_H is the fire level of house H as specified by the next-stage state s'. These rewards can be converted to the $R(s,\mathbf{a})$-form by taking the expectation over s' as follows

$$R(s,\mathbf{a}) = \sum_{s'} \Pr(s'|s,\mathbf{a}) R(s,\mathbf{a},s').$$

$\vec{\mathbf{o}}_\emptyset \rightarrow$ go house 3	$\vec{\mathbf{o}}_\emptyset \rightarrow$ go house 2
flames $\rightarrow$ go house 3 no flames $\rightarrow$ go house 1	flames $\rightarrow$ go house 2 no flames $\rightarrow$ go house 2
flames, flames $\rightarrow$ go house 1 flames, no flames $\rightarrow$ go house 1 no flames, flames $\rightarrow$ go house 2 no flames, no flames $\rightarrow$ go house 2	flames, flames $\rightarrow$ go house 1 flames, no flames $\rightarrow$ go house 1 no flames, flames $\rightarrow$ go house 1 no flames, no flames $\rightarrow$ go house 1

Figure 2.4: Optimal policy for 2-agent 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.

Figure 2.4 shows an optimal joint policy for horizon 3 of the FIREFIGHTING problem with $n = 2, N_H = 3, N_f = 3$. 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.

2.3.2 The Decentralized Tiger Problem

The most used Dec-POMDP benchmark is the decentralized tiger (DEC-TIGER) problem introduced by Nair, Tambe, Yokoo, Pynadath, and Marsella (2003c), which is a modification of the (single-agent) tiger problem (Kaelbling et al., 1998). It concerns two agents that are standing in a hallway with two doors. Behind one of the doors is a tiger, behind the other a treasure. Therefore there are two states: the tiger is behind the left door (s_l) or behind the right door (s_r). Both agents have 3 actions at their disposal: open the left door (a_{OL}), open the right door (a_{OR}) and listen (a_{Li}). But they cannot observe each other's actions. In fact, they can only receive 2 observations. Either they hear a sound left (o_{HL}) or right (o_{HR}).

At $t = 0$ the state is s_l or s_r with probability 0.5. As long as no agent opens a door the state does not change, when a door is opened, the state resets to s_l or s_r with probability 0.5. The full transition, observation and reward model are listed by Nair et al. (2003c). The observation probabilities are independent, and identical for both agents. For instance, when the state is s_l and both perform action a_{Li}, both agents have a 85% chance of observing o_{HL}, and the probability of both hearing the tiger left is $0.85 \cdot 0.85 = 0.72$.

When the agents open the door for the treasure they receive a positive reward (+9), while they receive a penalty for opening the wrong door (−101). When opening the wrong door jointly, the penalty is less severe (−50). Opening the correct door jointly leads to a higher reward (+20).

Note that, when the wrong door is opened by one or both agents, they are attacked by the tiger and receive a penalty. However, neither of the agents observe this attack nor the penalty and the episode continues. Arguably, a more natural representation would be to have the episode end after a door is opened or to let the agents observe whether they encountered the tiger or treasure. As such, the

reader should regard the Dec-Tiger problem purely as a benchmark, rather than a realistic scenario.

2.3.3 Other Problem Domains

Throughout this thesis we will also refer to other test problems. Here we provide very concise descriptions of these problems and pointers to more information.

Apart from the standard Dec-Tiger domain, we consider a modified version, called Skewed Dec-Tiger, in which the start distribution is not uniform. Instead, initially the tiger is located on the left with probability 0.8. The BroadcastChannel was introduced by Hansen, Bernstein, and Zilberstein (2004) and models two nodes that have to cooperate to maximize the throughput of a shared communication channel. Furthermore, a test problem called "Meeting on a Grid" is provided by Bernstein, Hansen, and Zilberstein (2005), in which two robots navigate on a two-by-two grid. We usually consider GridSmall, the version with 2 observations per agent (Amato, Bernstein, and Zilberstein, 2006). Other problems are Cooperative Box Pushing (Seuken and Zilberstein, 2007b) in which 2 robots have to cooperate to move boxes; Recycling Robots (Amato, Bernstein, and Zilberstein, 2007a) which involves two agents that have to recycle trash and Hotel 1, a benchmark where two travel agents have to allocate clients (Spaan and Melo, 2008).

A more elaborate description of these problems is given in Appendix A, which also includes a table that compares the number of agents, states, actions and observations for these problems and gives an indication of the number of joint policies.

2.4 Histories

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. Therefore, before we define a policy, we first need to define exactly what these specific situations are. In essence such situations are those parts of the history of the process that the agents can observe.

Let us first consider what the history of the process is. A Dec-POMDP specifies h time steps or stages $t = 0,...,h-1$. 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 will have to select their k-th actions (at $t = k-1$), the history of the process is a sequence of states, joint observations and joint actions, which has the following form:

$$\left(s^0,\mathbf{o}^0,\mathbf{a}^0,s^1,\mathbf{o}^1,\mathbf{a}^1,...,s^{k-1},\mathbf{o}^{k-1}\right). \tag{2.4.1}$$

Here s^0 is the initial state, drawn according to the initial state distribution $\boldsymbol{b}^0$. The initial joint observation $\mathbf{o}^0$ is assumed to be the empty joint observation: $\mathbf{o}^0 = \mathbf{o}_\emptyset = \langle o_{1,\emptyset},...,o_{n,\emptyset}\rangle$.

From this history of the process, the true states remain unobserved and agent i can only observe its own actions and observations. If the agents would be able to observe the true state at some point, they could disregard the complete history

before that since the states possess the Markov property and thus contain enough information to predict the future. However, since the agents do not observe such a Markovian signal, each agent will have to base its decision regarding which action to select on the complete history of actions and its observations observed up to that point.

Definition 2.8 (Action-observation history)**.** 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

$$\vec{\theta}_i^t = \left(o_i^0, a_i^0, o_i^1, \ldots, a_i^{t-1}, o_i^t\right). \tag{2.4.2}$$

The *joint action-observation history,* $\vec{\boldsymbol{\theta}}$, is the action-observation history for all agents:

$$\vec{\boldsymbol{\theta}}^t = \langle \vec{\theta}_1^t, \ldots, \vec{\theta}_n^t \rangle. \tag{2.4.3}$$

Agent i's set of possible AOHs at time t is $\vec{\Theta}_i^t = \times_t (\mathcal{O}_i \times \mathcal{A}_i)$. The set of all possible AOHs for agent i is $\vec{\Theta}_i = \cup_{t=0}^{h-1} \vec{\Theta}_i^t$.[1] Finally the set of all possible *joint* AOHs is given by $\vec{\boldsymbol{\Theta}} = \cup_{t=0}^{h-1} (\vec{\Theta}_1^t \times ... \times \vec{\Theta}_n^t)$. At $t = 0$, the action-observation history is empty, denoted by $\vec{\boldsymbol{\theta}}^0 = \vec{\boldsymbol{\theta}}_\emptyset$.

We will also use a notion of history only using the observations of an agent.

Definition 2.9 (Observation history)**.** The *observation history (OH) for agent i,* $\vec{o}_i$, is defined as the sequence of observations an agent has received. At a specific time step t, this is:

$$\vec{o}_i^t = \left(o_i^0, o_i^1, \ldots, o_i^t\right). \tag{2.4.4}$$

The *joint observation history,* $\vec{\mathbf{o}}$, is the observation history for all agents:

$$\vec{\mathbf{o}}^t = \langle \vec{o}_1^t, \ldots, \vec{o}_n^t \rangle. \tag{2.4.5}$$

The set of observation histories for agent i at time t is denoted $\vec{\mathcal{O}}_i^t = \times_t \mathcal{O}_i$. Similar to the notation for action-observation histories, we also use $\vec{\mathcal{O}}_i$ and $\vec{\mathcal{O}}$ and the empty observation history is denoted $\vec{\mathbf{o}}_\emptyset$.

Similarly we can define the action history as follows.

Definition 2.10 (Action history)**.** 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:

$$\vec{a}_i^t = \left(a_i^0, a_i^1, \ldots, a_i^{t-1}\right). \tag{2.4.6}$$

Notation for joint action histories and sets are analogous to those for observation histories. Also write $\vec{o}_{\neq i}, \vec{\theta}_{\neq i}$, etc. to denote a tuple of observation-, action-observation histories, etc. for all agents except i. Finally we note that, clearly, a (joint) AOH consists of a (joint) action- and a (joint) observation history: $\vec{\boldsymbol{\theta}}^t = \langle \vec{\mathbf{o}}^t, \vec{\mathbf{a}}^t \rangle$.

[1] 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.

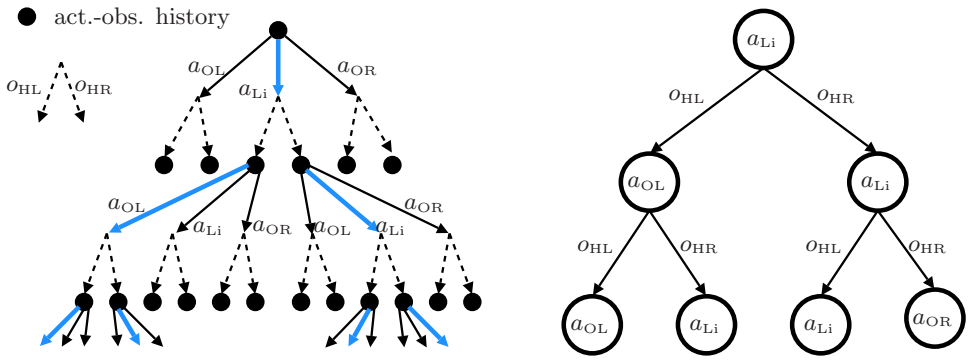


Figure 2.5: A deterministic policy can be represented as a tree. Left: a tree of action-observation histories $\vec{\theta}_i$ for one of the agents from the Dec-Tiger problem. An arbitrary deterministic policy π_i is highlighted. Clearly shown is that π_i only reaches a subset of of histories $\vec{\theta}_i$. ($\vec{\theta}_i$ that are not reached are not further expanded.) Right: The same policy can be shown in a simplified policy tree.

2.5 Policies

As discussed in the previous section, 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 sequences to actions.

The number of possible AOHs is usually very large as this set grows exponentially with the horizon of the problem. At time step t, there are $(|\mathcal{A}_i| \cdot |\mathcal{O}_i|)^t$ action-observation histories for agent i. As a consequence there are a total of

$$\sum_{t=0}^{h-1} (|\mathcal{A}_i| \cdot |\mathcal{O}_i|)^t = \frac{(|\mathcal{A}_i| \cdot |\mathcal{O}_i|)^h - 1}{(|\mathcal{A}_i| \cdot |\mathcal{O}_i|) - 1}$$

of such sequences for agent i. Therefore the number of policies for agent i becomes:

$$|\mathcal{A}_i|^{\frac{(|\mathcal{A}_i| \cdot |\mathcal{O}_i|)^h - 1}{(|\mathcal{A}_i| \cdot |\mathcal{O}_i|) - 1}}, \tag{2.5.1}$$

which is doubly exponential in the horizon h.

2.5.1 Pure and Stochastic Policies

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 Figure 2.5, 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 in order to specify a deterministic

policy, the observation history suffices: when an agent takes its actions deterministically, it will be able to infer what action it took from only the observation history as illustrated by the right side of Figure 2.5.

Definition 2.11. A *pure* or *deterministic policy*, π_i^p, for agent i in a Dec-POMDP is a mapping from observation histories to actions, $\pi_i^p : \vec{\mathcal{O}}_i \rightarrow \mathcal{A}_i$. The set of pure policies of agent i is denoted Π_i.

Note that also for pure policies we sometimes write $\pi_i^p(\vec{\theta}_i)$. In this case we mean the action that π_i^p specifies for the observation history contained in $\vec{\theta}_i$. For instance, let $\vec{\theta}_i = \langle \vec{o}_i, \vec{a}_i \rangle$, then $\pi_i^p(\vec{\theta}_i) \equiv \pi_i^p(\vec{o}_i)$. Whether $\pi_i^p(\vec{\theta}_i)$ denotes such a pure policy as described here, or a more general policy $\pi_i : \vec{\Theta}_i \rightarrow \mathcal{A}_i$, should be clear from the context. We use $\boldsymbol{\pi} = \langle \pi_1, ..., \pi_n \rangle$ to denote a *joint policy*, a profile specifying a policy for each agent. We say that a pure joint policy is an *induced* or *implicit mapping* from joint observation histories to joint actions $\boldsymbol{\pi} : \vec{\mathcal{O}} \rightarrow \mathcal{A}$. That is, the mapping is induced by individual policies π_i that make up the joint policy. Also we use $\boldsymbol{\pi}_{\neq i} = \langle \pi_1, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_n \rangle$, to denote a profile of policies for all agents except 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. There are two types of randomized policies: mixed policies and stochastic policies.

Definition 2.12. A *mixed policy,* π_i^m, for an agent i is a set of pure policies, $\mathcal{M} \subseteq \Pi_i$, along with a probability distribution over this set. Thus a mixed policy $\pi_i^m \in \mathcal{P}(\mathcal{M})$ is an element of the set of probability distributions over $\mathcal{M}$.

Definition 2.13. A *stochastic* or *behavioral policy*, π_i^s, for agent i is a mapping from action-observation histories to probability distributions over actions, $\pi_i^s : \vec{\Theta}_i \rightarrow \mathcal{P}(\mathcal{A}_i)$. The probability of a_i taken for $\vec{\theta}_i$ according to π_i^s is written $\pi_i^s(a_i|\vec{\theta}_i)$.

When considering stochastic policies, keeping track of only the observations is insufficient, as in general all action-observation histories can be realized. That is why stochastic policies are defined as a mapping from the full space of action-observation histories to probability distributions over actions. Note that we use π_i and Π_i to denote a policy (space) in general, so also for randomized policies. We will only use π_i^p, π_i^m and π_i^s when there is a need to discriminate between different types of policies.

2.5.2 Temporal Structure in Policies

Policies specify actions for all stages of the Dec-POMDP. A common way to represent the temporal structure in a policy is to split it 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 case of a deterministic policy, the form of the decision rule for stage t is a mapping from length-t observation histories to actions $\delta_i^t : \vec{\mathcal{O}}_i^t \rightarrow \mathcal{A}_i$. The most general form of the decision rule for stage t is a mapping

from stage t action-observation histories to actions $\delta_i^t : \vec{\Theta}_i^t \to \mathcal{A}_i$. A joint decision rule $\boldsymbol{\delta}^t = \langle \delta_1^t, \ldots, \delta_n^t \rangle$ specifies a decision rule for each agent.

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

Definition 2.14 (Policy concatenation)**.** We write

$$\boldsymbol{\varphi}^{t+1} = (\boldsymbol{\delta}^0, \ldots, \boldsymbol{\delta}^{t-1}, \boldsymbol{\delta}^t) = \langle \boldsymbol{\varphi}^t \circ \boldsymbol{\delta}^t \rangle \tag{2.5.2}$$

to denote policy concatenation.

To complete the overview of possible policy constructs with respect to time, we also define future policies. A future policy ψ_i^t of agent i specifies all the future behavior. That is, $\psi_i^t = \left(\delta_i^{t+1}, \ldots, \delta_i^{h-1}\right)$. We also consider future joint policies $\boldsymbol{\psi}^t = (\boldsymbol{\delta}^{t+1}, \ldots, \boldsymbol{\delta}^{h-1})$.

Summarizing, the structure of a policy π_i can be represented as follows

$$\pi_i = (\underbrace{\delta_i^0, \delta_i^1, \ldots, \delta_i^{t-1}}_{\varphi_i^t}, \delta_i^t, \underbrace{\delta_i^{t+1}, \ldots, \delta_i^{h-1}}_{\psi_i^t}) \tag{2.5.3}$$

and similarly for joint policies.

2.5.3 The Quality of Joint Policies

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

Definition 2.15. The *value* $V(\boldsymbol{\pi})$ of a joint policy $\boldsymbol{\pi}$ is defined as

$$V(\boldsymbol{\pi}) \;\equiv\; E\Big[\sum_{t=0}^{h-1} R(s^t, \mathbf{a}^t) \Big| \boldsymbol{\pi}, \boldsymbol{b}^0\Big], \tag{2.5.4}$$

where the expectation is over states, observations and—in the case of a randomized $\boldsymbol{\pi}$—actions.

In particular we can compute this expectation using

$$V(\boldsymbol{\pi}) = \sum_{t=0}^{h-1} \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t} \sum_{s^t \in \mathcal{S}} \Pr(s^t, \vec{\boldsymbol{\theta}}^t | \boldsymbol{\pi}, \boldsymbol{b}^0) \sum_{\mathbf{a}^t \in \mathcal{A}} R(s^t, \mathbf{a}^t) \boldsymbol{\pi}(\mathbf{a}^t | \vec{\boldsymbol{\theta}}^t), \tag{2.5.5}$$

where $\boldsymbol{\pi}(\mathbf{a}^t | \vec{\boldsymbol{\theta}}^t)$ is the probability of $\mathbf{a}$ as specified by the possibly stochastic joint policy $\boldsymbol{\pi}$, and where $\Pr(s^t, \vec{\boldsymbol{\theta}}^t | \boldsymbol{\pi}, \boldsymbol{b}^0)$ is recursively defined as

$$\Pr(s^t, \vec{\boldsymbol{\theta}}^t | \boldsymbol{\pi}, \boldsymbol{b}^0) = \sum_{s^{t-1} \in \mathcal{S}} \Pr(\mathbf{o}^t | \mathbf{a}^{t-1}, s^t) \Pr(s^t | s^{t-1}, \mathbf{a}^{t-1}) \boldsymbol{\pi}(\mathbf{a}^{t-1} | \vec{\boldsymbol{\theta}}^{t-1}) \Pr(s^{t-1}, \vec{\boldsymbol{\theta}}^{t-1} | \boldsymbol{\pi}, \boldsymbol{b}^0), \tag{2.5.6}$$

For stage 0 we have that $\Pr(s^0,\vec{\boldsymbol{\theta}}_\emptyset|\boldsymbol{\pi},\boldsymbol{b}^0) = \boldsymbol{b}^0(s^0)$.

Because of the recursive nature of $\Pr(s^t,\vec{\boldsymbol{\theta}}^t|\boldsymbol{\pi},\boldsymbol{b}^0)$ it is more intuitive to specify the value recursively:

$$V_{\boldsymbol{\pi}}(s^t,\vec{\boldsymbol{\theta}}^t) = \sum_{\mathbf{a}^t\in\mathcal{A}} \boldsymbol{\pi}(\mathbf{a}^t|\vec{\boldsymbol{\theta}}^t) \left[R(s^t,\mathbf{a}^t) + \sum_{s^{t+1}\in\mathcal{S}} \sum_{\mathbf{o}^{t+1}\in\mathcal{O}} \Pr(s^{t+1},\mathbf{o}^{t+1}|s^t,\mathbf{a}^t) V_{\boldsymbol{\pi}}(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1})\right], \quad (2.5.7)$$

with $\vec{\boldsymbol{\theta}}^{t+1} = (\vec{\boldsymbol{\theta}}^t,\mathbf{a}^t,\mathbf{o}^{t+1})$. The value of joint policy $\boldsymbol{\pi}$ is then given by

$$V(\boldsymbol{\pi}) = \sum_{s^0\in\mathcal{S}} V_{\boldsymbol{\pi}}(s^0,\vec{\boldsymbol{\theta}}_\emptyset)\boldsymbol{b}^0(s^0). \quad (2.5.8)$$

By splitting $\Pr(s^t,\vec{\boldsymbol{\theta}}^t|\boldsymbol{\pi},\boldsymbol{b}^0)$ in a marginal and a conditional, for the special case of evaluating a pure joint policy $\boldsymbol{\pi}$, (2.5.5) can be written as

$$V(\boldsymbol{\pi}) = \sum_{t=0}^{h-1} \sum_{\vec{\boldsymbol{\theta}}^t\in\vec{\boldsymbol{\Theta}}^t} \Pr(\vec{\boldsymbol{\theta}}^t|\boldsymbol{\pi},\boldsymbol{b}^0) R(\vec{\boldsymbol{\theta}}^t,\boldsymbol{\pi}(\vec{\boldsymbol{\theta}}^t)), \quad (2.5.9)$$

where

$$R(\vec{\boldsymbol{\theta}}^t,\mathbf{a}^t) = \sum_{s^t\in\mathcal{S}} R(s^t,\mathbf{a}^t)\Pr(s^t|\vec{\boldsymbol{\theta}}^t,\boldsymbol{b}^0) \quad (2.5.10)$$

denotes the expected immediate reward. In this case, the recursive formulation (2.5.7) reduces to[1]

$$V^t_{\boldsymbol{\pi}}(s^t,\vec{\mathbf{o}}^t) = R\left(s^t,\pi(\vec{\mathbf{o}}^t)\right) + \sum_{s^{t+1}\in\mathcal{S}} \sum_{\mathbf{o}^{t+1}\in\mathcal{O}} \Pr(s^{t+1},\mathbf{o}^{t+1}|s^t,\pi(\vec{\mathbf{o}}^t)) V^{t+1}_{\boldsymbol{\pi}}(s^{t+1},\vec{\mathbf{o}}^{t+1}). \quad (2.5.11)$$

2.5.4 Existence of an Optimal Pure Joint Policy

Although randomized policies may be useful, we can restrict our attention to pure policies without sacrificing optimality, as shown by the following.

Proposition 2.1 (Existence optimal pure joint policy)**.** *A finite horizon Dec-POMDP has at least one optimal pure joint policy.*

Proof. This proof follows a proof by Schoute (1978). It is possible to convert a Dec-POMDP to an extensive game and thus to a strategic game, in which the actions are pure policies for the Dec-POMDP (Oliehoek and Vlassis, 2006). In this strategic game, there is at least one maximizing entry corresponding to a pure joint policy which we denote $\boldsymbol{\pi}_{\max}$. Now, assume that there is a joint stochastic

[1] Note that, intermediate results should be cached. A particular $(s^{t+1},\vec{\mathbf{o}}^{t+1})$-pair can be reached from $|\mathcal{S}|$ states s^t of the previous stage.

policy $\boldsymbol{\pi}^s = \langle \pi_1^s, \ldots, \pi_n^s \rangle$ that attains a higher payoff. Kuhn (1953) showed that for each stochastic π_i^s policy, there is a corresponding mixed policy π_i^m. Therefore $\boldsymbol{\pi}^s$ corresponds to a joint mixed policy $\boldsymbol{\pi}^m = \langle \pi_1^m, \ldots, \pi_n^m \rangle$. Clearly a mixed joint policy can never achieve a higher value than the maximum of the pure joint policies it assigns positive weight to, formalized in the following. Let us write Π_i^m for the support of π_i^m. $\boldsymbol{\pi}^m$ now induces a probability distribution p^m over the set of joint policies $\mathbf{\Pi}^m = \Pi_1^m \times \cdots \times \Pi_n^m \subseteq \mathbf{\Pi}$ which is a subset of the set of all joint policies. The expected payoff can now be written as

$$V(\boldsymbol{\pi}^s) = E_{p^m}(V(\boldsymbol{\pi}) | \boldsymbol{\pi} \in \mathbf{\Pi}^m) \leq \max_{\boldsymbol{\pi} \in \mathbf{\Pi}^m} V(\boldsymbol{\pi}) \leq \max_{\boldsymbol{\pi} \in \mathbf{\Pi}} V(\boldsymbol{\pi}) = V(\boldsymbol{\pi}_{\max}),$$

contradicting that $\boldsymbol{\pi}^s$ is a joint stochastic policy that attains a higher payoff. □

2.6 Solving Dec-POMDPs

This section gives an overview of methods proposed for finding (approximate) solutions for finite horizon Dec-POMDPs. Because the infinite-horizon problem is significantly different from the finite-horizon case, we will not go into details of work performed on infinite-horizon Dec-POMDPs, such as the work by Peshkin, Kim, Meuleau, and Kaelbling (2000); Bernstein et al. (2005); Szer and Charpillet (2005); Amato et al. (2006, 2007a); Amato and Zilberstein (2009). A major problem in this setting is how to represent policies. A common choice is to use finite state controllers (FSCs). Since an infinite-horizon Dec-POMDP is undecidable (Bernstein, Givan, Immerman, and Zilberstein, 2002), most research in this line of work focuses on approximate solutions (e.g., Bernstein et al., 2005) or finding optimal solutions *given a particular controller size* (Amato et al., 2006, 2007a).

2.6.1 Complexity

Section 2.5.4 assures that an optimal solution can be found in the set of pure joint policies. Still, the number of such pure joint policies is doubly exponential in the horizon h, which provides some intuition about how hard the problem is. This intuition is supported by the complexity result due to Bernstein et al. (2000).

Theorem 2.2 (Dec-POMDP complexity). *The problem of finding the optimal solution for a finite horizon Dec-POMDP with $n \geq 2$ is NEXP-complete.*

Moreover, Dec-POMDPs cannot be approximated efficiently: Rabinovich, Goldman, and Rosenschein (2003) showed that even finding an ϵ-approximate solution is NEXP-hard. As mentioned, the infinite-horizon problem is undecidable, which is a direct result of the undecidability of (single-agent) POMDPs over an infinite horizon (Madani, Hanks, and Condon, 1999).

2.6.2 Brute Force Policy Evaluation

Because there exists an optimal pure joint policy for a finite-horizon Dec-POMDP, it is in theory possible to enumerate all different pure joint policies, evaluate them

using equations (2.5.8) and (2.5.11) and choose the best one. The number of pure joint policies to be evaluated is

$$O\left(|\mathcal{A}_*|^{\frac{n(|\mathcal{O}_*|^h-1)}{|\mathcal{O}_*|-1}}\right), \tag{2.6.1}$$

where $|\mathcal{A}_*|$ and $|\mathcal{O}_*|$ denote the largest individual action and observation sets. The cost of evaluating each joint policy is $O\left(|\mathcal{S}| \cdot |\mathcal{O}_*|^{nh}\right)$. The resulting total cost of brute-force policy evaluation is

$$O\left(|\mathcal{A}_*|^{\frac{n(|\mathcal{O}_*|^h-1)}{|\mathcal{O}_*|-1}} \cdot |\mathcal{S}| \cdot |\mathcal{O}_*|^{nh}\right), \tag{2.6.2}$$

which is doubly exponential in the horizon h.

2.6.3 Alternating Maximization

Nair et al. (2003c) introduced *Joint Equilibrium based Search for Policies (JESP).* This method guarantees to find a locally optimal joint policy, more specifically, a *Nash equilibrium*: a tuple of policies such that for each agent i its policy π_i is a best response for the policies employed by the other agents $\pi_{\neq i}$. It relies on a process we refer to as *alternating maximization.* This is a procedure that computes a policy π_i for an agent i that maximizes the joint reward, while keeping the policies of the other agents fixed. Next, another agent is chosen to maximize the joint reward by finding its best-response to the fixed policies of the other agents. This process is repeated until the joint policy converges to a Nash equilibrium, which is a local optimum. The main idea of fixing some agents and having others improve their policy was presented before by Chades, Scherrer, and Charpillet (2002), but they used a heuristic approach for memory-less agents. The process of alternating maximization is also referred to as *hill-climbing* or *coordinate ascent.*

Nair et al. (2003c) describe two variants of JESP, the first of which, Exhaustive-JESP, implements the above idea in a very straightforward fashion: Starting from a random joint policy, the first agent is chosen. This agent then selects its best-response policy by evaluating the joint reward obtained for all of its individual policies when the other agents follow their fixed policy.

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

Like most methods proposed for Dec-POMDPs, JESP exploits the knowledge of the initial belief $\boldsymbol{b}^0$ by only considering reachable beliefs $b(\bar{s})$ in the solution of the POMDP. However, in some cases the initial belief might not be available. As demonstrated by Varakantham, Nair, Tambe, and Yokoo (2006), JESP can be extended to plan for the entire space of initial beliefs, overcoming this problem.

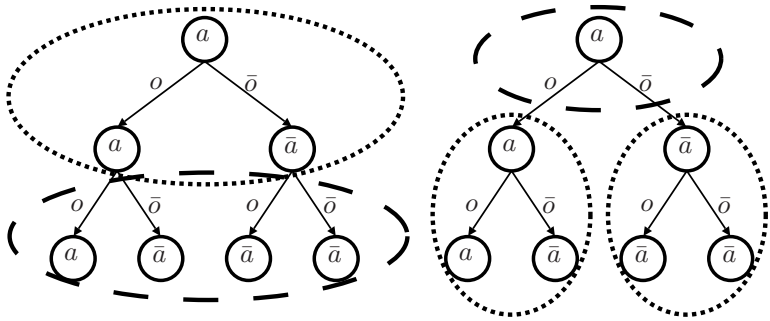


Figure 2.6: Difference between policy construction in MAA* (left) and dynamic programming (right) for an agent with actions a,$\bar{a}$ and observations o,$\bar{o}$. The dashed components are newly generated, dotted components result from the previous iteration. MAA* expands a partial policy from the leaves, while dynamic programming backs up a set of sub-tree policies forming new ones.

2.6.4 Multiagent A* (MAA*)

Szer, Charpillet, and Zilberstein (2005) introduced a heuristically guided policy search method called *multiagent* A* *(*MAA**)*. 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. Note that this implies that the planning process *itself* can be reformulated as a shortest path problem.

In particular MAA* considers joint policies that are partially specified with respect to time: a partial joint policy $\boldsymbol{\varphi}^t = (\boldsymbol{\delta}^0,\boldsymbol{\delta}^1,\ldots,\boldsymbol{\delta}^{t-1})$ specifies the joint decision rules for the first t stages. For such a partial joint policy $\boldsymbol{\varphi}^t$ a heuristic value $\widehat{V}(\boldsymbol{\varphi}^t)$ is calculated by taking $V^{0\ldots t-1}(\boldsymbol{\varphi}^t)$, the actual expected reward $\boldsymbol{\varphi}^t$ 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 $\widehat{V}(\boldsymbol{\varphi}^t)$.

MAA* starts by placing the completely unspecified joint policy $\boldsymbol{\varphi}^0$ in an open list. Then, it proceeds by selecting partial joint policies $\boldsymbol{\varphi}^t = (\boldsymbol{\delta}^0,\boldsymbol{\delta}^1,\ldots,\boldsymbol{\delta}^{t-1})$ from the list and expanding them: generating all $\boldsymbol{\varphi}^{t+1} = (\boldsymbol{\delta}^0,\boldsymbol{\delta}^1,\ldots,\boldsymbol{\delta}^{t-1},\boldsymbol{\delta}^t) = \langle\boldsymbol{\varphi}^t \circ \boldsymbol{\delta}^t\rangle$ by appending all possible joint decision rules $\boldsymbol{\delta}^t$ for the next time step (t). The left side of Figure 2.6 illustrates the expansion process. After expansion, all created children are heuristically valuated and placed in the open list, any partial joint policies $\boldsymbol{\varphi}^{t+1}$ with $\widehat{V}(\boldsymbol{\varphi}^{t+1})$ less than the expected value $V(\boldsymbol{\pi})$ of some earlier found (fully specified) joint policy $\boldsymbol{\pi}$, can be pruned. The search ends when the list becomes empty, at which point an optimal fully specified joint policy is found.

2.6.5 Dynamic Programming for Dec-POMDPs

MAA* incrementally builds policies from the first stage $t = 0$ to the last $t = h-1$. Prior to this work, Hansen, Bernstein, and Zilberstein (2004) introduced dynamic programming (DP) for Dec-POMDPs, which 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 MAA*. A partial policy in MAA* has the form $\boldsymbol{\varphi}^t = (\boldsymbol{\delta}^0, \boldsymbol{\delta}^1, \ldots, \boldsymbol{\delta}^{t-1})$. 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

$$\tau = h - t. \tag{2.6.3}$$

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 $\mathcal{Q}_i^{\tau=k}$. Dynamic programming for Dec-POMDPs is based on *backup* operations: constructing $\mathcal{Q}_i^{\tau=k+1}$ from $\mathcal{Q}_i^{\tau=k}$. For instance, the right side of Figure 2.6 shows how $q_i^{\tau=3}$, a 3-steps-to-go sub-tree policy, is constructed from two $q_i^{\tau=2} \in \mathcal{Q}_i^{\tau=2}$. Also illustrated is the difference between this process and MAA* expansion (on the left side).

Dynamic programming consecutively constructs $\mathcal{Q}_i^{\tau=1}, \mathcal{Q}_i^{\tau=2}, \ldots, \mathcal{Q}_i^{\tau=h}$ for all agents i. However, the size of the set $\mathcal{Q}_i^{\tau=k+1}$ constructed by an exhaustive backup is given by

$$|\mathcal{Q}_i^{\tau=k+1}| = |\mathcal{A}_i| \, |\mathcal{Q}_i^{\tau=k}|^{|\mathcal{O}_i|},$$

and as a result the sizes of the maintained sets grow doubly exponential with k (note that since the $q_i^{\tau=k}$ are essentially full policies for the horizon-k problem, their number is doubly exponential in k).

To counter this source of intractability, Hansen et al. (2004) 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 $q_j^{\tau=k} \in \mathcal{Q}_j^{\tau=k}$. In fact, if the other agents use a fixed (possibly mixed) policy, the problem can be interpreted as a single agent POMDP as described in Section 2.6.3, as such it is possible to reuse some ideas from POMDP solutions. If we let $q_{\neq i}^{\tau=k}$ denote a sub-tree profile for all agents but i, and $\mathcal{Q}_{\neq i}^{\tau=k}$ 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 $S \times \mathcal{Q}_{\neq i}^{\tau=k}$. 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* (Osborne and Rubinstein, 1994). Finally, when the last backup step is completed the optimal policy can be found by evaluating all joint policies $\boldsymbol{\pi} \in \mathcal{Q}_1^{\tau=h} \times \cdots \times \mathcal{Q}_n^{\tau=h}$ for the initial belief $\boldsymbol{b}^0$.

2.6.5.1 Extensions on DP for Dec-POMDPs

In the last few years several extensions to the dynamic programming algorithm for Dec-POMDPs have been proposed. The first of these extensions is due to Szer

and Charpillet (2006). Rather than testing for dominance over the entire multiagent belief space, Szer and Charpillet propose to perform point-based dynamic programming (PBDP). In order to prune the set of sub-tree policies $\mathcal{Q}_i^{\tau=k}$, the set of all the belief points $\mathcal{B}_{i,\text{reachable}} \subset \mathcal{P}(S \times \mathcal{Q}_{\neq i}^{\tau=k})$ that can possibly be reached by deterministic joint policies are generated. Only the sub-tree policies $q_i^{\tau=k}$ that maximize the value at some $b_i \in \mathcal{B}_{i,\text{reachable}}$ are kept. The proposed algorithm is optimal, but intractable because it needs to generate all the multiagent belief points that are reachable through all joint policies. To overcome this bottleneck, Szer and Charpillet propose to randomly sample one or more joint policies and use those to generate $\mathcal{B}_{i,\text{reachable}}$.

Seuken and Zilberstein (2007a) also proposed a point-based extension of the DP algorithm, called memory-bounded dynamic programming (MBDP). Rather than using a randomly selected policy to generate the belief points, they propose to use heuristic policies. A more important difference, however, lies in the pruning step. 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 *maxTrees* sub-tree policies are retained, which is a parameter of the planning method. 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 $\mathcal{Q}_i^{\tau=k+1}$ from $\mathcal{Q}_i^{\tau=k}$ which now contain $|\mathcal{A}_i|\, maxTrees^{|\mathcal{O}_i|}$ sub-tree policies. Moreover, in each iteration all $\left(|\mathcal{A}_*|\, maxTrees^{|\mathcal{O}_*|}\right)^n$ joint sub-tree policies have to be evaluated for each of the sampled belief points. To counter this growth, Seuken and Zilberstein (2007b) proposed an extension that limits the considered observations during the backup step to the *maxObs* most likely observations. This approach is refined by Carlin and Zilberstein (2008) who do not prune low-probability observations, but rather cluster observations together as to minimize the expected loss.

The method of Carlin and Zilberstein (2008) is able to bound the error introduced relative to MBDP without observation compression. The error caused by MBDP itself, however, is unbounded. Recently, Boularias and Chaib-draa (2008) proposed an algorithm based on DP that performs a lossless compression of the sub-tree policies and thus a more efficient optimal algorithm. The idea is that sub-tree policies are represented in a smaller more compact form by using ideas from the work on predictive state representations (Singh, James, and Rudary, 2004a).

A different extension of the DP for Dec-POMDPs algorithm is given by Amato, Carlin, and Zilberstein (2007b). Their approach, bounded DP (BDP), establishes a bound not on the used memory, but on the quality of approximation. In particular, BDP uses ϵ-pruning in each iteration. That is, a $q_i^{\tau=k}$ that is maximizing in some region of the multiagent belief space, but improves the value in this region by at most ϵ, is also pruned. Because iterated elimination using ϵ- pruning can still lead to an unbounded reduction in value, Amato et al. propose to perform one iteration of ϵ-pruning, followed by iterated elimination using normal pruning.

More recently, two papers have introduced new techniques to counter the complexity induced by the exhaustive backup. Dibangoye, Mouaddib, and Chai-draa (2009) replace the exhaustive backup performed in the dynamic programming algorithm by a branch-and-bound search. Amato, Dibangoye, and Zilberstein (2009)

introduce two techniques to improve the efficiency of DP techniques: First, incremental policy generation independently generates sets $\mathcal{Q}_i^{\tau=k+1,a_i,o_i}$ of useful (non-dominated) policies $q_i^{\tau=k+1}$ for each a_i, o_i separately. Second, by analyzing which states can occur, the amount of pruning is increased.

2.6.6 Other Finite-Horizon Methods

There are a few other approaches for finite-horizon Dec-POMDPs, which we will only briefly describe here. Aras, Dutech, and Charpillet (2007) proposed a mixed integer linear programming formulation for the optimal solution of finite-horizon Dec-POMDPs. Their approach is based on representing the set of possible policies for each agent in *sequence form* (Romanovskii, 1962; Koller, Megiddo, and von Stengel, 1994; Koller and Pfeffer, 1997).[1] In this representation, a single policy for an agent i is represented as a subset of the set of *sequences* (roughly corresponding to action-observation histories) for the agent. As such the problem can be interpreted as a combinatorial optimization problem, which Aras et al. propose to solve with a mixed integer linear program.

Oliehoek et al. (2007a, 2008a) also recognize that finding a solution for Dec-POMDPs in essence is a combinatorial optimization problem and propose to apply the Cross-Entropy method (de Boer, Kroese, Mannor, and Rubinstein, 2005), a method for combinatorial optimization that recently has become popular because of its ability to find near-optimal solutions in large optimization problems. The resulting algorithm DICE performs a sampling-based policy search for approximately solving Dec-POMDPs. It operates by sampling pure policies from an appropriately parametrized stochastic policy, and then evaluates these policies either exactly or approximately in order to define the next stochastic policy to sample from, and so on until convergence.

Finally, Emery-Montemerlo et al. (2004, 2005) proposed to approximate Dec-POMDPs through series of Bayesian games. Most of the work in this thesis is based on the same representation, and a detailed explanation is deferred to Chapter 3. We do mention here that while Emery-Montemerlo et al. assume that the algorithm is run on-line (interleaving planning and execution), no such assumption is necessary. Rather we will apply the same framework during a off-line planning phase, just like the other algorithms covered in this overview.

2.7 Generalization: Partially Observable Stochastic Games

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 want to maximize their individual expected cumulative reward.

[1] Both the method of Boularias and Chaib-draa (2008) and the work on predictive state representations are closely related to the sequence form.

The consequence of this is that there is no longer an optimal joint policy, simply because optimality is no longer defined. Rather the joint policy 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 (see Section 2.6.3) are no longer guaranteed to converge for POSGs. The dynamic programming method proposed by Hansen et al. (2004) does apply to POSGs: it finds the set of non-dominated policies for each agent.

Even though the consequences of switching to self-interested agents are severe from a computational perspective, from a modeling perspective the Dec-POMDP and POSG framework are very similar. In particular all dynamics with respect to transitions and observations are identical, and therefore computation of probabilities of action-observation histories and joint beliefs transfers to the POSG setting. As such, even though solution methods presented in this thesis may not transfer directly to the POSG case, the modeling aspect largely does. Therefore we expect that it should be possible to extend (parts of) these methods to POSG settings even if this is non-trivial.

2.8 Special Cases

Because of the negative complexity results for Dec-POMDPs, much research has focused on special cases of Dec-POMDPs. This section treats some special cases that are relevant for the work reported in this thesis. Related work that exploits particular independence assumptions for factored Dec-POMDPs is reported in Chapter 5. For a more comprehensive overview of all the special cases, the reader is referred to the articles by Pynadath and Tambe (2002b); Goldman and Zilberstein (2004); Seuken and Zilberstein (2008).

2.8.1 Degrees of Observability

Researchers have identified different categories of observation functions corresponding to degrees of observability (Pynadath and Tambe, 2002b; Goldman and Zilberstein, 2004). 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 *multiagent Markov decision process (MMDP)* introduced by Boutilier (1996).

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 $\boldsymbol{\pi}^*$ can be found efficiently[1] with

[1]Solving an MDP is P-complete (Papadimitriou and Tsitsiklis, 1987), but the underlying

standard dynamic programming techniques (Puterman, 1994). Such a solution $\boldsymbol{\pi}^* = (\boldsymbol{\delta}^0, \ldots, \boldsymbol{\delta}^{h-1})$ specifies a mapping from states to joint actions for each stage $\forall_t\ \boldsymbol{\delta}^t : \mathcal{S} \rightarrow \mathcal{A}$ and can be split into individual policies $\pi_i = \left(\delta_i^0, \ldots, \delta_i^{h-1}\right)$ with $\forall_t\ \delta_i^t : \mathcal{S} \rightarrow \mathcal{A}_i$ 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, $\forall_i\ \mathcal{O}_i = \left\{o_{i,\emptyset}\right\}$. Under non-observability agents can only employ an open-loop plan. A result of this is that the non-observable setting is easier from a complexity point of view (NP-complete, Pynadath and Tambe 2002b).

Between these two extremes there are partially observable problems which are the focus of this thesis. 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 2.16 (Dec-MDP)**.** A jointly observable Dec-POMDP is referred to as a *Dec-MDP.*

Even though all observations 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 (Bernstein et al., 2002).

2.8.2 The Single Agent Case

In case there is only one agent, the Dec-POMDP reduces to a standard POMDP. In a POMDP, the agent cannot observe the state, so it is not possible to specify a policy as a mapping from states to actions. However, it turns out that maintaining a probability distribution over states, called *belief*, $b \in \mathcal{P}(\mathcal{S})$, is a sufficient statistic:

$$\forall_{s^t} \quad b^t(s^t) \equiv \Pr(s^t|o^t,a^{t-1},o^{t-1},\ldots,a^0,o^0). \tag{2.8.1}$$

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 $\forall t\ \delta^t : \mathcal{P}(\mathcal{S}) \rightarrow \mathcal{A}$.

Unfortunately, in the general Dec-POMDP case considered in this thesis, no such space-saving simplifications of the policy are possible. Even though the transition and observation model can be used to compute a *joint* belief $\boldsymbol{b}$, this computation requires knowledge of the joint actions and observations. During execution, the agents simply have no access to this information and thus can not compute a joint belief.

2.8.3 Communication

The main focus of this thesis is the regular Dec-POMDP, i.e., the setting without explicit communication. The Dec-POMDP, however, has been extended to explicitly incorporate communication actions, and observations. The resulting model, the

MDP of a Dec-POMDP still has size exponential in the number of agents. However, given the MMDP representation for a particular (typically small) number of agents, solution is efficient.

Dec-POMDP-Com (Goldman and Zilberstein, 2003, 2004) 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 multiagent team decision problem (MTDP) that was mentioned in Section 2.2, has a similar extension, called the Com-MTDP (Pynadath and Tambe, 2002b), 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 (2003)

That is, in this perspective the semantics of the communication actions become part of the optimization problem. This problem is considered by Xuan, Lesser, and Zilberstein (2001); Goldman and Zilberstein (2003); Spaan, Gordon, and Vlassis (2006); Goldman, Allen, and Zilberstein (2007).

One can also consider the case where messages have fixed semantics. In such a case the agents need a mechanism to process these semantics (i.e., to allow the messages to affect their beliefs). In the Com-MTDP, the extended belief performs this function. The same functionality is also assumed in the Dec-POMDP-Com. For instance, when the agents share their local observations, each agent maintains a joint belief and performs an update of this joint belief, rather than maintaining the list of observations. It was shown by Pynadath and Tambe (2002b) that under cost-free communication, a joint communication policy that shares the local observations at each stage is optimal. Since intuitively this also makes sense, much research has investigated sharing local observations in models similar to the Dec-POMDP-Com (Ooi and Wornell, 1996; Pynadath and Tambe, 2002b; Nair, Roth, and Yohoo, 2004; Becker, Lesser, and Zilberstein, 2005; Roth, Simmons, and Veloso, 2005a,b; Spaan et al., 2006; Oliehoek et al., 2007b; Roth, Simmons, and Veloso, 2007; Goldman and Zilberstein, 2008).

Although models with explicit communication seem more general than the models without, it is possible to transform the former to the latter. That is, a Dec-POMDP-Com can be transformed to a Dec-POMDP (Goldman and Zilberstein, 2004; Seuken and Zilberstein, 2008). The complexity results also transfer to these models. This means that although this thesis focuses on the regular Dec-POMDP setting, contributions transfer to the case of general communication.

2.9 Summary

This chapter presented an overview of both game-theoretic as decision-theoretic models for multiagent decision making. The main emphasis in this overview was

on the decentralized POMDP and its solution over a finite horizon, which also is the focus of the rest of the thesis. In particular, the presented class of dynamic programming solution methods give a backwards perspective on the solution of Dec-POMDPs: a solution is constructed incrementally starting from the last stage and moving back through time. The next two chapters, in contrast, will present a forward perspective on the solution of Dec-POMDPs. This perspective will use the introduced single-shot Bayesian games.

Chapter 3

Optimal Value Functions for Dec-POMDPs

Over the last half century, the single-agent MDP framework has received much attention, and many results are known (Bellman, 1957b,a; Howard, 1960; Puterman, 1994; Sutton and Barto, 1998; Bertsekas, 2007). In particular it is known that an optimal plan, or policy, can be extracted from the optimal action-value, or *Q-value,* function $Q^*(s,a)$, and that the latter can be calculated efficiently. For POMDPs, similar results are available, although finding an optimal solution is harder (PSPACE-complete for finite-horizon problems, Papadimitriou and Tsitsiklis, 1987).

This chapter discusses how value functions can be computed for decentralized settings, and how policies can be extracted from those value functions. In particular, it treats the value functions that result of superimposing different assumptions with respect to communication on top of the Dec-POMDP model. The common assumption is that this communication is noise and cost free, the differences are with respect to the delay this communication incurs. We consider:

- The plain Dec-POMDP setting which assumes no communication. This can be interpreted as a delay of at least h stages.
- Instantaneous communication.
- One-step delayed communication.
- k-steps delayed communication.

The main contribution of this chapter lies in the contribution to the theory of Dec-POMDPs. In particular, it addresses the previously outstanding issue whether Q-value functions can be defined for Dec-POMDPs just as in (PO)MDPs, and whether an optimal policy can be extracted from such Q-value functions. Most algorithms for planning in Dec-POMDPs are based on some version of policy search, and a proper theory of Q-value functions in Dec-POMDPs has been lacking. Given

the wide range of applications of value functions in single-agent decision-theoretic planning, we expect that such a theory for Dec-POMDPs can provide great benefits, both in terms of providing insight as well as guiding the design of solution algorithms.

A second contribution is the integrated overview of value-functions for decentralized settings. For the communicative settings already some results are known. In particular the immediate communication case reduces to the POMDP setting, for which many results are known, and Dec-MDP settings with delayed communication have received some attention in control theory literature. Yet, to the author's knowledge, there has never been an overview presenting these different settings in an integrated manner. Also, this chapter extends to the Dec-POMDP setting with delayed communication.

3.1 No Communication

This section shows how value functions can be defined for Dec-POMDPs of a finite horizon h. That is we assume that the agents cannot communicate or, equivalently, that the delay of communication is h. These value functions can be employed by representing a Dec-POMDP as a series of Bayesian Games, which were discussed in Subsection 2.1.1.2. This idea of using a series of BGs to find policies for a Dec-POMDP was proposed in an approximate setting by Emery-Montemerlo et al. (2004). In particular, they showed that using series of BGs and an approximate payoff function, they were able to obtain approximate solutions for Dec-POMDPs.

The main result of this section is that an *optimal* joint Dec-POMDP policy can be computed from the solution of a sequence of Bayesian games. That is, there exists at least one Q-value function $Q_{\boldsymbol{\pi}^*}$ (corresponding to an optimal policy $\boldsymbol{\pi}^*$) that, when used as the payoff function of a sequence of Bayesian games, will yield an optimal solution. Thus, the results of Emery-Montemerlo et al. (2004) are extended to include the optimal setting.

Subsection 3.1.4 shows that without assuming a particular optimal joint policy $\boldsymbol{\pi}^*$, it is not possible to compute an optimal value function of the form $Q^*(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$. Rather, as Subsection 3.1.5 demonstrates, the optimal $Q^{t,*}$ for a stage t depends on the past joint policy executed over stages 0,...,$t-1$.

3.1.1 Modeling Dec-POMDPs as Series of Bayesian Games

Bayesian games can be used to model Dec-POMDPs. Essentially, a Dec-POMDP can be seen as a tree where nodes are joint action-observation histories and edges represent joint actions and observations, as illustrated in Figure 3.1. At a specific stage t in a Dec-POMDP, the main difficulty in coordinating action selection is presented by the fact that each agent has its own individual action-observation history. That is, there is no global signal that the agents can use to coordinate their actions. This situation can be conveniently modeled by a Bayesian game as we will now discuss.

At a time step t, one can directly associate the primitives of a Dec-POMDP with those of a BG with identical payoffs: the actions of the agents are the same in

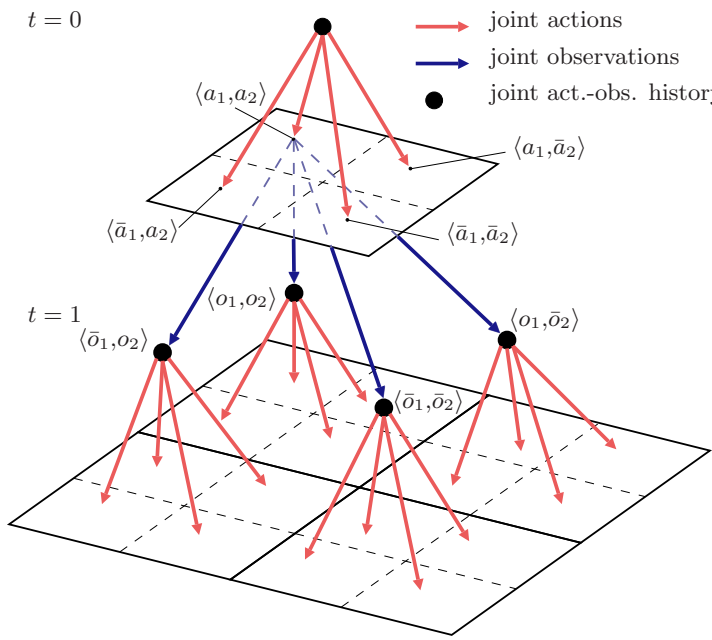


Figure 3.1: A Dec-POMDP can be seen as a tree of joint actions and observations. The indicated planes correspond with the Bayesian games for the first two stages.

both cases, and the types of agent i correspond to its action-observation histories $\Theta_i \equiv \vec{\Theta}_i^t$. Figure 3.2 shows the Bayesian games for $t = 0$ and $t = 1$ for a fictitious Dec-POMDP with 2 agents.

We denote the payoff function of the BG that models a stage of a Dec-POMDP by $Q(\vec{\boldsymbol{\theta}}^t, \mathbf{a})$. This payoff function should be naturally defined in accordance with the value function of the planning task. For instance, Emery-Montemerlo et al. define $Q(\vec{\boldsymbol{\theta}}^t, \mathbf{a})$ as the $\mathrm{Q}_{\mathrm{MDP}}$-value of the underlying MDP. We will more extensively discuss the payoff function in Subsection 3.1.2.

The probability $\Pr(\boldsymbol{\theta})$ is equal to the probability of the joint action-observation history $\vec{\boldsymbol{\theta}}^t$ to which $\boldsymbol{\theta}$ corresponds and depends on the past joint policy $\boldsymbol{\varphi}^t = (\boldsymbol{\delta}^0, \ldots, \boldsymbol{\delta}^{t-1})$ and the initial state distribution. It can be calculated as the marginal of (2.5.6):

$$\Pr(\boldsymbol{\theta}) = \Pr(\vec{\boldsymbol{\theta}}^t | \boldsymbol{\varphi}^t, \boldsymbol{b}^0) = \sum_{s^t \in \mathcal{S}} \Pr(s^t, \vec{\boldsymbol{\theta}}^t | \boldsymbol{\varphi}^t, \boldsymbol{b}^0). \tag{3.1.1}$$

When only considering pure joint policies $\boldsymbol{\varphi}^t$, the action probability component $\boldsymbol{\pi}(\mathbf{a}|\vec{\boldsymbol{\theta}})$ in (2.5.6) (which corresponds to $\boldsymbol{\varphi}(\mathbf{a}|\vec{\boldsymbol{\theta}})$ here) is 1 for joint action-observation histories $\vec{\boldsymbol{\theta}}^t$ that are 'consistent' with the past joint policy $\boldsymbol{\varphi}^t$ and 0 otherwise. We say that an action-observation $\vec{\theta}_i$ history is consistent with a pure policy π_i if it can occur when executing π_i, i.e., when the actions in $\vec{\theta}_i$ would be selected by π_i. Let us more formally define this consistency as follows.

Definition 3.1 (AOH Consistency)**.** Let us write $\vec{\theta}_i^k$ for the restriction of $\vec{\theta}_i^t$ to

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

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

Figure 3.2: The Bayesian game for the first and second time step (left: $t = 0$, right: $t = 1$). The entries $\vec{\boldsymbol{\theta}}^t$, $\mathbf{a}^t$ are given by the payoff function $Q(\vec{\boldsymbol{\theta}}^t,\mathbf{a}^t)$. Light shaded entries indicate the solutions. Dark entries will not be realized given $\langle a_1,a_2\rangle$ the solution of the BG for $t = 0$.

stages $0,\ldots,k$ (with $0 \leq k < t$). An action-observation history $\vec{\theta}_i^t$ of agent i is *consistent* with a pure policy π_i if and only if at each time step k with $0 \leq k < t$

$$\pi_i(\vec{\theta}_i^k) = \pi_i(\vec{o}_i^k) = a_i^k$$

is the $(k+1)$-th action in $\vec{\theta}_i^t$. A joint action-observation history $\vec{\boldsymbol{\theta}}^t = \langle\vec{\theta}_1^t,\ldots,\vec{\theta}_n^t\rangle$ is consistent with a pure joint policy $\boldsymbol{\pi} = \langle\pi_1,\ldots,\pi_n\rangle$ if each individual $\vec{\theta}_i^t$ is consistent with the corresponding individual policy π_i. C is the indicator function for consistency. For instance $C(\vec{\boldsymbol{\theta}}^t,\boldsymbol{\pi})$ 'filters out' the action-observation histories $\vec{\boldsymbol{\theta}}^t$ that are inconsistent with a joint pure policy $\boldsymbol{\pi}$:

$$C(\vec{\boldsymbol{\theta}}^t,\boldsymbol{\pi}) = \begin{cases} 1 & , \vec{\boldsymbol{\theta}}^t = \big(\mathbf{o}^0,\boldsymbol{\pi}(\mathbf{o}^0),\mathbf{o}^1,\boldsymbol{\pi}(\mathbf{o}^0,\mathbf{o}^1),\ldots\big)\,) \\ 0 & , \text{otherwise.} \end{cases} \tag{3.1.2}$$

We will also write $\vec{\boldsymbol{\Theta}}_{\boldsymbol{\pi}}^t \equiv \{\vec{\boldsymbol{\theta}}^t \mid C(\vec{\boldsymbol{\theta}}^t,\boldsymbol{\pi}) = 1\}$ for the set of $\vec{\boldsymbol{\theta}}^t$ consistent with $\boldsymbol{\pi}$.

This definition allows us to write

$$\Pr(\vec{\boldsymbol{\theta}}^t|\boldsymbol{\varphi}^t,\boldsymbol{b}^0) = C(\vec{\boldsymbol{\theta}}^t,\boldsymbol{\varphi}^t)\sum_{s^t\in\mathcal{S}}\Pr(s^t,\vec{\boldsymbol{\theta}}^t|\boldsymbol{b}^0) \tag{3.1.3}$$

with

$$\Pr(s^t,\vec{\boldsymbol{\theta}}^t|\boldsymbol{b}^0) = \sum_{s^{t-1}\in\mathcal{S}}\Pr(\mathbf{o}^t|\mathbf{a}^{t-1},s^t)\Pr(s^t|s^{t-1},\mathbf{a}^{t-1})\Pr(s^{t-1},\vec{\boldsymbol{\theta}}^{t-1}|\boldsymbol{b}^0). \tag{3.1.4}$$

Figure 3.2 illustrates how the indicator function 'filters out' action-observation histories: when $\boldsymbol{\pi}^{t=0}(\vec{\boldsymbol{\theta}}^{t=0}) = \langle a_1,a_2\rangle$, only the non-shaded part of the BG for $t = 1$ 'can be reached' (has positive probability).

3.1.2 The Q-Value Function of an Optimal Joint Policy

Given the perspective of a Dec-POMDP interpreted as a series of BGs as outlined in the previous section, the solution of the BG for stage t is a joint decision rule $\boldsymbol{\delta}^t$ for that stage. If the payoff function for the BG is chosen well, the quality of

$\boldsymbol{\delta}^t$ should be high. Emery-Montemerlo et al. (2004) try to find a good joint policy $\boldsymbol{\pi} = (\boldsymbol{\delta}^0, \ldots, \boldsymbol{\delta}^{h-1})$ by a procedure we refer to as *forward-sweep policy computation (FSPC)*: in one sweep forward through time, the BG for each stage $t = 0,1,\ldots,h-1$ is consecutively solved. As such, the payoff functions for the BGs constitute a Q-value function for the Dec-POMDP.

Here, we show that there exists an optimal Q-value function Q^*: when using this as the payoff functions for the BGs, forward-sweep policy computation will lead to an optimal joint policy $\boldsymbol{\pi}^* = (\boldsymbol{\delta}^{0,*}, \ldots, \boldsymbol{\delta}^{h-1,*})$. In particular, given an optimal joint policy $\boldsymbol{\pi}^*$ we identify a normative description of $Q_{\boldsymbol{\pi}^*}$ as the Q-value function for an optimal joint policy.

Proposition 3.1 (Value of an optimal joint policy.)**.** *The expected cumulative reward over stages $t,\ldots,h-1$ induced by $\boldsymbol{\pi}^*$, an optimal pure joint policy for a Dec-POMDP, is given by:*

$$V^t(\boldsymbol{\pi}^*) = \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\pi}^*}} \Pr(\vec{\boldsymbol{\theta}}^t | \boldsymbol{b}^0) Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^t, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^t)), \tag{3.1.5}$$

where $\vec{\boldsymbol{\theta}}^t = \langle \vec{\mathbf{o}}^t, \vec{\mathbf{a}}^t \rangle$, where $\boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^t) = \boldsymbol{\pi}^*(\vec{\mathbf{o}}^t)$ denotes the joint action $\boldsymbol{\pi}^*$ specifies for $\vec{\mathbf{o}}^t$, and where

$$Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^t, \mathbf{a}) = R(\vec{\boldsymbol{\theta}}^t, \mathbf{a}) + \sum_{\mathbf{o}^{t+1} \in \mathcal{O}} \Pr(\mathbf{o}^{t+1} | \vec{\boldsymbol{\theta}}^t, \mathbf{a}) Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^{t+1}, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{t+1})) \tag{3.1.6}$$

is the Q-value function for $\boldsymbol{\pi}^*$, which gives the expected cumulative future reward when taking joint action $\mathbf{a}$ at $\vec{\boldsymbol{\theta}}^t$ given that $\boldsymbol{\pi}^*$ is followed hereafter.

Sketch of proof. By filling out (2.5.5) for an optimal pure joint policy $\boldsymbol{\pi}^*$ it is possible to derive the equations. This derivation is listed in Appendix D. □

3.1.3 Deriving an Optimal Joint Policy

At this point we have derived $Q_{\boldsymbol{\pi}^*}$, a Q-value function for an optimal joint policy. Now, we extend the results of Emery-Montemerlo et al. (2004) into the exact setting by the following theorem:

Theorem 3.1 (Optimality of FSPC)**.** *Applying forward-sweep policy computation using $Q_{\boldsymbol{\pi}^*}$ as defined by* (3.1.6) *yields an optimal joint policy.*

Proof. Note that, per definition, an optimal Dec-POMDP policy $\boldsymbol{\pi}^*$ maximizes the expected future reward $V^t(\boldsymbol{\pi}^*)$ specified by (3.1.5). Therefore $\boldsymbol{\delta}^{t,*}$, the optimal decision rule for stage t, is identical to an optimal joint policy $\boldsymbol{\beta}^{t,*}$ for the Bayesian game for time step t, if the payoff function of the BG is given by $Q_{\boldsymbol{\pi}^*}$, that is:

$$\boldsymbol{\delta}^{t,*} \equiv \boldsymbol{\beta}^{t,*} = \arg\max_{\boldsymbol{\beta}^t} \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\pi}^*}} \Pr(\vec{\boldsymbol{\theta}}^t | \boldsymbol{b}^0) Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^t, \boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t)). \tag{3.1.7}$$

Equation (3.1.7) tells us that $\boldsymbol{\delta}^{t,*} \equiv \boldsymbol{\beta}^{t,*}$. This means that it is possible to construct the complete optimal Dec-POMDP policy $\boldsymbol{\pi}^* = (\boldsymbol{\delta}^{0,*}, \ldots, \boldsymbol{\delta}^{h-1,*})$, by computing $\boldsymbol{\delta}^{t,*}$ for all t. □

A subtlety in the calculation of $\boldsymbol{\pi}^*$ is that (3.1.7) itself is dependent on an optimal joint policy, as the summation is over all $\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\pi}^*} \equiv \{\vec{\boldsymbol{\theta}}^t \mid C(\vec{\boldsymbol{\theta}}^t,\boldsymbol{\pi}^*) = 1\}$. This is resolved by realizing that only the past actions influence which action-observation histories can be reached at time step t. If we denote the optimal past joint policy by $\boldsymbol{\varphi}^{t,*} = (\boldsymbol{\delta}^{0,*},\ldots,\boldsymbol{\delta}^{t-1,*})$, we have that $\vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\pi}^*} = \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\varphi}^{t,*}}$, and therefore that:

$$\boldsymbol{\beta}^{t,*} = \arg\max_{\boldsymbol{\beta}^t} \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\varphi}^{t,*}}} \Pr(\vec{\boldsymbol{\theta}}^t|\boldsymbol{b}^0) Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t)). \tag{3.1.8}$$

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

In this way, we have identified how a Q-value function of the form $Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$ can be used in the planning phase to find an optimal policy through solution of a series of BGs. Note that the fact that the true AOH, $\vec{\boldsymbol{\theta}}^t$, is not observed during execution is not a problem: the joint policy is constructed in the (off-line) planning phase, based on the expectation over all possible AOHs. This expectation is implicit in what can be called the 'BG operator', i.e., the solution of a BG.

3.1.4 Computing an Optimal Q-Value Function

So far we discussed that $Q_{\boldsymbol{\pi}^*}$ can be used to find an optimal joint policy $\boldsymbol{\pi}^*$. Unfortunately, when an optimal joint policy $\boldsymbol{\pi}^*$ is not known, computing $Q_{\boldsymbol{\pi}^*}$ itself is impractical, as we will discuss here. This is in contrast with the (fully observable) single-agent case where the optimal Q-values can be found relatively easily in a single sweep backward through time.

In Section 3.1.2, the optimal expected return was expressed as $Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$ *by assuming an optimal joint policy* $\boldsymbol{\pi}^*$ *is followed up to the current stage* t. However, when no such previous policy is assumed, the optimal expected return is not defined.

Proposition 3.2 (No $Q^*(\vec{\boldsymbol{\theta}}^t,\mathbf{a}^t)$ except for last stage)**.** *For a pair* $(\vec{\boldsymbol{\theta}}^t,\mathbf{a}^t)$ *with* $t < h-1$ *the optimal value* $Q^*(\vec{\boldsymbol{\theta}}^t,\mathbf{a}^t)$ *cannot be defined without assuming some (possibly randomized) past policy* $\boldsymbol{\varphi}^{t+1} = \big(\boldsymbol{\delta}^0,\ldots,\boldsymbol{\delta}^t\big)$*. Only for the last stage* $t = h-1$ *such expected reward is defined as*

$$Q^*(\vec{\boldsymbol{\theta}}^{h-1},\mathbf{a}^{h-1}) \;\equiv\; R(\vec{\boldsymbol{\theta}}^{h-1},\mathbf{a}^{h-1}) \tag{3.1.9}$$

without assuming a past policy.

Proof. Let us try to deduce $Q^*(\vec{\boldsymbol{\theta}}^t,\mathbf{a}^t)$ the optimal value for a particular $\vec{\boldsymbol{\theta}}^t$ assuming the Q^*-values for the next time step $t+1$ are known. The $Q^*(\vec{\boldsymbol{\theta}}^t,\mathbf{a}^t)$-values for each of the possible joint actions can be evaluated as follows

$$\forall_{\mathbf{a}} \quad Q^*(\vec{\boldsymbol{\theta}}^t,\mathbf{a}^t) = R(\vec{\boldsymbol{\theta}}^t,\mathbf{a}^t) + \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1}|\vec{\boldsymbol{\theta}}^t,\mathbf{a}^t) Q^*(\vec{\boldsymbol{\theta}}^{t+1},\boldsymbol{\delta}^{t+1,*}(\vec{\boldsymbol{\theta}}^{t+1})). \tag{3.1.10}$$

where $\boldsymbol{\delta}^{t+1,*}$ is an optimal decision rule for the next stage. But what should $\boldsymbol{\delta}^{t+1,*}$ be? If we assume that up to stage $t+1$ we followed a particular (possibly

randomized) past joint policy $\boldsymbol{\varphi}^{t+1}$, then

$$\boldsymbol{\delta}_{\boldsymbol{\varphi}}^{t+1,*} = \arg\max_{\boldsymbol{\beta}^{t+1}} \sum_{\vec{\boldsymbol{\theta}}^{t+1} \in \vec{\boldsymbol{\Theta}}^{t+1}} \Pr(\vec{\boldsymbol{\theta}}^{t+1} | \boldsymbol{\varphi}^{t+1}, \boldsymbol{b}^0) Q^*(\vec{\boldsymbol{\theta}}^{t+1}, \boldsymbol{\beta}^{t+1}(\vec{\boldsymbol{\theta}}^{t+1})). \tag{3.1.11}$$

is optimal. However, there are many pure and infinite randomized past policies $\boldsymbol{\varphi}^{t+1}$ that are consistent with $\vec{\boldsymbol{\theta}}^t, \mathbf{a}^t$, and thus many different $\vec{\boldsymbol{\Theta}}_{\boldsymbol{\varphi}}^{t+1}$ over which the above maximization could take place, leading to many $\boldsymbol{\delta}_{\boldsymbol{\varphi}}^{t+1,*}$ that might be optimal. The conclusion we can draw is that $Q^*(\vec{\boldsymbol{\theta}}^t, \mathbf{a}^t)$ is ill-defined without $\Pr(\vec{\boldsymbol{\theta}}^{t+1} | \boldsymbol{\varphi}^{t+1}, \boldsymbol{b}^0)$, the probability distribution (belief) over joint action-observation histories, which is induced by $\boldsymbol{\varphi}^{t+1}$, the policy followed for stages $0, \ldots, t$. □

Let us investigate what the consequences of this insight are for the formulation of the optimal Q-value function as defined in Section 3.1.2. Consider $\boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{t+1})$ in (3.1.6). This optimal policy is a mapping from observation histories to actions $\boldsymbol{\pi}^* : \vec{\mathcal{O}} \to \mathcal{A}$ induced by the individual policies and observation histories. This means that for two joint action-observation histories with the same joint observation history, $\boldsymbol{\pi}^*$ results in the same joint action. That is $\forall_{\vec{\mathbf{a}}, \vec{\mathbf{o}}, \vec{\mathbf{a}}'} \; \boldsymbol{\pi}^*(\langle \vec{\mathbf{a}}, \vec{\mathbf{o}} \rangle) = \boldsymbol{\pi}^*(\langle \vec{\mathbf{a}}', \vec{\mathbf{o}} \rangle)$. Effectively this means that when we reach some $\vec{\boldsymbol{\theta}}^t \notin \vec{\boldsymbol{\Theta}}_{\boldsymbol{\pi}^*}^t$, say through a mistake[1], $\boldsymbol{\pi}^*$ continues to specify actions as if no mistake ever happened: That is, *still* assuming that $\boldsymbol{\pi}^*$ has been followed up to this stage t. In fact, $\boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^t)$ *might not even be optimal* if $\vec{\boldsymbol{\theta}}^t \notin \vec{\boldsymbol{\Theta}}_{\boldsymbol{\pi}^*}^t$. This in turn means that $Q^*(\vec{\boldsymbol{\theta}}^{t-1}, \mathbf{a})$, the Q-values for predecessors of $\vec{\boldsymbol{\theta}}^t$, might not be the optimal.

3.1.5 Optimal Dec-POMDP Value Functions

We demonstrated that the optimal Q-value function for a Dec-POMDP is not well-defined without assuming a past joint policy. We propose a definition of the optimal value function of a Dec-POMDP that explicitly incorporates the past joint policy.

Theorem 3.2 (Optimal Q^*)**.** *The optimal Q-value function is properly defined as a function of the initial state distribution and joint past policies, action-observation histories and decision rules $Q^*(\boldsymbol{b}^0, \boldsymbol{\varphi}^t, \vec{\boldsymbol{\theta}}^t, \boldsymbol{\delta}^t)$. This Q^* specifies the optimal value given for all $\vec{\boldsymbol{\theta}}^t$, even for $\vec{\boldsymbol{\theta}}^t$ that are not reached by execution of an optimal joint policy $\boldsymbol{\pi}^*$.*

Proof. For all $\vec{\boldsymbol{\theta}}^t, \boldsymbol{\varphi}^t, \boldsymbol{\delta}^t$, the optimal expected return, respectively for the last stage and for all $0 \leq t < h-1$, is given by

$$Q^*(\boldsymbol{b}^0, \boldsymbol{\varphi}^{h-1}, \vec{\boldsymbol{\theta}}^{h-1}, \boldsymbol{\delta}^{h-1}) = R(\vec{\boldsymbol{\theta}}^{h-1}, \boldsymbol{\delta}^{h-1}(\vec{\boldsymbol{\theta}}^{h-1})), \tag{3.1.12}$$

$$Q^*(\boldsymbol{b}^0, \boldsymbol{\varphi}^t, \vec{\boldsymbol{\theta}}^t, \boldsymbol{\delta}^t) = R(\vec{\boldsymbol{\theta}}^t, \boldsymbol{\delta}^t(\vec{\boldsymbol{\theta}}^t)) + \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1} | \vec{\boldsymbol{\theta}}^t, \boldsymbol{\delta}^t(\vec{\boldsymbol{\theta}}^t)) Q^*(\boldsymbol{b}^0, \boldsymbol{\varphi}^{t+1}, \vec{\boldsymbol{\theta}}^{t+1}, \boldsymbol{\delta}^{t+1,*}), \tag{3.1.13}$$

[1] The question as to how the mistake of one agent should be detected by another agent is a different matter altogether and beyond the scope of this text.

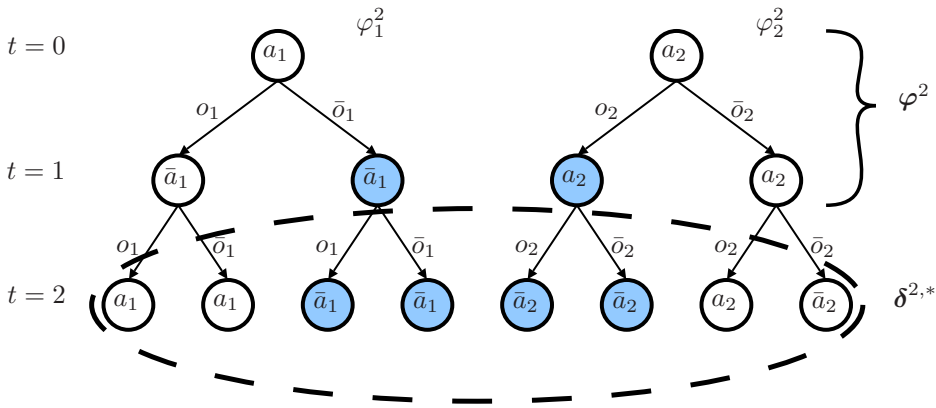


Figure 3.3: Computation of Q^*. $\boldsymbol{\delta}^{2,*}$ is the optimal decision rule for stage $t=2$, given that $\boldsymbol{\varphi}^2 = \langle \boldsymbol{\varphi}^1 \circ \boldsymbol{\delta}^1 \rangle$ is followed for the first two stages. $Q^*(\vec{\boldsymbol{\theta}}^1, \boldsymbol{\varphi}^1, \boldsymbol{\delta}^1)$ entries are computed by propagating relevant Q^*-values of the next stage. For instance, for the shaded joint history $\vec{\boldsymbol{\theta}}^1 = \langle (a_1, \bar{o}_1), (a_2, o_2) \rangle$, the Q^*-value under $\boldsymbol{\varphi}^2$ is computed by propagating the values of the four successor joint histories, as per (3.1.13).

with $\boldsymbol{\varphi}^{t+1} = \langle \boldsymbol{\varphi}^t \circ \boldsymbol{\delta}^t \rangle$ and

$$\boldsymbol{\delta}^{t+1,*} = \arg\max_{\boldsymbol{\delta}^{t+1}} \sum_{\vec{\boldsymbol{\theta}}^{t+1} \in \vec{\boldsymbol{\Theta}}^{t+1}} \Pr(\vec{\boldsymbol{\theta}}^{t+1} | \boldsymbol{b}^0, \boldsymbol{\varphi}^{t+1}) Q^*(\boldsymbol{b}^0, \boldsymbol{\varphi}^{t+1}, \vec{\boldsymbol{\theta}}^{t+1}, \boldsymbol{\delta}^{t+1}). \tag{3.1.14}$$

These definitions are consistent. Because of (3.1.12) for the last stage (3.1.14) will maximize the expected reward and thus is optimal. Equation (3.1.13) propagates these optimal value to the preceding stage. As such optimality for all stages follows by induction. □

The above equations constitute a dynamic program. When assuming that only pure joint past policies $\boldsymbol{\varphi}$ can be used, the dynamic program can be evaluated from the end ($t = h-1$) to the beginning ($t = 0$). Figure 3.3 illustrates the computation of Q^*.

When arriving at stage 0, the past joint policy is empty $\boldsymbol{\varphi}^0 = ()$ and joint decision rules are simply joint actions, thus it is possible to select

$$\boldsymbol{\delta}^{0,*} = \arg\max_{\boldsymbol{\delta}^0} Q^*(\boldsymbol{b}^0, \boldsymbol{\varphi}^0, \vec{\boldsymbol{\theta}}_\emptyset, \boldsymbol{\delta}^0) = \arg\max_{\mathbf{a}} Q^*(\boldsymbol{b}^0, (), \vec{\boldsymbol{\theta}}_\emptyset, \mathbf{a}). \tag{3.1.15}$$

Then given $\boldsymbol{\varphi}^1 = \boldsymbol{\delta}^{0,*}$ we can determine $\boldsymbol{\delta}^{1,*}$ using (3.1.14), etc. This essentially is the forward-sweep policy computation using optimal Q-value function as defined by (3.1.13). Note that the solution of the Bayesian game (i.e., performing the maximization in (3.1.14)) has already been done and can be cached.

At this point, there is no clear understanding of $Q^*(\boldsymbol{b}^0, \boldsymbol{\varphi}^t, \vec{\boldsymbol{\theta}}^t, \boldsymbol{\delta}^t)$ as an action-value function anymore, i.e., it is no longer coupled to domain level (joint) actions. Of course it is possible to recover these. The value of performing $\mathbf{a} = \boldsymbol{\delta}^t(\vec{\boldsymbol{\theta}}^t)$ from

$\vec{\boldsymbol{\theta}}^t$ after having followed $\boldsymbol{\varphi}^{t+1} = \boldsymbol{\varphi}^t,\boldsymbol{\delta}^t$ and continuing optimally afterward is given by

$$Q_{\boldsymbol{\varphi}^{t+1}=\langle\boldsymbol{\varphi}^t\circ\boldsymbol{\delta}^t\rangle}(\vec{\boldsymbol{\theta}}^t,\boldsymbol{\delta}^t(\vec{\boldsymbol{\theta}}^t)) = Q^*(\boldsymbol{b}^0,\boldsymbol{\varphi}^t,\vec{\boldsymbol{\theta}}^t,\boldsymbol{\delta}^t). \tag{3.1.16}$$

This can be used for instance to compute $Q_{\boldsymbol{\pi}^*}$.

Another emerging question is why we still refer to a Q-value function, i.e., why we use the symbol 'Q' to refer to the value function defined here. The answer is that $\boldsymbol{\delta}^t$ can be seen as an action on the meta-level of the planning process. In the same way we can interpret $\boldsymbol{\varphi}^t$ as the state in this planning process and we can define V and Q with their usual interpretations. In particular, it is possible to write

$$V^*(\boldsymbol{b}^0,\boldsymbol{\varphi}^t) = \max_{\boldsymbol{\delta}^t} Q^*(\boldsymbol{b}^0,\boldsymbol{\varphi}^t,\boldsymbol{\delta}^t) \tag{3.1.17}$$

where Q^* is defined as

$$Q^*(\boldsymbol{b}^0,\boldsymbol{\varphi}^t,\boldsymbol{\delta}^t) = \sum_{\vec{\boldsymbol{\theta}}^t} \Pr(\vec{\boldsymbol{\theta}}^t|\boldsymbol{b}^0,\boldsymbol{\varphi}^t) Q^*(\boldsymbol{b}^0,\boldsymbol{\varphi}^t,\vec{\boldsymbol{\theta}}^t,\boldsymbol{\delta}^t). \tag{3.1.18}$$

Note that this Q^* indeed has the regular interpretation of the expected immediate reward induced by first taking 'action' $\boldsymbol{\delta}^t$ plus the cumulative reward of continuing optimally afterward. We can see this by rewriting

$$\begin{aligned}
&Q^*(\boldsymbol{b}^0,\boldsymbol{\varphi}^t,\boldsymbol{\delta}^t)\\
&= \sum_{\vec{\boldsymbol{\theta}}^t} \Pr(\vec{\boldsymbol{\theta}}^t|\boldsymbol{b}^0,\boldsymbol{\varphi}^t)\\
&\quad \Big[R(\vec{\boldsymbol{\theta}}^t,\boldsymbol{\delta}^t(\vec{\boldsymbol{\theta}}^t)) + \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1}|\vec{\boldsymbol{\theta}}^t,\boldsymbol{\delta}^t(\vec{\boldsymbol{\theta}}^t)) Q^*(\boldsymbol{b}^0,\langle\boldsymbol{\varphi}^t\circ\boldsymbol{\delta}^t\rangle,\vec{\boldsymbol{\theta}}^{t+1},\boldsymbol{\delta}^{t+1,*})\Big]\\
&= \sum_{\vec{\boldsymbol{\theta}}^t} \Pr(\vec{\boldsymbol{\theta}}^t|\boldsymbol{b}^0,\boldsymbol{\varphi}^t) R(\vec{\boldsymbol{\theta}}^t,\boldsymbol{\delta}^t(\vec{\boldsymbol{\theta}}^t))\\
&\quad + \sum_{\vec{\boldsymbol{\theta}}^t} \Pr(\vec{\boldsymbol{\theta}}^t|\boldsymbol{b}^0,\boldsymbol{\varphi}^t) \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1}|\vec{\boldsymbol{\theta}}^t,\boldsymbol{\delta}^t(\vec{\boldsymbol{\theta}}^t)) Q^*(\boldsymbol{b}^0,\langle\boldsymbol{\varphi}^t\circ\boldsymbol{\delta}^t\rangle,\vec{\boldsymbol{\theta}}^{t+1},\boldsymbol{\delta}^{t+1,*})\\
&= E\big[R(s^t,\mathbf{a}^t) \;\big|\; \boldsymbol{b}^0,\boldsymbol{\varphi}^t,\boldsymbol{\delta}^t\big] + \sum_{\vec{\boldsymbol{\theta}}^{t+1}} \Pr(\vec{\boldsymbol{\theta}}^{t+1}|\boldsymbol{b}^0,\langle\boldsymbol{\varphi}^t\circ\boldsymbol{\delta}^t\rangle) Q^*(\boldsymbol{b}^0,\langle\boldsymbol{\varphi}^t\circ\boldsymbol{\delta}^t\rangle,\vec{\boldsymbol{\theta}}^{t+1},\boldsymbol{\delta}^{t+1,*})\\
&= E\big[R(s^t,\mathbf{a}^t) \;\big|\; \boldsymbol{b}^0,\boldsymbol{\varphi}^t,\boldsymbol{\delta}^t\big] + Q^*(\boldsymbol{b}^0,\langle\boldsymbol{\varphi}^t\circ\boldsymbol{\delta}^t\rangle,\boldsymbol{\delta}^{t+1,*})\\
&= E\big[R(s^t,\mathbf{a}^t) \;\big|\; \boldsymbol{b}^0,\boldsymbol{\varphi}^t,\boldsymbol{\delta}^t\big] + \max_{\boldsymbol{\delta}^{t+1}} Q^*(\boldsymbol{b}^0,\langle\boldsymbol{\varphi}^t\circ\boldsymbol{\delta}^t\rangle,\boldsymbol{\delta}^{t+1})\\
&= E\big[R(s^t,\mathbf{a}^t) \;\big|\; \boldsymbol{b}^0,\boldsymbol{\varphi}^t,\boldsymbol{\delta}^t\big] + V^*(\boldsymbol{b}^0,\langle\boldsymbol{\varphi}^t\circ\boldsymbol{\delta}^t\rangle).
\end{aligned} \tag{3.1.19}$$

Here $E\big[R(s^t,\mathbf{a}^t) \;\big|\; \boldsymbol{b}^0,\boldsymbol{\varphi}^t,\boldsymbol{\delta}^t\big]$ corresponds to the expected immediate reward at stage t and $V^*(\boldsymbol{b}^0,\langle\boldsymbol{\varphi}^t\circ\boldsymbol{\delta}^t\rangle)$ to the optimal value of continuing afterward.

3.1.5.1 The Relation to (Point-Based) Dynamic Programming

The computation of Q^* is closely related to (point-based) dynamic programming for Dec-POMDPs as discussed in Subsection 2.6.5 and 2.6.5.1. Suppose that $t = 2$

in Figure 3.3 is the last stage (i.e., $h = 3$). When for each $\boldsymbol{\varphi}^2 = \langle \boldsymbol{\varphi}^1 \circ \boldsymbol{\delta}^1 \rangle$ the maximizing $\boldsymbol{\delta}^{2,*} = \langle \delta_1^{2,*}, \ldots, \delta_n^{2,*} \rangle$ has been computed, it is easy to construct the sets of non-dominated actions for each agent: every action a_i of agent i that is specified by some $\delta_i^{2,*}$ is non-dominated.

Once we have computed the values for all $(\vec{\boldsymbol{\theta}}^1, \boldsymbol{\varphi}^2 = \langle \boldsymbol{\varphi}^1 \circ \boldsymbol{\delta}^1 \rangle)$ at $t = 1$ (i.e., all $Q^*(\boldsymbol{b}^0, \boldsymbol{\varphi}^1, \vec{\boldsymbol{\theta}}^1, \boldsymbol{\delta}^1)$ are computed), each $\boldsymbol{\varphi}^2$ has an associated next-stage joint decision rule $\boldsymbol{\delta}^{2,*}$, and in general an optimal joint future policy $\boldsymbol{\psi}^1 = (\boldsymbol{\delta}^{2,*}, \ldots, \boldsymbol{\delta}^{h-1,*})$ [1].

This means that, given $\boldsymbol{\varphi}^1, \boldsymbol{\delta}^1$, we can define a joint sub-tree policy $\boldsymbol{q}^{\tau=2}$ for each $\vec{\boldsymbol{\theta}}^1$.[2] This is done by taking $\langle \boldsymbol{\delta}^1 \circ \boldsymbol{\psi}^1 \rangle$ and restricting this joint policy to the histories consistent with $\vec{\boldsymbol{\theta}}^1$. For instance, in Figure 3.3 the shaded trees represent the joint sub-tree policy for $\vec{\boldsymbol{\theta}}^1$ given the indicated past policy $\boldsymbol{\varphi}^2 = \langle \boldsymbol{\varphi}^1 \circ \boldsymbol{\delta}^1 \rangle$. Clearly, $Q^*(\boldsymbol{b}^0, \boldsymbol{\varphi}^1, \vec{\boldsymbol{\theta}}^1, \boldsymbol{\delta}^1)$ corresponds to the expected value of this associated joint sub-tree policy. Dynamic programming keeps track of these sub-trees policies. In contrast, the algorithm to compute Q^* presented in this section keeps track of the values.

3.1.5.2 The Relation to Sequential Rationality

The fact that it is not possible to simply define $Q^*(\vec{\boldsymbol{\theta}}^t, \mathbf{a}^t)$ is very much related to the notion of *sub-game perfect equilibria* from game theory. As explained in Section (2.2), a sub-game perfect Nash equilibrium $\boldsymbol{\pi} = \langle \pi_1, \ldots, \pi_n \rangle$ has the characteristic that the contained policies π_i specify an optimal action for *all* possible situations—even situations that can not occur when following $\boldsymbol{\pi}$. A commonly given rationale behind this concept is that, by a mistake of one of the agents during execution, situations that should not occur according to $\boldsymbol{\pi}$, may occur, and also in these situations the agents should act optimally. A different rationale is given by Binmore (1992), who remarks that although it is tempting to ignore situations that should not occur according to $\boldsymbol{\pi}$, it would clearly be a mistake, because the agents "remain on the equilibrium path because of what they anticipate *would* happen if they *were* to deviate". This implies that agents can decide upon a Nash equilibrium by analyzing what the expected outcome would be by following other policies: that is, when acting optimally from other situations. In the previous section, we performed a similar reasoning for Dec-POMDPs, which—in a similar fashion—resulted in a description that allows to deduce an optimal Q-value function and thus joint policy.

A Dec-POMDP can be modeled as an extensive form game of imperfect information (Oliehoek and Vlassis, 2006). For such games, the notion of sub-game perfect equilibria is inadequate; because this type of games often do not contain proper sub-games, every Nash equilibrium is trivially sub-game perfect. (The extensive form of a Dec-POMDP indeed does not contain proper sub-games, because agents can never discriminate between the other agents' observations.) To overcome this problem different refinements of the Nash equilibrium concept have been defined, of which we will mention the *assessment equilibrium* (Binmore, 1992) and the closely

[1] In this example $\boldsymbol{\psi}^1 = \boldsymbol{\delta}^{2,*}$ because $h = 3$

[2] Remember a τ-stages-to-go sub-tree policy is rooted at stage $t = h - \tau$, so $\boldsymbol{q}^{\tau=2}$ starts at stage $t = 3 - 2 = 1$.

related, but stronger *sequential equilibrium* (Osborne and Rubinstein, 1994). Both these equilibria are based on the concept of an assessment, which is a pair $\langle \boldsymbol{\pi}, B \rangle$ consisting of a joint policy $\boldsymbol{\pi}$ and a *belief system* B. The belief system maps the information sets of each agent—also the ones that are not reachable given $\boldsymbol{\pi}$—to a probability distributions over nodes and thus possible joint histories. Roughly speaking, an assessment equilibrium requires *sequential rationality* and *belief consistency*.[1] The former entails that the joint policy $\boldsymbol{\pi}$ specifies optimal actions for each information set given B. Belief consistency means that all the beliefs that are assigned by B are Bayes-rational given the specified joint policy $\boldsymbol{\pi}$, i.e., the beliefs are computed through proper application of Bayes' rule.[2]

This is in direct correspondence to (3.1.14), which specifies to follow a strategy that is rational given the probabilities of all possible histories $\vec{\boldsymbol{\theta}}$, and those probabilities (the corresponding consistent belief system) are computed correctly. As such we also refer to Q^* as defined in Theorem 3.2 as the 'sequentially rational' Q-value function, in contrast to the normative description $Q_{\boldsymbol{\pi}^*}$ of (3.1.6). Also note that using Q^* the optimal future policy can be computed for *any* past policy. This may have important applications in an online setting. For instance, suppose agent i makes a mistake at stage t, executing an action not prescribed by π_i^*, assuming the other agents execute their policy $\pi_{\neq i}$ without mistakes, agent i knows the actually executed previous policy $\boldsymbol{\varphi}^{t+1}$. Therefore it can compute a new individual policy by

$$\delta_i^{t+1,*} = \arg\max_{\delta_i^{t+1}} \sum_{\vec{\boldsymbol{\theta}}^{t+1}} \Pr(\vec{\boldsymbol{\theta}}^{t+1} | \boldsymbol{b}^0, \boldsymbol{\varphi}^{t+1}) Q^*(\vec{\boldsymbol{\theta}}^{t+1}, \boldsymbol{\varphi}^{t+1}, \langle \delta_i^{t+1}, \delta_{\neq i}^{t+1} \rangle). \tag{3.1.20}$$

3.1.5.3 The Complexity of Computing Q^*

Although we have now found a way to compute the optimal Q-value function, this computation is intractable for all but the smallest problems. The last stage is trivial, since the Q-values are given directly by the immediate reward function. However, for stage $t = h - 2$ the Q-value function $Q^*(\boldsymbol{b}^0, \boldsymbol{\varphi}^t, \vec{\boldsymbol{\theta}}^t, \boldsymbol{\delta}^t)$ has a huge number of entries: effectively we need to compute an entry for each combination of $\boldsymbol{\varphi}^{h-1} = \langle \boldsymbol{\varphi}^{h-2} \circ \boldsymbol{\delta}^{h-2} \rangle$ and each consistent joint action-observation history $\vec{\boldsymbol{\theta}}^{h-2}$.

Up to and including an arbitrary stage $t-1$, there are $\sum_{t'=0}^{t-1} |\mathcal{O}_i|^{t'} = \frac{|\mathcal{O}_i|^t - 1}{|\mathcal{O}_i| - 1}$ observation histories for agent i and thus the number of $\boldsymbol{\varphi}^t$ is

$$O\left(|\mathcal{A}_*|^{\frac{n(|\mathcal{O}_*|^t - 1)}{|\mathcal{O}_*| - 1}} \right).$$

For each of these past joint policies $\boldsymbol{\varphi}^t$ there are $|\vec{\mathcal{O}}^t| = |\mathcal{O}|^t$ consistent joint action-observation histories at stage t (for each observation history $\vec{\boldsymbol{o}}^t$, $\boldsymbol{\varphi}^t$ specifies the actions forming $\vec{\boldsymbol{\theta}}^t$). This means that for stage $h-2$ (for $h-1$, the Q-values are

[1] Osborne and Rubinstein (1994) refer to this second requirement as simply 'consistency'. In order to avoid any confusion with definition 3.1 we will use the term 'belief consistency'.

[2] A sequential equilibrium includes a more technical part in the definition of belief consistency that addresses what beliefs should be held for information sets that are not reached according to $\boldsymbol{\pi}$. For more information we refer to Osborne and Rubinstein (1994).

easily calculated), the number of entries to be computed is the number of joint past policies $\boldsymbol{\varphi}^{h-1}$ times the number of joint histories

$$O\left(|\mathcal{A}_*|^{\frac{n\left(|\mathcal{O}_*|^{h-1}-1\right)}{|\mathcal{O}_*|-1}} \cdot |\mathcal{O}|^{h-2}\right),$$

indicating that computation of this function is doubly exponential in the horizon. Also, for each joint past policy $\boldsymbol{\varphi}^{h-2}$,$\boldsymbol{\delta}^{h-2}$, we need to compute $\boldsymbol{\delta}^{h-1,*}$ by solving a BG for the last stage. To the author's knowledge, the only method to optimally solve these BGs is evaluation of all $O(|\mathcal{A}_*|^{n|\mathcal{O}_*|^{h-1}})$ joint BG-policies.

As such, computing optimal value functions is intractable. In the remainder of this chapter we will treat value functions for Dec-POMDPs with communication and show that these are easier to compute. In the next chapter, we will consider using approximate value functions for the non-communicative setting.

3.2 Instantaneous Communication

This section describes the optimal value functions for the setting in which the agents are capable of instantaneous, noise-free and cost-free communication. In particular Pynadath and Tambe (2002b) showed that under such circumstances, it is optimal for the agents to share their local observations (e.g., by broadcasting the individual observations to all other agents).

In effect such communication transforms a Dec-POMDP to what we refer to as a *multiagent POMDP (MPOMDP)*.[1] We can think of the MPOMDP as a POMDP in which there is a puppeteer agent that takes joint actions and receives joint observations. By solving this underlying POMDP of the Dec-POMDP, all POMDP-literature applies to this setting.

In particular, the optimal 0-steps delay value function V_0 for an underlying POMDP satisfies:

$$V_0^*(\boldsymbol{b}^t) = \max_{\mathbf{a}} Q_0^*(\boldsymbol{b}^t,\mathbf{a}) \tag{3.2.1}$$

$$Q_0^*(\boldsymbol{b}^t,\mathbf{a}) = R(\boldsymbol{b}^t,\mathbf{a}) + \sum_{\mathbf{o}^{t+1}\in\mathcal{O}} P(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\mathbf{a}) \max_{\mathbf{a}^{t+1}} Q_0^*(\boldsymbol{b}^{t+1},\mathbf{a}^{t+1}), \tag{3.2.2}$$

where $\boldsymbol{b}^t$ is the *joint belief* (corresponding to some joint AOH $\vec{\boldsymbol{\theta}}^t$) of the single puppeteer agent that selects joint actions and receives joint observations at time step t, where

$$R(\boldsymbol{b}^t,\mathbf{a}) = \sum_{s\in\mathcal{S}} R(s,\mathbf{a})\boldsymbol{b}^t(s) \tag{3.2.3}$$

is the expected immediate reward, and where $\boldsymbol{b}^{t+1}$ is the joint belief resulting from $\boldsymbol{b}^t$ by action $\mathbf{a}$ and joint observation $\mathbf{o}^{t+1}$, calculated by Bayes' rule:

$$\forall_{s'} \quad \boldsymbol{b}^{t+1}(s') = \frac{\Pr(\mathbf{o}|\mathbf{a},s')\sum_{s\in\mathcal{S}}\Pr(s'|s,\mathbf{a})\boldsymbol{b}^t(s)}{\sum_{s'\in\mathcal{S}}\Pr(\mathbf{o}|\mathbf{a},s')\sum_{s\in\mathcal{S}}\Pr(s'|s,\mathbf{a})\boldsymbol{b}^t(s)}. \tag{3.2.4}$$

[1] This name is chosen in analogy with the multiagent MDP, described in Subsection 2.8.1.

$\vec{\theta}_1^{t=0}$	$\vec{\theta}_2^{t=0}$	()	
		a_2	$\bar{a}_2$
()	a_1	+3.1	−4.1
	$\bar{a}_1$	−0.9	+0.3

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

Figure 3.4: Backward calculation of Q_0^*-values. Note that the solutions (the highlighted entries) are different from those in Figure 3.2: in a MPOMDP the joint actions can be conditioned on the joint action-observation history. The highlighted '+3.1' entry for the Bayesian game for $t = 0$ is calculated as the expected immediate reward (= 0) plus a weighted sum of the maximizing entry (joint action) per next joint observation history. When assuming a uniform distribution over joint observations given $\langle a_1,a_2\rangle$ the future reward is given by: $+3.1 = 0 + 0.25 \cdot 2.0 + 0.25 \cdot 4.0 + 0.25 \cdot 4.4 + 0.25 \cdot 2.0$.

For a finite horizon, Q_0^* can be computed by generating all possible joint beliefs and solving the *belief MDP*. Generating all possible beliefs is easy: starting with $\boldsymbol{b}^0$ corresponding to the empty joint action-observation history $\vec{\boldsymbol{\theta}}^{t=0}$, for each $\mathbf{a}$ and $\mathbf{o}$ we calculate the resulting $\vec{\boldsymbol{\theta}}^{t=1}$ and corresponding joint belief and continue recursively. Solving the belief MDP amounts to recursively applying (3.2.2).

The cost of computing the optimal MPOMDP Q-value function can be divided in the cost of calculating the expected immediate reward for all $\vec{\boldsymbol{\theta}}^t$,$\mathbf{a}$, and the cost of evaluating future reward for all $\vec{\boldsymbol{\theta}}^t$,$\mathbf{a}$, with $t = 0,...,h-2$. The former operation has cost $O(|\mathcal{S}|)$ per $(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$-pair. The latter requires selecting the maximizing joint action for each joint observation which induces a cost of $(|\mathcal{A}|\,|\mathcal{O}|)$ per $(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$-pair. The total complexity of computing Q_0^* becomes

$$O\left(\frac{(|\mathcal{A}|\,|\mathcal{O}|)^{h-1}-1}{(|\mathcal{A}|\,|\mathcal{O}|)-1}|\mathcal{A}|\,(|\mathcal{A}|\,|\mathcal{O}|) + \frac{(|\mathcal{A}|\,|\mathcal{O}|)^{h}-1}{(|\mathcal{A}|\,|\mathcal{O}|)-1}|\mathcal{A}||\mathcal{S}|\right). \tag{3.2.5}$$

Evaluating (3.2.2) for (joint beliefs for) all joint action-observation histories $\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t$ can be done in a single backward sweep through time. This can also be visualized in Bayesian games as illustrated in Figure 3.4; the expected future reward is calculated as a maximizing weighted sum of the entries of the next time step BG.[1]

Nevertheless, solving a POMDP optimally is also known as an intractable problem. As a result, POMDP research in the last decade has focused on approximate solutions for POMDPs. In particular, it is known that the value function of a POMDP is *piecewise-linear and convex (PWLC)* over the (joint) belief space (Sondik, 1971). This property is exploited by many approximate POMDP solution methods (Pineau, Gordon, and Thrun, 2003; Spaan and Vlassis, 2005).

[1]Note that the maximization in (3.2.2) can be seen as a special instance of the 'BG-operator' expressed by (3.1.14). Because, under instantaneous communication, it will be possible to prune all histories that are not realized from the BG, the BG-operator reduces to a simple maximization.

3.3 One-Step Delayed Communication

Here we describe the optimal value functions for a Dec-POMDP with noise-free and cost-free communication that arrives with a one-step delay (1-SD). I.e., the assumption is that during execution at stage t the agents know $\vec{\boldsymbol{\theta}}^{t-1}$, the joint action-observation history up to time step $t-1$, and the joint action $\mathbf{a}^{t-1}$ that was taken at the previous time step. Because all the agents know $\vec{\boldsymbol{\theta}}^{t-1}$, they can compute the joint belief $\boldsymbol{b}^{t-1}$ it induces which is a Markovian signal. Therefore the agents do not need to maintain prior information, $\boldsymbol{b}^{t-1}$ takes the same role as $\boldsymbol{b}^0$ in a regular Dec-POMDP (i.e., without communication).

The 1-SD-setting has also been considered in the field of decentralized control, where it is usually referred to as decentralized control with a "one-step delayed sharing pattern". In particular, Varaiya and Walrand (1978) showed that in this setting state estimation and control are *separable.* I.e., there exist an optimal *separable* joint policy, that specifies a separable individual policy for each agent that maps from $\boldsymbol{b}^{t-1}$ and the *individual* history of observations received since $\boldsymbol{b}^{t-1}$, to actions. Hsu and Marcus (1982) extended this work by deriving dynamic programming algorithms for the finite- and infinite-horizon case. Their approach, however, is based on the one-step predictor of s^t, leading to a rather involved formulation. In this section we present what we believe to be a conceptually clearer formulation, by resorting to Bayesian games.

As mentioned, in the 1-SD setting the agents know $\vec{\boldsymbol{\theta}}^{t-1}$ and thus can compute $\boldsymbol{b}^{t-1}$. Also, since we assume that during execution each agent knows the joint policy, each agent can defer the taken joint action $\mathbf{a}^{t-1}$. However, the agents are uncertain regarding each other's last observation, and thus regarding the joint observation $\mathbf{o}^t$. Effectively, this situation defines a BG for each possible joint belief $\boldsymbol{b}^{t-1}$ (induced by all possible $\vec{\boldsymbol{\theta}}^{t-1}$) and joint action $\mathbf{a}^{t-1}$. Note, however, that these BGs are different from the BGs used in Section 3.1.1: the BGs here have types that correspond to single observations, whereas the BGs in 3.1.1 have types that correspond to complete action-observation histories. Hence, the BGs of here are much smaller in size and thus easier to solve.

Lemma 3.1 (Value of one-step delayed communication)**.** *The optimal value function for a finite-horizon Dec-POMDP with one-step delayed communication is given by*

$$V_1^{t,*}(\boldsymbol{b}^{t-1},\mathbf{a}^{t-1}) = \max_{\boldsymbol{\beta}^t} Q_1^{t,*}(\boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\boldsymbol{\beta}^t) \tag{3.3.1}$$

$$Q_1^{t,*}(\boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\boldsymbol{\beta}^t) = \sum_{\mathbf{o}^t} \Pr(\mathbf{o}^t|\boldsymbol{b}^{t-1},\mathbf{a}^{t-1}) Q_1^{t,*}(\boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\mathbf{o}^t,\boldsymbol{\beta}^t) \tag{3.3.2}$$

$$\begin{aligned} Q_1^{t,*}(\boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\mathbf{o}^t,\boldsymbol{\beta}^t) &= R(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\mathbf{o}^t)) \\ &\quad + \max_{\boldsymbol{\beta}^{t+1}} \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{\beta}^t(\mathbf{o}^t)) Q_1^{t+1,*}(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\mathbf{o}^t),\mathbf{o}^{t+1},\boldsymbol{\beta}^{t+1}), \end{aligned} \tag{3.3.3}$$

where $\boldsymbol{b}^t$ *results from* $\boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\mathbf{o}^t$.

Proof. $Q_1^{t,*}(\boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\boldsymbol{\beta}^t)$ should be defined as the sum of the expected immediate and future reward, so (3.3.2) should equal

$$E\left[R(s^t,\mathbf{a}^t) \mid \boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\boldsymbol{\beta}^t\right] + E\left[V_1^{t+1,*}(\boldsymbol{b}^t,\mathbf{a}^t) \mid \boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\boldsymbol{\beta}^t\right] = \sum_{\mathbf{o}^t} \Pr(\mathbf{o}^t|\boldsymbol{b}^{t-1},\mathbf{a}^{t-1}) R(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\mathbf{o}^t)) + \sum_{\mathbf{o}^t} \Pr(\mathbf{o}^t|\boldsymbol{b}^{t-1},\mathbf{a}^{t-1}) V_1^{t+1,*}(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t)) \quad (3.3.4)$$

substitution of $V_1^{t+1,*}$ yields

$$\sum_{\mathbf{o}^t} \Pr(\mathbf{o}^t|\boldsymbol{b}^{t-1},\mathbf{a}^{t-1}) \Big[R(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\mathbf{o}^t)) + \max_{\boldsymbol{\beta}^{t+1}} \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{\beta}^t(\mathbf{o}^t)) Q_1^{t+1,*}(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\mathbf{o}^t),\mathbf{o}^{t+1},\boldsymbol{\beta}^{t+1}) \Big] \quad (3.3.5)$$

which shows that the definitions are consistent. For the last stage $h-1$, the maximization in (3.3.1) assures that the future reward is maximized, as such (3.3.1) is optimal for the last stage. Given the optimality of the last stage, optimality for previous stages follows by induction: given the optimality of $V_1^{h-1,*}$, $V_1^{h-2,*}$ maximizes the sum of the expected immediate reward at $h-2$ plus the future reward $V_1^{h-1,*}$, etc. □

There is a clear correspondence between the equations in Lemma 3.1 and those in Section 3.1.5. In particular, (3.3.1) corresponds to (3.1.17), because $\mathbf{a}^{t-1}$ (resp. $\boldsymbol{\varphi}^t$) is the joint policy taken since the last known Markovian signal (distribution over states) $\boldsymbol{b}^{t-1}$ (resp. $\boldsymbol{b}^0$). Similarly, (3.3.2) corresponds to (3.1.18) where $\boldsymbol{\beta}^t$ (resp. $\boldsymbol{\delta}^t$) is the joint policy for stage t that implicitly maps sequences of joint observations $\mathbf{o}^t$ (resp. $\vec{\boldsymbol{\theta}}^t$) received since the last Markov signal to actions $\mathbf{a}^t$.

3.3.1 Immediate Reward Formulation

Note that the arguments of (3.3.3) are somewhat redundant. In particular, the right side only depends on the joint belief $\boldsymbol{b}^t$ that results from $\boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\mathbf{o}^t$ and on the joint action $\mathbf{a}^t = \boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t)$. As such it can be rewritten simpler as

$$V_1^{t,*}(\boldsymbol{b}^t,\mathbf{a}^t) = R(\boldsymbol{b}^t,\mathbf{a}^t) + \max_{\boldsymbol{\beta}^{t+1}} \sum_{\mathbf{o}^{t+1}\in\mathcal{O}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\mathbf{a}^t) V_1^{t+1,*}(\boldsymbol{b}^{t+1},\boldsymbol{\beta}^{t+1}(\mathbf{o}^{t+1})) \quad (3.3.6)$$

In doing so, we have now constructed a formulation that specifies the value over stages $t,\ldots,h-1$ using arguments of stage t. We refer to this formulation as an *immediate reward value function*. This should be seen in contrast to formulas of the form of (3.3.1) that specify the expected value over stages $t,\ldots,h-1$, but using arguments (in particular a Markovian signal $\boldsymbol{b}^{t-1}$) of stage $t-1$ (or in general $t-k$). We refer to this type as *expected reward value function*, since they specify the expected value over later stages given arguments of an earlier stage.

The remainder of this chapter will use the expected reward formulation, but occasionally we will make a remark about the other immediate reward formulation,

as done here. We also note that in (3.3.6) we use V_1 and not Q_1 because this is consistent with the notation for immediate reward value function formulations. An overview of immediate reward value functions and the relation to expected reward formulations is given in Appendix B.

3.3.2 Complexity

The cost of computing $V_1^{t,*}(\boldsymbol{b}^t,\mathbf{a}^t)$ for all $\vec{\boldsymbol{\theta}}^t$,$\mathbf{a}$ can be split up in the cost of computing the immediate reward and the cost of computing the future reward (solving a BG over the last received observation), which is $O\big(\,|\mathcal{A}_*|^{n|\mathcal{O}_*|}\big)$, leading to a total cost of:

$$O\left(\frac{(|\mathcal{A}|\,|\mathcal{O}|)^{h-1}-1}{(|\mathcal{A}|\,|\mathcal{O}|)-1}|\mathcal{A}|\cdot|\mathcal{A}_*|^{n|\mathcal{O}_*|}+\frac{(|\mathcal{A}|\,|\mathcal{O}|)^{h}-1}{(|\mathcal{A}|\,|\mathcal{O}|)-1}|\mathcal{A}||\mathcal{S}|\right). \tag{3.3.7}$$

Comparing to the cost of computing V_0 given by (3.2.5), this contains an additional exponential term, but this term does not depend on the horizon of the problem.

As discussed in Section 3.2, V_0, the value function of the underlying POMDP, can be efficiently approximated by exploiting the PWLC-property of the value function. Hsu and Marcus (1982) showed that the value function of their formulation for 1-step delayed communication also preserves the PWLC property. Not surprisingly, $V_1^{t,*}(\boldsymbol{b}^t,\mathbf{a}^t)$ in (3.3.6) is also PWLC over the joint belief space (Oliehoek et al., 2007c) and, as a result, approximation methods for POMDPs can be transferred to its computation (Oliehoek et al., 2007b).

3.4 k-Steps Delayed Communication

This section describes the setting where there is cost-free and noise-free communication, but there is a delay of k stages. Aicardi, Davoli, and Minciardi (1987) and Ooi and Wornell (1996) performed similar work on decentralized control in which there is a k-steps delayed state observation. That is, they consider the setting where, at time step t, all agents i know their own observations $o_i^0,\ldots,o_i^t$ and the states that have occurred up to $t-k$: $s^0,\ldots,s^{t-k}$. In particular Aicardi et al. (1987) consider the Dec-MDP setting in which agent i's observations are local states $\hat{s}_i$ and where a joint observation identifies the state $s=\langle\hat{s}_1,\ldots,\hat{s}_n\rangle$. As a consequence the delayed observation of the state s^{t-k} can be interpreted as the result of k-steps delayed communication in a Dec-MDP. Aicardi et al. present a dynamic programming formulation to optimally solve the finite-horizon problem. Ooi and Wornell (1996) examine the decentralized control of a broadcast channel over an infinite horizon and present a reformulation of the stochastic control problem, that is very close to the description that will be presented in this section. Ooi and Wornell extend upon the previous work by splitting the (individual) observations in a local and global part, lowering the complexity of evaluating the resulting dynamic program.

In this section we will present a reformulation and minor extension of previous work by demonstrating the solution for Dec-POMDPs where there is a k-steps delayed communication of the individual observation. That is, for systems where not the state, but the joint action-observation history is perceived with a k stage delay: at stage t, each agent i knows $\vec{\theta}^{t-k}$ and its individual $\vec{\theta}_i^t$. For these systems the optimal value functions are discussed, reformulated to naturally fit in the overview of different communication assumptions. A main contribution is the result that decreased communication delays cannot decrease the expected value of decentralized systems in Section 3.4.4. For the centralized setting a similar result was shown by Bander and White (1999). Although it is a very intuitive result, and Ooi and Wornell (1996) use this intuition to motivate their approach, for the decentralized setting a formal proof had been lacking.

3.4.1 Modeling Systems with k-Steps Delay

In the setting of k-steps delayed communication, at stage t each agent agent knows $\vec{\theta}^{t-k}$, the joint action-observation history of k stages earlier, and therefore can compute $\boldsymbol{b}^{t-k}$ the joint belief induced by $\vec{\theta}^{t-k}$. Again, $\boldsymbol{b}^{t-k}$ is a Markov signal, so no further history needs to be retained and $\boldsymbol{b}^{t-k}$ takes the role of $\boldsymbol{b}^0$ in the no-communication setting and $\boldsymbol{b}^{t-1}$ in the one-step delay setting. Indeed, one-step delay is just a special case of the k-steps delay setting.

In contrast to the one-step delayed communication case, the agents do not know the last taken joint action. However, since we assume the agents know each other policies, they do know $\boldsymbol{q}^{\tau=k,t-k}$, the joint policy that has been executed during stages $t-k,\ldots,t-1$. This $\boldsymbol{q}^{\tau=k,t-k}$ is a length-k joint sub-tree policy rooted at stage $t-k$: it specifies a sub-tree policy $q_i^{\tau=k,t-k}$ for each agent i which is a policy tree that specifies actions for k stages $t-k,\ldots,t-1$. [1]

In this section we more concisely write $\boldsymbol{q}_{|k|}^{t-k}$ for a length-k joint sub-tree policy. The technicalities of how these sub-tree policies should be maintained are somewhat involved and require some more notation that will first be introduced. Afterward we will define optimal value functions and discuss the complexity of computing them.

Let us assume that at a particular stage t the situation is as depicted in the top half of Figure 3.5: the system with 2 agents has $k=2$ steps delayed communication, so each agent knows $\boldsymbol{b}^{t-k}$ and $\boldsymbol{q}_{|k|}^{t-k}$ the joint sub-tree policy that has been executed during stages $t-k$,$t-1$. At this point the agents need to select an action, but they don't know each others individual observation history since stage $t-k$. That is they have uncertainty with respect to the length-k observation history $\vec{\mathbf{o}}_{|k|}^t = (\mathbf{o}^{t-k+1},\ldots,\mathbf{o}^t)$. Effectively, this means that the agents have to use a joint BG-policy $\boldsymbol{\beta}_{|k|}^t = \langle \beta_{1|k|}^t,\ldots,\beta_{n|k|}^t \rangle$ that implicitly maps length-k observation histories to joint actions $\boldsymbol{\beta}_{|k|}^t(\vec{\mathbf{o}}_{|k|}^t) = \mathbf{a}^t$.

We assume that in the planning phase we computed such a joint BG-policy $\boldsymbol{\beta}_{|k|}^t$

[1] Remember from Subsection 2.6.5 that we use τ to denote the number of steps-to-go and in the context of a sub-tree policy τ refers to its 'length': the number of time-steps for which it specifies actions.

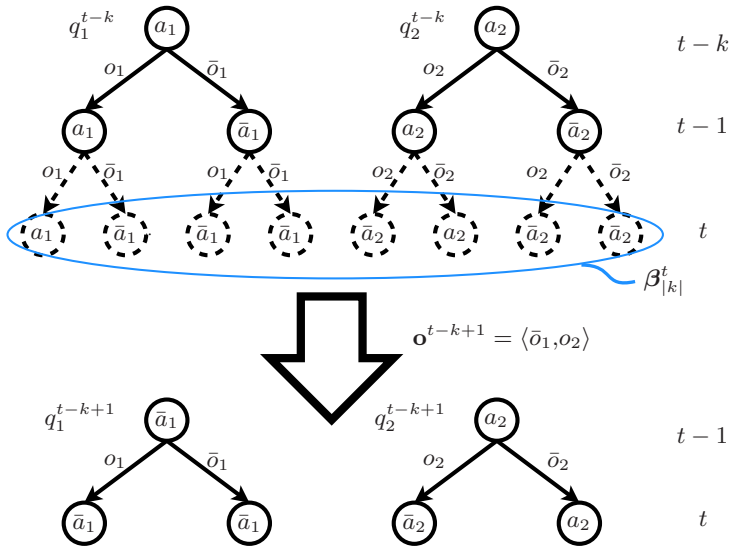


Figure 3.5: Sub-tree policies in a system with $k = 2$ steps delayed communication. Top: policies at $t - k$. The policies are extended by a joint BG-policy $\boldsymbol{\beta}^t_{|k|}$ shown dashed. Bottom: The resulting policies after joint observation $\langle \bar{o}_1, o_2 \rangle$.

as indicated in the figure. As is shown, $\boldsymbol{\beta}^t_{|k|}$ can be used to extend the sub-tree policy $\boldsymbol{q}^{t-k}_{|k|}$ to form a longer sub-tree policy with $\tau = k + 1$ stages-to-go. Each agent has knowledge of this extended joint sub-tree policy

$$\boldsymbol{q}^{t-k}_{|k+1|} = \langle \boldsymbol{q}^{t-k}_{|k|} \circ \boldsymbol{\beta}^t_{|k|} \rangle.$$

Consequently each agent i executes the action corresponding to its individual observation history $\beta^t_{i|k|}(\vec{o}^t_{i|k|}) = a^t_i$ and a transition occurs to stage $t + 1$. At that point each agent receives a new observation o^{t+1}_i through perception and the joint observation $\mathbf{o}^{t-k+1}$ through communication, it transmits its individual observation, and computes $\boldsymbol{b}^{t-k+1}$. Now, all the agents know what action was taken at $t - k$ and what the following observation $\mathbf{o}^{t-k+1}$ was. Therefore the agents know which part of $\boldsymbol{q}^{t-k}_{|k+1|}$ has been executed during the last k stages $t - k + 1, \ldots, t$ and they discard the part not needed further. I.e., the joint observation 'consumes' part of the joint sub-tree policy.

Definition 3.2 (Policy consumption)**.** Feeding a length-k joint sub-tree policy $\boldsymbol{q}$ with a sequence $l < k$ joint observations *consumes* a part of $\boldsymbol{q}$ leading to a joint sub-tree policy $\boldsymbol{q}'$ which is a sub-tree of $\boldsymbol{q}$. In particular, consumption $\Downarrow$ by a single joint observation $\mathbf{o}^{t-k+1}$ is written as

$$\boldsymbol{q}^{t-k+1}_{|k|} = \boldsymbol{q}^{t-k}_{|k+1|} \Downarrow_{\mathbf{o}^{t-k+1}}. \tag{3.4.1}$$

This process is illustrated in the bottom part of Figure 3.5. Policy consumption also applies to joint BG-policies:

$$\boldsymbol{\beta}^t_{|k-1|} = \boldsymbol{\beta}^t_{|k|}\Downarrow_{\mathbf{o}^{t-k+1}}$$

Proposition 3.3 (Distributivity of policy operations)**.** *Policy concatenation and consumption are distributive. That is*

$$\langle \boldsymbol{q}^{t-k}_{|k|}\Downarrow_{\mathbf{o}^{t-k+1}} \circ \boldsymbol{\beta}^t_{|k|}\Downarrow_{\mathbf{o}^{t-k+1}}\rangle = \langle \boldsymbol{q}^{t-k}_{|k|} \circ \boldsymbol{\beta}^t_{|k|}\rangle\Downarrow_{\mathbf{o}^{t-k+1}}$$

Proof. This statement can easily be verified by inspection of Figure 3.5. A formal proof is omitted. □

3.4.2 Optimal Value Functions

This section discusses the optimal value functions under k-steps delayed communication. To ease notation we will simply write $\boldsymbol{q}^{t-k}$,$\boldsymbol{\beta}^t$ for $\boldsymbol{q}^{t-k}_{|k|}$,$\boldsymbol{\beta}^t_{|k|}$. With some abuse of notation we write $\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|})$ to denote $\boldsymbol{\beta}^t_{|k|}(\mathbf{o}^{t-k+1},\ldots,\mathbf{o}^t)$ the application of the length-k joint BG policy to the last k joint observations of $\vec{\boldsymbol{\theta}}^t_{|k|}$.

Also we will consider probabilities of the form $\Pr(\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k})$. These are defined as marginals of $\Pr(s^t,\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k})$ that are defined analogous to (2.5.6)

$$\begin{aligned}\Pr(s^t,\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}) = \Pr(\mathbf{o}^t|\mathbf{a}^{t-1},s^t)\sum_{s^{t-1}}\Pr(s^t|s^{t-1},\mathbf{a}^{t-1})\\ \Pr(\mathbf{a}^{t-1}|\vec{\boldsymbol{\theta}}^{t-1}_{|k-1|},\boldsymbol{q}^{t-k})\Pr(s^{t-1},\vec{\boldsymbol{\theta}}^{t-1}_{|k-1|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k})\end{aligned} \quad (3.4.2)$$

In a similar way as the no-communication and one-step delayed communication settings we have the following value function in the k-steps delayed communication case. That is, the 'start point' is $\boldsymbol{b}^{t-k}$ instead of $\boldsymbol{b}^0$ or $\boldsymbol{b}^{t-1}$ and $\boldsymbol{q}^{t-k}$ takes the role of respectively $\boldsymbol{\varphi}^t$, $\mathbf{a}^{t-1}$.

Lemma 3.2 (Value of k-steps delayed communication)**.** *The optimal value function for a finite-horizon Dec-POMDP with k-steps delayed, cost and noise free, communication is given by:*

$$V^{t,*}_k(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}) = \max_{\boldsymbol{\beta}^t} Q^{t,*}_k(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t). \quad (3.4.3)$$

$$Q^{t,*}_k(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t) = \sum_{\vec{\boldsymbol{\theta}}^t_{|k|}}\Pr(\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k})Q^{t,*}_k(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\vec{\boldsymbol{\theta}}^t_{|k|},\boldsymbol{\beta}^t) \quad (3.4.4)$$

$$\begin{aligned}Q^{t,*}_k(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\vec{\boldsymbol{\theta}}^t_{|k|},\boldsymbol{\beta}^t) = R(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|}))+\\ \sum_{\mathbf{o}^{t+1}}\Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|}))Q^{t+1,*}_k(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1},\vec{\boldsymbol{\theta}}^{t+1}_{|k|},\boldsymbol{\beta}^{t+1,*})\end{aligned} \quad (3.4.5)$$

where $\boldsymbol{b}^t$ *results from* $\boldsymbol{b}^{t-k}$,$\vec{\boldsymbol{\theta}}^t_{|k|}$, *and*

$$\boldsymbol{\beta}^{t+1,*} = \arg\max_{\boldsymbol{\beta}^{t+1}} \sum_{\vec{\boldsymbol{\theta}}^{t+1}_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^{t+1}_{|k|} | \boldsymbol{b}^{t-k+1}, \boldsymbol{q}^{t-k+1}) Q_k^{t+1,*}(\boldsymbol{b}^{t-k+1}, \boldsymbol{q}^{t-k+1}, \vec{\boldsymbol{\theta}}^{t+1}_{|k|}, \boldsymbol{\beta}^{t+1}) \tag{3.4.6}$$

Sketch of proof. We will show that

$$\begin{aligned} &Q_k^{t,*}(\vec{\boldsymbol{\theta}}^{t-k}, \boldsymbol{q}^{t-k}, \boldsymbol{\beta}^t) \\ &\quad = E\Big[R(s^t, \mathbf{a}^t) | \vec{\boldsymbol{\theta}}^{t-k}, \boldsymbol{q}^{t-k}, \boldsymbol{\beta}^t\Big] + E\left[V_k^{t+1,*}(\vec{\boldsymbol{\theta}}^{t-k+1}, \boldsymbol{q}^{t-k+1}) \;\middle|\; \vec{\boldsymbol{\theta}}^{t-k}, \boldsymbol{q}^{t-k}, \boldsymbol{\beta}^t\right] \end{aligned} \tag{3.4.7}$$

and that the other equations are consistent with this definition. This means that, for the last stage (3.4.3) maximizes the expected reward and therefore is optimal, optimality for other stages follows immediately by induction. The full proof is listed in the Appendix D. □

Again, there is a clear correspondence between the equations presented here and those for the no-communication and one-step delayed communication setting. In particular, (3.4.3) corresponds to (3.3.1), (3.1.17) and (3.4.4) corresponds to (3.3.2),(3.1.18). More relations between the value functions are discussed in Appendix B.

3.4.3 Complexity

The equations in Lemma 3.2 form a dynamic program that can be evaluated from end to begin. We give an analysis of the complexity by considering the number of entries of $V_k^{h-1,*}(\boldsymbol{b}^{h-1-k}, \boldsymbol{q}^{h-1-k})$ and the amount of work needed to compute one entry. The number of length-k joint sub-tree policies $\boldsymbol{q}$ is

$$O\left(|\mathcal{A}_*|^{\frac{n(|\mathcal{O}_*|^k - 1)}{|\mathcal{O}_*| - 1}}\right) \tag{3.4.8}$$

where $\frac{|\mathcal{O}_*|^k - 1}{|\mathcal{O}_*| - 1} = \sum_{t=0}^{k-1} |\mathcal{O}_*|^t$. The number of joint beliefs $\boldsymbol{b}$ is bounded by the number of $\vec{\boldsymbol{\theta}}$ that induce those joint beliefs, is given by:

$$\sum_{t=0}^{h-1-k} \left|\vec{\boldsymbol{\Theta}}^t\right| = \sum_{t=0}^{h-1-k} (|\mathcal{A}|\,|\mathcal{O}|)^t = \frac{(|\mathcal{A}|\,|\mathcal{O}|)^{h-k}}{(|\mathcal{A}|\,|\mathcal{O}|) - 1}. \tag{3.4.9}$$

The maximization over $\boldsymbol{\beta}$ needs to consider $|\mathcal{A}_*|^{n|\mathcal{O}_*|^k}$ joint policies. As a result, complexity induced by selecting the best $\boldsymbol{\beta}$ for each $\boldsymbol{b}^{h-1-k}$,$\boldsymbol{q}^{h-1-k}$ is

$$O\left(\frac{(|\mathcal{A}|\,|\mathcal{O}|)^{h-k}}{(|\mathcal{A}|\,|\mathcal{O}|) - 1} \cdot |\mathcal{A}_*|^{\frac{n|\mathcal{O}_*|^k}{|\mathcal{O}_*| - 1}} \cdot |\mathcal{A}_*|^{n|\mathcal{O}_*|^k}\right) \tag{3.4.10}$$

which should be compared to the first term in (3.3.2). The total complexity is the sum of (3.4.10) and the complexity induced by computing the expected immediate rewards. Careful inspection of equations in Lemma 3.2 reveals that for each $\boldsymbol{b}^t$ induced by a pair $(\boldsymbol{b}^{t-k},\vec{\boldsymbol{\theta}}^t_{|k|})$ the expected immediate reward will need to be computed for each joint action. The number of such $\boldsymbol{b}^t$ is again bounded by the number of joint AOHs. Therefore, the complexity induced by these computations is

$$O(\frac{(|\mathcal{A}|\,|\mathcal{O}|)^h - 1}{(|\mathcal{A}|\,|\mathcal{O}|) - 1}|\mathcal{A}||\mathcal{S}|)$$

as before in (the right side of) (3.3.2).

In the sections on immediate and 1-step delayed communication, we mentioned that because these value functions are PWLC over the joint belief space, we could use efficient approximate computation methods. Clearly it would be beneficial to also perform approximate computation for V_k. Unfortunately is turns out that this is not straightforward: for $k \geq 2$ Varaiya and Walrand (1978) showed that k-steps delayed communication systems[1] are not separable. The result of this is that, just as in the Dec-POMDP case (Oliehoek et al., 2007c), V_k is not a function of the joint belief space, let alone PWLC over this space.

3.4.4 Less Delay Cannot Decrease Value

This section shows that when the delay of communication decreases, the expected value cannot become less. That is, less delay in communication is not harmful. A related result by Bander and White (1999) shows that for a single agent POMDP with delayed observations, decreasing delays will not decrease the expected return. For the decentralized case, however, no such results were available.

Theorem 3.3 (Shorter communication delays cannot decrease the expected value)**.** *The expected value over stages $t,\ldots,h-1$ given a joint belief $\boldsymbol{b}^{t-k-1}$ and joint policy $\boldsymbol{q}^{t-k-1}_{|k+1|}$ followed during stages $t-k-1,\ldots,t-1$ is no less under k-steps communication delay, than under $(k+1)$-steps delay. That is*

$$\forall_t \forall_{\boldsymbol{b}^{t-k-1}} \forall_{\boldsymbol{q}^{t-k-1}_{|k+1|}} \quad E\big[V^{t,*}_k(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}_{|k|}) \mid \boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|}\big] \geq V^{t,*}_{k+1}(\boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|}), \tag{3.4.11}$$

where the expectation on the left-hand side is over joint observations $\mathbf{o}^{t-k}$ that together with $\boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|}$ induce $\boldsymbol{b}^{t-k}$.

Sketch of proof. The proof is by induction. The base case is that (3.4.11) holds for the last stage $t = h-1$. The full proof is listed in Appendix D. □

3.5 Conclusions

A large body of work in single-agent decision-theoretic planning is based on value functions, but such theory has been lacking thus far for Dec-POMDPs. Given

[1] Varaiya and Walrand refer to k units delayed 'sharing patterns'.

the large impact of value functions on single-agent planning under uncertainty, we expect that a thorough study of value functions for Dec-POMDPs can greatly benefit multiagent planning under certainty. This chapter presented a framework of Q-value functions for Dec-POMDPs under different communication assumptions, providing a significant contribution to fill this gap in Dec-POMDP theory.

The main contributions are for the setting without communication, where an optimal joint policy $\boldsymbol{\pi}^*$ induces an optimal Q-value function $Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$, and how it is possible to construct an optimal joint policy $\boldsymbol{\pi}^*$ using forward-sweep policy computation. This entails solving Bayesian games for time steps $t = 0,\ldots,h-1$ which use $Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$ as the payoff function. Because $Q_{\boldsymbol{\pi}^*}$ implicitly depends on the optimal past joint policy, there is no clear way to compute $Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$ directly. To overcome this problem, we introduced a different description of the optimal Q-value function $Q^*(\boldsymbol{b}^0,\boldsymbol{\varphi}^{h-1},\vec{\boldsymbol{\theta}}^{h-1},\boldsymbol{\delta}^{h-1})$ this makes the dependence on the past joint policy explicit. This new description of Q^* can be computed using dynamic programming and can then be used to construct $\boldsymbol{\pi}^*$.

Another important contribution of this chapter is that it shows that a decrease in communication delay cannot lead to a decrease in expected return. That is, shorter communication delays are not harmful.

Finally, the chapter has presented a unified overview of the optimal value functions under various delays of communication and discussed how they relate to each other. Two formulations, the expected reward and immediate reward formulation, were identified, of which the former is used in this chapter. The latter form and the relations between the two is explained in Appendix B.

3.6 Future Work

An interesting direction of future work is to try to extend the results of this chapter to partially observable stochastic games (POSGs) (Hansen et al., 2004), which are Dec-POMDPs with an individual reward function for each agent. Since the dynamics of the POSG model are identical to those of a Dec-POMDP, a similar modeling via Bayesian games is possible when allowing the past policy to be stochastic. An interesting question is whether also in this case, an 'optimal' joint policy (i.e., a Pareto-optimal Bayes-Nash equilibrium) can be found by forward-sweep policy computation.

Chapter 4

Approximate Value Functions & Heuristic Policy Search

In the previous chapter it was discussed that a Dec-POMDP can be represented by a series of Bayesian games (BGs), one for each stage. In these BGs, the payoff functions $Q(\vec{\boldsymbol{\theta}},\mathbf{a})$ of these BGs represent the expected return. A solution for the Dec-POMDP can be computed by solving the BGs for subsequent stages, a procedure first proposed by Emery-Montemerlo et al. (2004) and to which we refer as *forward-sweep policy computation (FSPC).* When using a heuristic Q-value function as the payoff function for the BGs FSPC finds an approximate solution. However, the quality of such solutions may be poor. When using an optimal payoff function Q^*, the procedure can find an optimal solution. However, Q^* is costly to compute and thus impractical.

This chapter brings leverage to this problem in two ways. First, we investigate approximate Q-value functions that are easier to compute. In particular, we reuse the value-functions for 0 and 1-step delayed communication identified in the previous chapter. Next, we mitigate the problem of poor solutions by allowing backtracking. The resulting value-based policy search algorithm named GENERALIZED MULTIAGENT A* (GMAA*) unifies FSPC and MAA*.

4.1 Approximate Q-Value Functions

Although an optimal Q-value function Q^* exists, it is costly to compute and thus impractical. In this section, we review some other Q-value functions, $\widehat{Q}$, that can be used as an approximation for Q^*. We will discuss underlying assumptions, computation, computational complexity and other properties, thereby providing a taxonomy of approximate Q-value functions for Dec-POMDPs. In particular we will treat two well-known approximate Q-value functions, $\mathrm{Q_{MDP}}$ and $\mathrm{Q_{POMDP}}$, and $\mathrm{Q_{BG}}$ recently introduced by Oliehoek and Vlassis (2007a). We also establish that these value functions are *admissible heuristics* when used in FSPC. I.e., they are guaranteed overestimates of Q^* the optimal Q-value function.

4.1.1 $\mathrm{Q_{MDP}}$

$\mathrm{Q_{MDP}}$ was originally proposed to approximately solve POMDPs by Littman, Cassandra, and Kaelbling (1995), but has also been applied to Dec-POMDPs (Emery-Montemerlo et al., 2004; Szer et al., 2005). The idea is that Q^* can be approximated using the state-action values $Q_{\mathrm{M}}(s,\mathbf{a})$ found when solving the 'underlying MDP' of a Dec-POMDP. This underlying MDP is the horizon-h MDP defined by a single 'puppeteer' agent that takes joint actions $\mathbf{a} \in \mathcal{A}$ and observes the nominal state s that has the same transition model T and reward model R as the original Dec-POMDP. Solving this underlying MDP can be efficiently done using dynamic programming techniques (Puterman, 1994), resulting in the optimal non-stationary MDP Q-value function:

$$Q_{\mathrm{M}}^{t,*}(s^t,\mathbf{a}) = R(s^t,\mathbf{a}) + \sum_{s^{t+1}\in\mathcal{S}} \Pr(s^{t+1}|s^t,\mathbf{a}) \max_{\mathbf{a}} Q_{\mathrm{M}}^{t+1,*}(s^{t+1},\mathbf{a}). \tag{4.1.1}$$

Note that $Q_{\mathrm{M}}^{t,*}$ also is an optimal Q-value function, but in the MDP setting. Q^* denotes the optimal value function for the (original) Dec-POMDP. In order to transform the $Q_{\mathrm{M}}^{t,*}(s^t,\mathbf{a})$-values to approximate $\widehat{Q}_{\mathrm{M}}(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$-values to be used for the original Dec-POMDP, we compute:

$$\widehat{Q}_{\mathrm{M}}(\vec{\boldsymbol{\theta}}^t,\mathbf{a}) = \sum_{s\in\mathcal{S}} Q_{\mathrm{M}}^{t,*}(s,\mathbf{a}) \Pr(s|\vec{\boldsymbol{\theta}}^t,\boldsymbol{b}^0), \tag{4.1.2}$$

where $\Pr(s|\vec{\boldsymbol{\theta}}^t,\boldsymbol{b}^0)$ can be computed from (3.1.4). Note that $\widehat{Q}_{\mathrm{M}}$ is consistent with the established definition of Q-value functions since it is defined as the expected immediate reward of performing (joint) action $\mathbf{a}$ plus the value of following an optimal joint policy (in this case the optimal MDP-policy) thereafter. This can be seen by combining (4.1.1) and (4.1.2):

$$\widehat{Q}_{\mathrm{M}}(\vec{\boldsymbol{\theta}}^t,\mathbf{a}) = R(\vec{\boldsymbol{\theta}}^t,\mathbf{a}) + \sum_{s^{t+1}\in\mathcal{S}} \Pr(s^{t+1}|\vec{\boldsymbol{\theta}}^t,\mathbf{a}) \max_{\mathbf{a}'} Q_{\mathrm{M}}^{t+1,*}(s^{t+1},\mathbf{a}'), \tag{4.1.3}$$

Because calculation of the $Q_{\mathrm{M}}^{t}(s,\mathbf{a})$-values by dynamic programming (which has a cost of $O(|\mathcal{S}|^2 \cdot |\mathcal{A}| \cdot h)$) can be performed in a separate phase, the cost of computation of $\mathrm{Q_{MDP}}$ is only dependent on the cost of evaluation of (4.1.3), which is $O(|\mathcal{S}|\,|\mathcal{A}|)$. When we want to evaluate $\mathrm{Q_{MDP}}$ for all $\sum_{t=0}^{h-1} (|\mathcal{A}|\,|\mathcal{O}|)^t = \frac{(|\mathcal{A}||\mathcal{O}|)^h-1}{(|\mathcal{A}||\mathcal{O}|)-1}$ joint action-observation histories is, the total computational cost becomes:

$$O\left(\frac{(|\mathcal{A}|\,|\mathcal{O}|)^h - 1}{(|\mathcal{A}|\,|\mathcal{O}|) - 1} |\mathcal{A}|^2 |\mathcal{S}|\right). \tag{4.1.4}$$

However, when applying $\mathrm{Q_{MDP}}$ in forward-sweep policy computation, we do not have to consider *all* action-observation histories, but only those that are consistent with the policy found for earlier stages. Effectively we only have to evaluate (4.1.3) for all observation histories and joint actions, leading to:

$$O\left(\frac{(|\mathcal{O}|)^h - 1}{(|\mathcal{O}|) - 1} |\mathcal{A}|^2 |\mathcal{S}|\right). \tag{4.1.5}$$

When used in the context of POMDPs, $\mathrm{Q}_{\mathrm{MDP}}$ solutions are known to undervalue actions that gain information (Fernández, Sanz, Simmons, and Diéguez, 2006). This is explained by realizing that (4.1.3) assumes that the state will be fully observable in the next time step. Therefore actions that provide information about the state, and thus can lead to a high future reward (but might have a low immediate reward), will be undervalued. When applying $\mathrm{Q}_{\mathrm{MDP}}$ in the Dec-POMDP setting, this effect can also be expected. Another consequence of the simplifying assumption is that the $\mathrm{Q}_{\mathrm{MDP}}$-value function is an upper bound to the optimal value function when used to approximate a POMDP (Hauskrecht, 2000), i.e., it is an *admissible heuristic* (Russell and Norvig, 2003). As a consequence it is also an upper bound to the optimal value function of a Dec-POMDP. This is intuitively clear, as a Dec-POMDP is a POMDP but with the additional difficulty of decentralization. A formal argument will be presented in Section 4.1.4.

4.1.2 $\mathrm{Q}_{\mathrm{POMDP}}$

Similar to the underlying MDP, one can define the 'underlying POMDP' of a Dec-POMDP as the POMDP with the same T, O and R, but in which there is only a single agent that takes joint actions $\mathbf{a} \in \mathcal{A}$ and receives joint observations $\mathbf{o} \in \mathcal{O}$. Note that the underlying POMDP is identical to the multiagent POMDP to which a Dec-POMDP reduces under instantaneous communication, as was discussed in Section 3.2. $\mathrm{Q}_{\mathrm{POMDP}}$ approximates Q^* using the solution of the underlying POMDP (Szer et al., 2005; Roth et al., 2005a).

Therefore, the $\mathrm{Q}_{\mathrm{POMDP}}$ function really is just Q_0 as given by (3.2.2). For each $\vec{\boldsymbol{\theta}}^t$ there is one joint belief $\boldsymbol{b}^t$. In this chapter we will on a few occasions denote such a joint belief by $\boldsymbol{b}^{\vec{\boldsymbol{\theta}}^t}$ to make explicit the history that it is induced by. Now, it is possible to directly use Q_0-values as payoffs for the BGs of the Dec-POMDP. That is, we define the approximate $\mathrm{Q}_{\mathrm{POMDP}}$ value function $\widehat{Q}_{\mathrm{P}}$ as

$$\widehat{Q}_{\mathrm{P}}(\vec{\boldsymbol{\theta}}^t, \mathbf{a}) \equiv Q_0^*(\boldsymbol{b}^{\vec{\boldsymbol{\theta}}^t}, \mathbf{a}). \tag{4.1.6}$$

It is intuitively clear that $\mathrm{Q}_{\mathrm{POMDP}}$ is an admissible heuristic for Dec-POMDPs, as it still assumes that more information is available than actually is the case (again a formal proof will be given in Section 4.1.4). Also it should be clear that, as fewer assumptions are made, $\mathrm{Q}_{\mathrm{POMDP}}$ should yield less of an over-estimation than $\mathrm{Q}_{\mathrm{MDP}}$. I.e., the $\mathrm{Q}_{\mathrm{POMDP}}$-values should lie between the $\mathrm{Q}_{\mathrm{MDP}}$ and optimal Q^*-values.

In contrast to $\mathrm{Q}_{\mathrm{MDP}}$, $\mathrm{Q}_{\mathrm{POMDP}}$ does not assume full observability of nominal states. As a result the latter does not share the drawback of undervaluing actions that will gain information regarding the nominal state. When applied in a Dec-POMDP setting, however, $\mathrm{Q}_{\mathrm{POMDP}}$ does share the assumption of centralized control. This assumption might also cause a relative undervaluation: there might be situations where some action might gain information regarding the joint (i.e., each other's) observation history. Under $\mathrm{Q}_{\mathrm{POMDP}}$ this will be considered redundant, while in decentralized execution this might be very beneficial, as it allows for better coordination.

4.1.3 $\mathrm{Q_{BG}}$

$\mathrm{Q_{MDP}}$ approximates Q^* by assuming that the state becomes fully observable in the next time step, while $\mathrm{Q_{POMDP}}$ assumes that at every time step t the agents know the joint action-observation history $\vec{\boldsymbol{\theta}}^t$ (i.e., it assumes instantaneous communication). Here we present a new approximate Q-value function, called $\mathrm{Q_{BG}}$, that relaxes the assumptions further: it assumes that the agents know $\vec{\boldsymbol{\theta}}^{t-1}$, the joint action-observation history up to time step $t-1$, and the joint action $\mathbf{a}^{t-1}$ that was taken at the previous time step. That is, it assumes the setting of one-step delayed communication, as described in Section 3.3.

As such we define the $\mathrm{Q_{BG}}$ heuristic $\widehat{Q}_{\mathrm{B}}$ as

$$\widehat{Q}_{\mathrm{B}}(\vec{\boldsymbol{\theta}}^t,\mathbf{a}) \equiv V_1^*(\boldsymbol{b}^{\vec{\boldsymbol{\theta}}^t},\mathbf{a}), \tag{4.1.7}$$

with V_1^* as defined in (3.3.6).

4.1.4 Generalized $\mathrm{Q_{BG}}$ and Bounds

In the previous two subsections, the value functions for systems under respectively 0 and 1 step delayed communication have been used to define heuristic payoff functions for BGs that represent a stage of a Dec-POMDP. An obvious idea is to extend to using V_k, the value function of a k-steps delayed communication system.

However, because such a system is not separable, it is not trivial to derive a heuristic payoff function of the form $\widehat{Q}(\vec{\boldsymbol{\theta}}^t,\mathbf{a}^t)$. In particular, we have that the expected immediate reward formulation is of the form $Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\vec{\boldsymbol{\theta}}^t_{|k|},\boldsymbol{\beta}^t_{|k|})$. While it is possible to split $\vec{\boldsymbol{\theta}}^t$ in $(\boldsymbol{b}^{\vec{\boldsymbol{\theta}}^{t-k}},\vec{\boldsymbol{\theta}}^t_{|k|})$ and $\boldsymbol{q}^{t-k}$ is given by the past joint policy $\boldsymbol{\varphi}^t$, there is no single value for $\mathbf{a}^t$. That is, there are many $\boldsymbol{\beta}^t_{|k|}$ that specify $\boldsymbol{\beta}^t_{|k|}(\vec{\boldsymbol{\theta}}^t_{|k|}) = \mathbf{a}^t$. One option is the select the one that maximizes the value, but is not obvious whether this is indeed the best choice and more study is needed.

There is another use of the k-step delayed communication model. It allows expressing the different Q-value functions defined in this section as optimal value functions of appropriately chosen k-step delay models. Since we know that a shorter delay cannot decrease the optimal value, we can prove a hierarchy of bounds that hold over the various Q-functions.

Theorem 4.1 (Hierarchy of upper bounds)**.** *The approximate Q-value functions Q_{BG} and Q_{POMDP} correspond to the optimal Q-value functions of appropriately defined k-step delayed communication models. Moreover these Q-value functions form a hierarchy of upper bounds to the optimal Q^* of the Dec-POMDP:*

$$Q^* \leq Q_{BG} \leq Q_{POMDP} \leq Q_{MDP}. \tag{4.1.8}$$

Proof. $\mathrm{Q_{POMDP}}$ corresponds to a system with no (0-steps) delayed communication, while the $\mathrm{Q_{BG}}$-setting corresponds to a 1-step delayed communication system. Q^* corresponds to the value of a h-step delayed communication system. Theorem 3.3 and Theorem B.1 show that both the expected and immediate reward value functions of a system with k steps delay forms an upper bound to that of a decentralized

system with $k+1$ steps delay. We note that the last inequality of (4.1.8) is a well-known result (Hauskrecht, 2000). □

4.1.5 Recursive Solution

Szer et al. (2005) consider a heuristic that employs the recursive solution of problems of horizon $h-1$ to which they refer as 'recursive MAA*'. In particular $\widehat{V}^{t\ldots h-1}(\boldsymbol{\varphi}^t)$, the heuristic value for the remaining $h-t$ stages is defined as

$$\widehat{V}^{t\ldots h-1}(\boldsymbol{\varphi}^t) = \sum_{s^t \in \mathcal{S}} \Pr(s^t|\boldsymbol{b}^0,\boldsymbol{\varphi}^t) V^*(s^t), \tag{4.1.9}$$

where $V^*(s^t)$ is the optimal value of the horizon $h-t$ Dec-POMDP starting in state s^t (found by recursive solution of this shorter Dec-POMDP). For clarity we will denote this shorter Dec-POMDP by D' and the original horizon-h Dec-POMDP by D.

Szer et al. (2005) reason that the optimal value is the tightest possible heuristic. However, they employ the optimal value function of D', not of D (for which the optimal value function was presented in Section 3.1.5). In particular $V^*(s^t)$ assumes that at stage t (so that is the initial stage $t' = 0$ in D'), there is no uncertainty regarding the history, which clearly results in an over-estimation. Moreover, linear combination of optimal values V^* results in another over estimation. I.e., even though it is possible to take a linear combination of values *for a fixed joint policy* as in (2.5.8), we have that

$$\begin{aligned} V^*(\boldsymbol{b}^0) \equiv \max_{\boldsymbol{\pi}} \sum_{s^0 \in \mathcal{S}} \boldsymbol{b}^0(s^0) V_{\boldsymbol{\pi}}(s^0,\vec{\boldsymbol{\theta}}_\emptyset) \leq \\ \sum_{s^0 \in \mathcal{S}} \boldsymbol{b}^0(s^0) \max_{\boldsymbol{\pi}} V_{\boldsymbol{\pi}}(s^0,\vec{\boldsymbol{\theta}}_\emptyset) = \sum_{s^0 \in \mathcal{S}} \boldsymbol{b}^0(s^0) V^*(s^0). \end{aligned} \tag{4.1.10}$$

That is, such linear combination as in (4.1.9) is an overestimation because it assumes correctly guessing the hidden state and then executing the optimal joint policy.

4.2 Generalized Value-Based Policy Search

The hierarchy of approximate Q-value functions implies that all of these Q-value functions can be used as *admissible heuristics* in MAA* policy search, treated in Section 2.6.4. In this section we will present a more general heuristic policy search framework which we will call Generalized MAA* (GMAA*), and show how it unifies some of the solution methods proposed for Dec-POMDPs.

GMAA* generalizes MAA* (Szer et al., 2005) by making explicit different procedures that are implicit in MAA*: (1) iterating over a pool of partial joint policies, pruning this pool whenever possible, (2) selecting a partial joint policy from the policy pool, and (3) finding some new partial and/or full joint policies

Algorithm 4.1 GMAA*

$\underline{v}^\star \leftarrow -\infty$
$\mathrm{P} \leftarrow \{\boldsymbol{\varphi}^0 = ()\}$
repeat
  $\boldsymbol{\varphi}^t \leftarrow \texttt{Select}(\mathrm{P})$
  $\Phi_{\texttt{Expand}} \leftarrow \texttt{Expand}(\boldsymbol{\varphi}^t)$
  if $\Phi_{\texttt{Expand}}$ contains a subset of full policies $\boldsymbol{\Pi}_{\texttt{Expand}} \subseteq \Phi_{\texttt{Expand}}$ **then**
    $\boldsymbol{\pi}' \leftarrow \arg\max_{\boldsymbol{\pi} \in \boldsymbol{\Pi}_{\texttt{Expand}}} V(\boldsymbol{\pi})$
    if $V(\boldsymbol{\pi}') > \underline{v}^\star$ **then**
      $\underline{v}^\star \leftarrow V(\boldsymbol{\pi}')$
      $\boldsymbol{\pi}^\star \leftarrow \boldsymbol{\pi}'$
      $\mathrm{P} \leftarrow \left\{ \boldsymbol{\varphi} \in \mathrm{P} \mid \widehat{V}(\boldsymbol{\varphi}) > \underline{v}^\star \right\}$ {prune the policy pool}
    end if
    $\Phi_{\texttt{Expand}} \leftarrow \Phi_{\texttt{Expand}} \setminus \boldsymbol{\Pi}_{\texttt{Expand}}$ {remove full policies}
  end if
  $\mathrm{P} \leftarrow \mathrm{P} \setminus \boldsymbol{\varphi}^t$ {remove processed $\boldsymbol{\varphi}^t$}
  $\mathrm{P} \leftarrow \mathrm{P} \cup \{\boldsymbol{\varphi} \in \Phi_{\texttt{Expand}} \mid \widehat{V}(\boldsymbol{\varphi}) > \underline{v}^\star\}$ {add new $\boldsymbol{\varphi}$}
until P is empty

given the selected policy. The first procedure is the core of GMAA* and is fixed, while the other two procedures can be performed in many ways.

The second procedure, `Select`, chooses which policy to process next and thus determines the type of search (e.g., depth-first, breadth-first, A*-like) (Russell and Norvig, 2003; Bertsekas, 2005). The third procedure, which we will refer to as `Expand`, determines how the set of next (partial) joint policies are constructed, given a previous partial joint policy. The original MAA* can be seen as an instance of the generalized case with a particular `Expand`-operator, namely that shown in algorithm 4.2.

4.2.1 The GMAA* Algorithm

In GMAA* the policy pool P is initialized with a completely unspecified joint policy $\boldsymbol{\varphi}^0 = ()$ and the maximum lower bound (found so far) $\underline{v}^\star$ is set to $-\infty$. $\boldsymbol{\pi}^\star$ denotes the best joint policy found so far.

At this point GMAA* starts. First, the selection operator, `Select`, selects a partial joint policy $\boldsymbol{\varphi}$ from P. We will assume that, in accordance with MAA*, the partial policy with the highest heuristic value is selected. In general, however, any kind of selection algorithm may be used. Next, the selected policy is processed by the policy search operator `Expand`, which returns a set of (partial) joint policies $\Phi_{\texttt{Expand}}$ and their heuristic values. When `Expand` returns one or more full (i.e., fully specified) joint policies $\boldsymbol{\pi} \in \Phi_{\texttt{Expand}}$, the provided values $\widehat{V}(\boldsymbol{\pi}) = V(\boldsymbol{\pi})$ are a lower bound for an optimal joint policy, which can be used to prune the search space. Any found partial joint policies $\boldsymbol{\varphi} \in \Phi_{\texttt{Expand}}$ with a heuristic value $\widehat{V}(\boldsymbol{\varphi}) > \underline{v}^\star$ are added to P. The process is repeated until the policy pool is empty.

Algorithm 4.2 `Expand`$(\boldsymbol{\varphi}^t)$ — MAA*

1: $\Phi^{t+1} \leftarrow \left\{\boldsymbol{\varphi}^{t+1} = \langle \boldsymbol{\varphi}^t \circ \boldsymbol{\delta}^t \rangle\right\}$
2: $\forall_{\boldsymbol{\varphi}^{t+1} \in \Phi^{t+1}} \quad \widehat{V}(\boldsymbol{\varphi}^{t+1}) \leftarrow V^{0\ldots t-1}(\boldsymbol{\varphi}^t) + E\left[R(s^t,\mathbf{a}) \middle| \boldsymbol{\varphi}^{t+1}\right] + \widehat{V}^{(t+1)\ldots h}(\boldsymbol{\varphi}^{t+1})$
3: **return** Φ^{t+1}

4.2.2 The `Expand` Operator

Here we describe some different choices for the `Expand`-operator and how they correspond to existing Dec-POMDP solution methods.

4.2.2.1 MAA*

GMAA* reduces to standard MAA* by using the `Expand`-operator described by Algorithm 4.2. Line 1 expands $\boldsymbol{\varphi}^t$ by appending all possible joint decision rules $\boldsymbol{\delta}^t$ for stage t. This results in Φ^{t+1} the set of partial joint policies of length $t+1$. Line 2 valuates all the $\boldsymbol{\varphi}^{t+1} \in \Phi^{t+1}$. The first part gives the true expected reward over the first $t+1$ stages:

$$V^{0\ldots t}(\boldsymbol{\varphi}^{t+1}) = V^{0\ldots t-1}(\boldsymbol{\varphi}^t) + E\left[R(s^t,\mathbf{a}) \middle| \boldsymbol{\varphi}^{t+1}\right]. \tag{4.2.1}$$

The second part, $\widehat{V}^{(t+1)\ldots h}(\boldsymbol{\varphi}^{t+1})$, is the heuristic value over stages $t+1,\ldots,h-1$ given that $\boldsymbol{\varphi}^{t+1}$ has been followed the first $t+1$ stages.

When using an admissible heuristic, GMAA* will never prune a partial policy that can be expanded into an optimal policy. When combining this with the fact that the MAA*-`Expand` operator returns all possible $\boldsymbol{\varphi}^{t+1}$ for a $\boldsymbol{\varphi}^t$, it is clear that when P becomes empty an optimal policy has been found.

4.2.2.2 Forward-Sweep Policy Computation

Forward-sweep policy computation, as introduced in Section 3.1.2, is described by algorithms 4.1 and 4.3 jointly. Given a partial joint policy $\boldsymbol{\varphi}^t$, the `Expand` operator now constructs and solves a BG for time step t. Because `Expand` in algorithm 4.3 only returns the best-ranked policy, P will never contain more than 1 joint policy and the whole search process reduces to solving BGs for time steps $0,\ldots,h-1$.

The approach of Emery-Montemerlo et al. (2004) is identical to forward-sweep policy computation, except that 1) smaller BGs are created by discarding or clustering low probability action-observation histories, and 2) the BGs are approximately solved by alternating maximization. Therefore this approach can also be incorporated in the GMAA* policy search framework by making the appropriate modifications in Algorithm 4.3.

4.2.2.3 A Unified Perspective of MAA* and FSPC

Here we will give a unified perspective of the MAA* and forward-sweep policy computation by examining the relation between the corresponding `Expand`-operators. In particular we show that, when using $\mathrm{Q}_{\mathrm{MDP}}$, $\mathrm{Q}_{\mathrm{POMDP}}$ or Q_{BG} as a heuristic, the

Algorithm 4.3 `Expand`$(\boldsymbol{\varphi}^t)$ — Forward-sweep policy computation

1: $BG \leftarrow \langle \mathcal{A}, \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\varphi}^t}, \Pr(\vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\varphi}^t}), \widehat{Q}^t \rangle$
2: **for all** $\boldsymbol{\beta} = \langle \beta_1, \ldots, \beta_n \rangle$ s.t. $\beta_i : \vec{\mathcal{O}}^t_i \rightarrow \mathcal{A}_i$ **do**
3: $\quad \widehat{V}^t(\boldsymbol{\beta}) \leftarrow \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\varphi}^t}} \Pr(\vec{\boldsymbol{\theta}}^t) \widehat{Q}^t(\vec{\boldsymbol{\theta}}^t, \boldsymbol{\beta}(\vec{\boldsymbol{\theta}}^t))$
4: $\quad \boldsymbol{\varphi}^{t+1} \leftarrow \langle \boldsymbol{\varphi}^t \circ \boldsymbol{\beta} \rangle$
5: $\quad \widehat{V}(\boldsymbol{\varphi}^{t+1}) \leftarrow V^{0 \ldots t-1}(\boldsymbol{\varphi}^t) + \widehat{V}^t(\boldsymbol{\beta})$
6: **end for**
7: **return** $\arg\max_{\boldsymbol{\varphi}^{t+1}} \widehat{V}(\boldsymbol{\varphi}^{t+1})$

sole difference between the two is that FSPC returns only the joint policy with the highest heuristic value.

Proposition 4.1. *If a heuristic $\widehat{Q}$ has the following form*

$$\widehat{Q}^t(\vec{\boldsymbol{\theta}}^t, \mathbf{a}) = R(\vec{\boldsymbol{\theta}}^t, \mathbf{a}) + \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1} | \vec{\boldsymbol{\theta}}^t, \mathbf{a}) \widehat{V}^{t+1}(\vec{\boldsymbol{\theta}}^{t+1}), \tag{4.2.2}$$

then for a partial policy $\boldsymbol{\varphi}^{t+1} = \left(\boldsymbol{\varphi}^t, \boldsymbol{\beta}^t\right)$

$$\sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\varphi}}} \Pr(\vec{\boldsymbol{\theta}}^t) \widehat{Q}^t(\vec{\boldsymbol{\theta}}^t, \boldsymbol{\beta}(\vec{\boldsymbol{\theta}}^t)) = E\left[R(s^t, \mathbf{a}) \middle| \boldsymbol{\varphi}^{t+1}\right] + \widehat{V}^{(t+1) \ldots h}(\boldsymbol{\varphi}^{t+1}) \tag{4.2.3}$$

holds.

Proof. The expectation of R^t given $\boldsymbol{\varphi}^{t+1}$ can be written as

$$E\left[R(s^t, \mathbf{a}) \middle| \boldsymbol{\varphi}^{t+1}\right] = \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\varphi}}} \Pr(\vec{\boldsymbol{\theta}}^t) \sum_{s \in \mathcal{S}} R(s, \boldsymbol{\varphi}^{t+1}(\vec{\boldsymbol{\theta}}^t)) \Pr(s | \vec{\boldsymbol{\theta}}^t) = \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\varphi}}} \Pr(\vec{\boldsymbol{\theta}}^t) R(\vec{\boldsymbol{\theta}}^t, \boldsymbol{\varphi}^{t+1}(\vec{\boldsymbol{\theta}}^t)). \tag{4.2.4}$$

Also, we can rewrite $\widehat{V}^{(t+1) \ldots h}(\boldsymbol{\varphi}^{t+1})$ as

$$\widehat{V}^{(t+1) \ldots h}(\boldsymbol{\varphi}^{t+1}) = \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\varphi}}} \Pr(\vec{\boldsymbol{\theta}}^t) \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1} | \vec{\boldsymbol{\theta}}^t, \boldsymbol{\varphi}^{t+1}(\vec{\boldsymbol{\theta}}^t)) \widehat{V}^{(t+1) \ldots h}(\vec{\boldsymbol{\theta}}^{t+1}),$$

such that

$$E\left[R(s^t, \mathbf{a}) \middle| \boldsymbol{\varphi}^{t+1}\right] + \widehat{V}^{(t+1) \ldots h}(\boldsymbol{\varphi}^{t+1}) = \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\varphi}}} \Pr(\vec{\boldsymbol{\theta}}^t) \left[R(\vec{\boldsymbol{\theta}}^t, \boldsymbol{\varphi}^{t+1}(\vec{\boldsymbol{\theta}}^t)) + \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1} | \vec{\boldsymbol{\theta}}^t, \boldsymbol{\varphi}^{t+1}(\vec{\boldsymbol{\theta}}^t)) \widehat{V}^{(t+1) \ldots h}(\vec{\boldsymbol{\theta}}^{t+1}) \right]. \tag{4.2.5}$$

Therefore, assuming (4.2.2) yields (4.2.3). □

This means that if a heuristic satisfies (4.2.2), which is the case for all the Q-value functions discussed in Section 4.1, the `Expand` operators of algorithms 4.2 and 4.3 evaluate the expanded policies in the same way. I.e., algorithms 4.2 and 4.3 calculate identical heuristic values for identical next-stage joint policies. Also the expanded policies $\boldsymbol{\varphi}^{t+1}$ are formed in the same way: by considering all possible $\boldsymbol{\delta}^t$ respectively $\boldsymbol{\beta}^t$ to extend $\boldsymbol{\varphi}^t$. Therefore, the sole difference in this case is that the latter returns only the joint policy with the highest heuristic value.

Clearly, there is a computation time versus quality trade-off between MAA* and FSPC: MAA* is guaranteed to find an optimal policy (given an admissible heuristic), while FSPC is guaranteed to finish in one forward sweep. We propose a generalization, that returns the k-best ranked policies. We refer to this as the 'k-best joint BG policies' GMAA* variant, or k-GMAA*. In this way, k-GMAA* reduces to forward-sweep policy computation for $k = 1$ and to MAA* for $k = \infty$.

4.3 Experiments

In order to compare the different approximate Q-value functions discussed in this chapter, as well as to show the flexibility of the GMAA* algorithm, this section presents several experimental results using Q_{MDP}, Q_{POMDP} and Q_{BG} as heuristic estimates of Q^*. The first experiment provides some qualitative insight in the different Q-value functions we considered. Next, experiments testing the impact on computing optimal policies using MAA*, and on the performance of forward-sweep policy computation are described.

4.3.1 Comparing Q-Value Functions

Before providing a comparison of performance of some of the approximate Q-value functions, this section tries to provide some more insights in their actual values. For the $h = 4$ Dec-Tiger problem, we generated all possible $\vec{\boldsymbol{\theta}}^t$ and the corresponding $\Pr(s_l|\vec{\boldsymbol{\theta}}^t)$, according to (3.1.4). For each of these, the maximal $Q(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$-value is plotted in Figure 4.1. Apart from the three approximate Q-value functions, the optimal value Q^* for each joint action-observation history $\vec{\boldsymbol{\theta}}^t$ that can be realized when using $\boldsymbol{\pi}^*$ is also plotted. Note that it is possible that different $\vec{\boldsymbol{\theta}}^t$ have different optimal values, but induce the same $\Pr(s_l|\vec{\boldsymbol{\theta}}^t)$, as demonstrated in the figure: there are multiple Q^*-values plotted for some $\Pr(s_l|\vec{\boldsymbol{\theta}}^t)$. For the horizon-3 Meeting on a Grid problem we also collected all $\vec{\boldsymbol{\theta}}^t$ that can be visited by the optimal policy, and in Figure 4.2 we again plotted maximal $Q(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$-values. Because this problem has many states, a representation as in Figure 4.1 is not possible. Instead, the $\vec{\boldsymbol{\theta}}$ are ordered according to their optimal value. We can see that the bounds are tight for some $\vec{\boldsymbol{\theta}}$, while for others they can be quite loose. However, when used in the GMAA* framework, their actual performance as a heuristic also depends on their valuation of $\vec{\boldsymbol{\theta}} \in \vec{\boldsymbol{\Theta}}$ not shown by Figure 4.2, namely those that will *not* be visited by an optimal policy: especially when these are overestimated, GMAA* will first examine a sub-optimal branch of the search tree. A tighter upper bound can speed up computation to a very large extent, as it allows the algorithm to

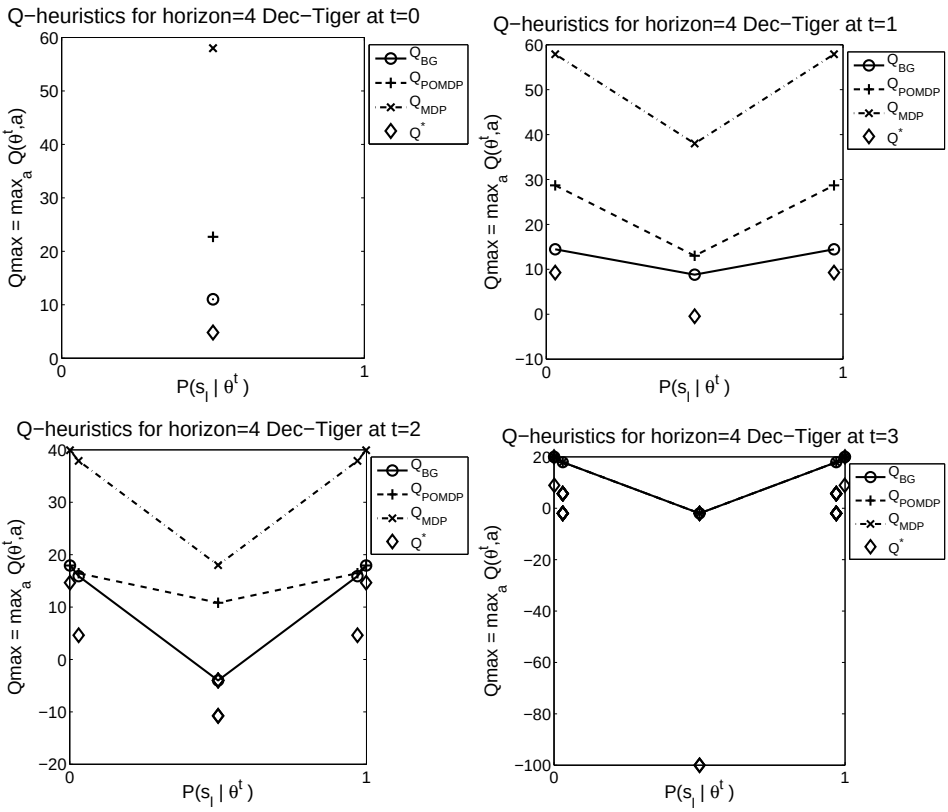


Figure 4.1: Q-values for horizon 4 DEC-TIGER. For each $\vec{\boldsymbol{\theta}}^t$, corresponding to some $\Pr(s_l|\vec{\boldsymbol{\theta}}^t)$, the maximal $Q(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$-value is plotted. Shown are the heuristic $\mathrm{Q}_{\mathrm{MDP}}$, $\mathrm{Q}_{\mathrm{POMDP}}$ and Q_{BG} Q-value functions and Q^* the value function induced by an optimal joint policy $\boldsymbol{\pi}^*$.

prune the policy pool more, reducing the number of Bayesian games that need to be solved. Both figures clearly illustrate that $Q^* \leq \mathrm{Q}_{\mathrm{BG}} \leq \mathrm{Q}_{\mathrm{POMDP}} \leq \mathrm{Q}_{\mathrm{MDP}}$ (see Theorem 4.1).

4.3.2 Computing Optimal Policies

As shown above, the hierarchy of upper bounds $Q^* \leq \mathrm{Q}_{\mathrm{BG}} \leq \mathrm{Q}_{\mathrm{POMDP}} \leq \mathrm{Q}_{\mathrm{MDP}}$ is not just a theoretical construct, but the differences in value specified can be significant for particular problems. This section describes the impact of these differences when applied in MAA*. As Q_{BG}, $\mathrm{Q}_{\mathrm{POMDP}}$ and $\mathrm{Q}_{\mathrm{MDP}}$ are upper bounds to Q^*, MAA* is guaranteed to find the optimal policy when using them as heuristic, however the timing results may differ. All timing results are CPU times with a resolution of 0.01s, and were obtained on 3.4GHz Intel Xeon processors.

Table 4.1 shows the results MAA* obtained on the original DEC-TIGER problem for horizon 3 and 4. It shows for each h the optimal value V^* and for each heuristic

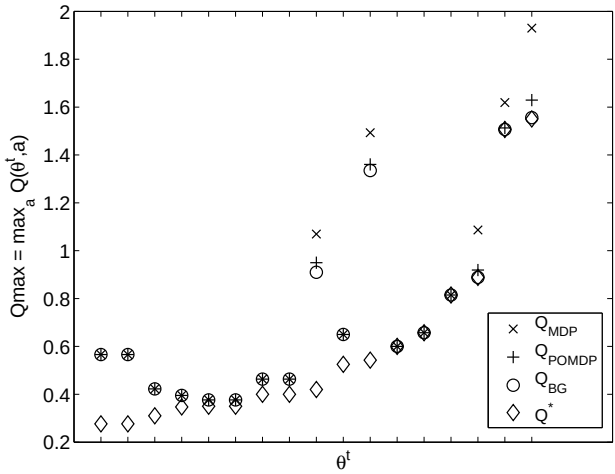


Figure 4.2: Comparison of maximal $Q(\vec{\theta}^t,\mathbf{a})$-values for Meeting on a Grid. We plot the value of all θ^t that can be reached by an optimal policy, ordered according their optimal value.

h	V^*		n_{φ}	T_{GMAA^*}	T_Q
3	5.1908	Q_{MDP}	105,228	0.31 s	< 0.01 s
		Q_{POMDP}	6,651	0.02 s	< 0.01 s
		Q_{BG}	6,651	0.02 s	0.02 s
4	4.8028	Q_{MDP}	37,536,938,118	431,776 s	< 0.01 s
		Q_{POMDP}	559,653,390	5,961 s	0.13 s
		Q_{BG}	301,333,698	3,208 s	0.94 s

Table 4.1: MAA* results for Dec-Tiger.

h	V^*		n_{φ}	T_{GMAA^*}	T_Q
3	5.8402	Q_{MDP}	151,236	0.46 s	< 0.01 s
		Q_{POMDP}	19,854	0.06 s	0.01 s
		Q_{BG}	13,212	0.04 s	0.03 s
4	11.1908	Q_{MDP}	33,921,256,149	388,894 s	< 0.01 s
		Q_{POMDP}	774,880,515	8,908 s	0.13 s
		Q_{BG}	86,106,735	919 s	0.92 s

Table 4.2: MAA* results for Skewed Dec-Tiger.

h	V^*		n_{φ}	T_{GMAA^*}	T_Q
4	3.8900	Q_{MDP}	328,212	3.54 s	< 0.01 s
		Q_{POMDP}	531	< 0.01 s	0.01 s
		Q_{BG}	531	< 0.01 s	0.03 s
5	4.7900	Q_{MDP}	N/A	$> 4.32\text{e}5$ s	< 0.01 s
		Q_{POMDP}	196,883	5.30 s	0.20 s
		Q_{BG}	196,883	5.15 s	0.53 s

Table 4.3: MAA* results for BroadcastChannel.

h	V^*		n_{φ}	T_{GMAA^*}	T_Q
2	0.9100	Q_{MDP}	1,275	< 0.01 s	< 0.01 s
		Q_{POMDP}	1,275	< 0.01 s	< 0.01 s
		Q_{BG}	194	< 0.01 s	< 0.01 s
3	1.5504	Q_{MDP}	29,688,775	81.93 s	< 0.01 s
		Q_{POMDP}	3,907,525	10.80 s	0.15 s
		Q_{BG}	1,563,775	4.44 s	1.37 s

Table 4.4: MAA* results for GridSmall.

the number of partial joint policies evaluated n_{φ}, CPU time spent on the GMAA* phase T_{GMAA^*}, and CPU time spent on calculating the heuristic T_Q. Note that V^* for $h = 4$ is lower than for horizon 3. This is because, like for $h = 3$, the optimal joint policy for $h = 4$ (shown in Figure 4.3) specifies only to open the door at the last stage. This means that the agents will listen one additional time (in comparison to $h = 3$) leading to a better chance of opening the right door, but also to an additional cost of -2 for the listening itself. With respect to the performance of the different heuristics, we see that for $h = 3$ using Q_{POMDP} and Q_{BG} only a fraction of the number of policies are evaluated when compared to Q_{MDP}. This is reflected proportionally in the time spent on GMAA*. For this horizon Q_{POMDP} and Q_{BG} perform the same, but the time needed to compute the Q_{BG} heuristic is as long as the GMAA*-phase, therefore Q_{POMDP} outperforms Q_{BG} here. For $h = 4$, the impact of using tighter heuristics becomes even more pronounced. In this case the computation time of the heuristic is negligible, and Q_{BG} outperforms both, as it is able to prune much more partial joint policies from the policy pool.

Table 4.2 shows results for Skewed Dec-Tiger. For this problem the Q_{MDP} and Q_{BG} results are roughly the same as the original Dec-Tiger problem; for $h = 3$ the timings are a bit slower, and for $h = 4$ they are faster. For Q_{POMDP}, however, we see that for $h = 4$ the results are slower as well and that Q_{BG} outperforms Q_{POMDP} by almost an order of magnitude.

Results for the BroadcastChannel (Table 4.3), GridSmall (Table 4.4) and the FireFighting problem (Table 4.5) are similar. The N/A entry in Table 4.3 indicates the Q_{MDP} was not able to compute a solution within 5 days. For these problems we also see that the performance of Q_{POMDP} and Q_{BG} is roughly equal. For the Meeting on a Grid problem, Q_{BG} yields a significant speedup over Q_{POMDP}.

h	V^*		n_{φ}	T_{GMAA^*}	T_Q
3	−5.7370	Q_{MDP}	446,724	1.58 s	0.56 s
		Q_{POMDP}	26,577	0.08 s	0.21 s
		Q_{BG}	26,577	0.08 s	0.33 s
4	−6.5788	Q_{MDP}	25,656,607,368	309,235 s	0.85 s
		Q_{POMDP}	516,587,229	5,730 s	7.22 s
		Q_{BG}	516,587,229	5,499 s	11.72 s

Table 4.5: MAA* results for FIREFIGHTING $\langle N_H = 3, N_f = 3\rangle$.

$\vec{\mathbf{o}}_{\emptyset} \rightarrow a_{\text{Li}}$
$o_{\text{HL}} \rightarrow a_{\text{Li}}$
$o_{\text{HR}} \rightarrow a_{\text{Li}}$
$\boxed{o_{\text{HL}}, o_{\text{HL}} \rightarrow a_{\text{OR}}}$
$o_{\text{HL}}, o_{\text{HR}} \rightarrow a_{\text{Li}}$
$o_{\text{HR}}, o_{\text{HL}} \rightarrow a_{\text{Li}}$
$\boxed{o_{\text{HR}}, o_{\text{HR}} \rightarrow a_{\text{OL}}}$
$o_{\text{HL}}, o_{\text{HL}}, o_{\text{HL}} \rightarrow a_{\text{Li}}$
$o_{\text{HL}}, o_{\text{HL}}, o_{\text{HR}} \rightarrow a_{\text{Li}}$
$o_{\text{HL}}, o_{\text{HR}}, o_{\text{HL}} \rightarrow a_{\text{Li}}$
$o_{\text{HL}}, o_{\text{HR}}, o_{\text{HR}} \rightarrow a_{\text{Li}}$
$o_{\text{HR}}, o_{\text{HL}}, o_{\text{HL}} \rightarrow a_{\text{Li}}$
$o_{\text{HR}}, o_{\text{HL}}, o_{\text{HR}} \rightarrow a_{\text{Li}}$
$o_{\text{HR}}, o_{\text{HR}}, o_{\text{HL}} \rightarrow a_{\text{Li}}$
$o_{\text{HR}}, o_{\text{HR}}, o_{\text{HR}} \rightarrow a_{\text{Li}}$

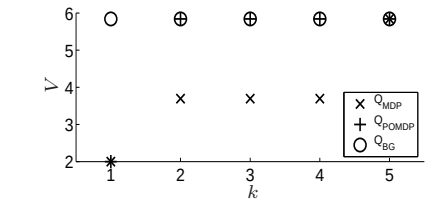
$\vec{\mathbf{o}}_{\emptyset} \rightarrow a_{\text{Li}}$
$o_{\text{HL}} \rightarrow a_{\text{Li}}$
$o_{\text{HR}} \rightarrow a_{\text{Li}}$
$o_{\text{HL}}, o_{\text{HL}} \rightarrow a_{\text{Li}}$
$o_{\text{HL}}, o_{\text{HR}} \rightarrow a_{\text{Li}}$
$o_{\text{HR}}, o_{\text{HL}} \rightarrow a_{\text{Li}}$
$o_{\text{HR}}, o_{\text{HR}} \rightarrow a_{\text{Li}}$
$\boxed{o_{\text{HL}}, o_{\text{HL}}, o_{\text{HL}} \rightarrow a_{\text{OR}}}$
$o_{\text{HL}}, o_{\text{HL}}, o_{\text{HR}} \rightarrow a_{\text{Li}}$
$o_{\text{HL}}, o_{\text{HR}}, o_{\text{HL}} \rightarrow a_{\text{Li}}$
$o_{\text{HL}}, o_{\text{HR}}, o_{\text{HR}} \rightarrow a_{\text{Li}}$
$o_{\text{HR}}, o_{\text{HL}}, o_{\text{HL}} \rightarrow a_{\text{Li}}$
$o_{\text{HR}}, o_{\text{HL}}, o_{\text{HR}} \rightarrow a_{\text{Li}}$
$o_{\text{HR}}, o_{\text{HR}}, o_{\text{HL}} \rightarrow a_{\text{Li}}$
$\boxed{o_{\text{HR}}, o_{\text{HR}}, o_{\text{HR}} \rightarrow a_{\text{OL}}}$

Figure 4.3: Policies found using forward-sweep policy computation (i.e., $k = 1$) for the $h = 4$ DEC-TIGER problem. Left: the policy resulting from Q_{MDP}. Right: the optimal policy as calculated by Q_{POMDP} and Q_{BG}. The framed entries highlight the crucial differences.

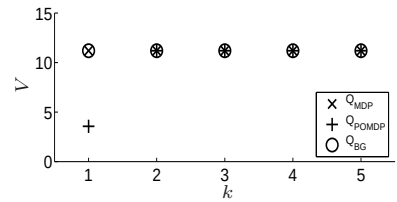
4.3.3 Forward-Sweep Policy Computation and k-GMAA*

The MAA* results described above indicate that the use of a tighter heuristic can yield substantial time savings. In this section, the approximate Q-value functions are used in forward-sweep policy computation. The expectation is that when using a Q-value function that more closely resembles Q^*, the quality of the resulting policy will be higher. We also tested how the quality of the policies computed by k-GMAA* improves when increasing k (the number of best joint BG policies returned by the `Expand` operator). In particular, we tested $k = 1,2,\ldots,5$.

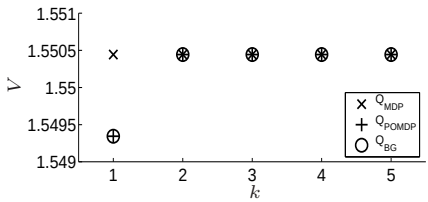
For the DEC-TIGER problem, k-GMAA* with $k = 1$ (and thus also $2 \leq k \leq 5$) found the optimal policy (with $V(\boldsymbol{\pi}^*) = 5.19$) for horizon 3 using all approximate Q-value functions. For horizon $h = 4$, also all different values of k produced the same result for each approximate Q-value function. In this case, however, Q_{MDP} found a policy with expected return of 3.19. Q_{POMDP} and Q_{BG} did find the optimal policy ($V(\boldsymbol{\pi}^*) = 4.80$). Figure 4.3 illustrates the optimal policy (right) and the one found by Q_{MDP} (left). It shows that Q_{MDP} overestimates the value for opening the door in stage $t = 2$.



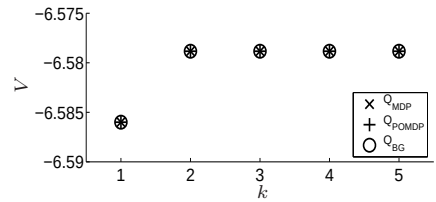
(a) SKEWED DEC-TIGER, $h = 3$.



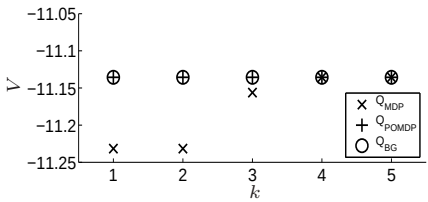
(b) SKEWED DEC-TIGER, $h = 4$.



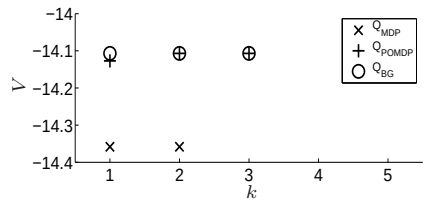
(c) GRIDSMALL, $h = 3$.



(d) FIREFIGHTING $\langle N_H = 3, N_f = 3 \rangle$, $h = 4$.



(e) FIREFIGHTING $\langle N_H = 4, N_f = 3 \rangle$, $h = 3$.



(f) FireFighting $\langle N_H = 4, N_f = 3 \rangle$, $h = 4$.

Figure 4.4: k-GMAA* results for different problems and horizons. The y-axis indicates value of the initial joint belief, while the x-axis denotes k.

For the SKEWED DEC-TIGER problem, different values of k did produce different results. In particular, for $h = 3$ only $\mathrm{Q_{BG}}$ finds the optimal policy (and thus attains the optimal value) for all values of k, as shown in Figure 4.4a. $\mathrm{Q_{POMDP}}$ does find it starting from $k = 2$, and $\mathrm{Q_{MDP}}$ only from $k = 5$. Figure 4.4b shows a somewhat unexpected result for $h = 4$: here for $k = 1$ $\mathrm{Q_{MDP}}$ and $\mathrm{Q_{BG}}$ find the optimal policy, but $\mathrm{Q_{POMDP}}$ doesn't. This clearly illustrates that a tighter approximate Q-value function is not a guarantee for a better joint policy, which is also illustrated by the results for GRIDSMALL in Figure 4.4c.

We also performed the same experiment for two settings of the FIREFIGHTING problem. For $\langle N_H = 3, N_f = 3 \rangle$ and $h = 3$ all Q-value functions found the optimal policy (with value -5.7370) for all k, and horizon 4 is shown in Figure 4.4d. Figures 4.4e and 4.4f show the results for $\langle N_H = 4, N_f = 3 \rangle$. For $h = 4$, $\mathrm{Q_{MDP}}$ did not finish for $k \geq 3$ within 5 days.

It is encouraging that for all experiments k-GMAA* using $\mathrm{Q_{BG}}$ and $\mathrm{Q_{POMDP}}$ with $k \leq 2$ found the optimal policy. Using $\mathrm{Q_{MDP}}$ the optimal policy was also always found with $k \leq 5$, except in horizon 4 DEC-TIGER and the $\langle N_H = 4, N_f = 3 \rangle$ FIREFIGHTING problem. These results seem to indicate that this type of approx-

imation might be likely to produce (near-) optimal results for other domains as well.

4.4 Conclusions

Because calculating the optimal Dec-POMDP Q-value function Q^* is too computationally expensive, we examined approximate Q-value functions that can be calculated more efficiently and we discussed how they relate to Q^*. Section 4.1 covered $\mathrm{Q}_{\mathrm{MDP}}$, $\mathrm{Q}_{\mathrm{POMDP}}$, and Q_{BG}, which was recently proposed by Oliehoek and Vlassis (2007a). Also, using the fact that decreasing communication delays in decentralized systems cannot decrease the expected value we established that $Q^* \leq \mathrm{Q}_{\mathrm{BG}} \leq \mathrm{Q}_{\mathrm{POMDP}} \leq \mathrm{Q}_{\mathrm{MDP}}$. Experimental evaluation indicated that these upper bounds are not just of theoretical interest, but that significant differences exist in the tightness of the various approximate Q-value functions.

Additionally we showed how the approximate Q-value functions can be used as heuristics in a generalized policy search method GMAA*, thereby presenting a unified perspective of forward-sweep policy computation and the recent Dec-POMDP solution techniques of Emery-Montemerlo et al. (2004) and Szer et al. (2005). Finally, an empirical evaluation of GMAA* shows significant reductions in computation time when using tighter heuristics to calculate optimal policies. Also Q_{BG} generally found better approximate solutions in forward-sweep policy computation and the 'k-best joint BG policies' GMAA* variant, or k-GMAA*, although there are no guarantees.

Still, the scalability of GMAA* as discussed here is limited. The bottleneck encountered for the experimental results presented in this chapter is the exact solution of the BGs. For approximate settings, the bottleneck can be alleviated by using approximate BG solvers such as alternating maximization (Emery-Montemerlo et al., 2004). However, this will only provide limited possibility of scaling, since the BGs will grow exponentially with the number of agents and the horizon. The growth induced by the number of agents is addressed in the next chapter. The exponential growth with respect to the horizon is caused by the growth in the number of AOHs and thus the number of types in the BGs and is addressed in Chapter 6.

4.5 Future Work

Future research could further generalize GMAA*, by defining other `Expand` or `Select` operators, with the hope that the resulting algorithms will be able to scale to larger problems. For instance by introducing weights as proposed by Szer et al. (2005). Also it is important to establish bounds on the performance and learning curves of GMAA* in combination with different `Expand` operators and heuristics. A different direction is to experimentally evaluate the use of even tighter heuristics such as Q-value functions for the case of observations delayed by multiple time steps. This research should be paired with methods to efficiently find such Q-value functions. Finally, there is a need for efficient approximate methods for solving the Bayesian games.

Chapter 5

Factored Dec-POMDPs: Exploiting Locality of Interaction

A Dec-POMDP can be represented by a series of Bayesian games (BGs), one for each time step, as introduced in Chapter 3. Subsequently, Chapter 4 showed how this can be used in a policy search method called Generalized MAA* (GMAA*). Still, this method is limited to small problems. Two sources of intractability are the number of agents n and the horizon h, because the size of the BGs grows exponentially with respect to these parameters. In particular, the number of joint policies for a BG for the last stage $t = h-1$, and thus the cost of optimally solving such a BG, is

$$O\Big(|\mathcal{A}_*|^{n(|\mathcal{O}_*|^{h-1})}\Big), \tag{5.0.1}$$

where $\mathcal{A}_*$ and $\mathcal{O}_*$ denote the largest individual action and observation sets. This chapter only focuses on providing scaling with respect to the number of agents n. Techniques to improve scalability in the horizon are the topic of Chapter 6.

Previous work tried to overcome the complexity introduced by the number of agents by superimposing assumptions of independence between agents on the Dec-POMDP model such as transition and observation independence (Becker, Zilberstein, Lesser, and Goldman, 2003; Nair et al., 2005; Varakantham et al., 2007; Kumar and Zilberstein, 2009). In such models it is easy to prove that there is locality of interaction, which can be exploited for their efficient solution. However, such assumptions are very restrictive and preclude many interesting problems.

Still, in many problems the agents may be nearly independent: for instance each agent may only need to interact with a particular neighbor. This chapter formalizes the interaction of several agents under uncertainty as a *factored Dec-POMDP* with an *additively factored immediate reward function* that has *restricted scopes*. I.e., each component of the reward function is only influenced by a subset of agents. Rather than superimposing independence assumptions, a more general analysis of dependencies in factored Dec-POMDPs is presented by analyzing value functions for such models.

The main idea in this chapter is to apply GMAA* to factored Dec-POMDPs, exploiting any independence between agents that may be present. To this end, we replace the regular Bayesian games (BGs) from GMAA* by *collaborative graphical BGs (CGBGs)* that can represent independence between agents and therefore may be specified much more compactly and solved more efficiently. In order to get this idea to work, however, there are a few issues that need to be taken care of.

CGBGs can be represented compactly, but constructing them exactly requires performing exact inference over the space of states and joint histories, whose size is exponential in the number of state factors and the number of agents respectively. Therefore, we will construct them using approximate inference. To subsequently use CGBGs in the solution of factored Dec-POMDPs, a *factored* Q-value function is necessary to serve as payoff function for the CGBGs. An optimal Q-value function Q^* is factored: it can be specified as the sum of multiple components with different scopes. However, these scopes grow when moving from the last stage to the first (i.e., when going back in time towards the initial stage $t = 0$), and thus using Q^* is impractical because it is fully coupled for earlier stages. Also, it is computationally too expensive to compute. Therefore we will consider approximate Q-value functions *given a predetermined scope structure* that specifies the scopes for each stage. Two methods to compute a factored approximation of the Q_{MDP} value function will be introduced. We also propose a third way of computing an approximate Q-value function, namely *transfer planning (TP)* that uses an approximate (e.g., Q_{MDP}) value function for a smaller source problem that involves a smaller number of agents. Given the factored payoff functions, the CGBGs are fully specified. We will discuss how they can be solved efficiently using either nonserial dynamic programming or a message passing algorithm called Max-Plus. Finally we combine all these elements in a family of algorithms which we refer to as Factored GMAA*.

Organization of Chapter

This chapter is organized as follows. First, Section 5.1 introduces factored Dec-POMDPs with additively factored immediate reward functions. Section 5.2 shows that for such models, there is an optimal factored value function that has scopes that grow when moving from the last stage to the first.

Section 5.3 shows how a factored Dec-POMDP can be represented by a series of collaborative graphical Bayesian games (CGBG). When using Q^* as a payoff function for the BGs, this modeling is exact, but the earlier stages will be fully coupled. To avoid this problem, Section 5.4 proposes to compute approximate factored Q-value functions given a predetermined scope structure. The efficient solution of CGBGs is described in Section 5.5.

Given the complete specification of the CGBGs and a method to solve them, GMAA* policy search methods can be employed, leading to a family of methods we refer to as Factored GMAA* described in Section 5.6. It introduces the GMAA*-ELSI (Exploiting Last-Stage Independence) algorithm that can exploit locality of interaction in the last stage to speed up computation of optimal solutions and Factored FSPC and Factored k-GMAA* that are scalable and efficient

approximate methods.

Section 5.7 reports on the empirical evaluation on two problem domains: the FireFightingGraph problem and a problem domain called Aloha. This evaluation shows that GMAA*-ELSI can significantly speed up the computation of optimal solutions and that the approximate Factored GMAA* methods are efficient and scale well with respect to the number of agents, but that the quality of the found solutions depends very much on the used heuristic. Although identifying more and better heuristics remains future work, we show that one of proposed heuristics is able to compute good joint policies for up to 1000 agents.

5.1 Factored Dec-POMDPs

This section provides the necessary background, by introducing the formal model of factored Dec-POMDPs and illustrating it with an example.

5.1.1 The Formal Model

The notion of factored Dec-POMDP extends the definition of the regular Dec-POMDP given by Definition 2.6.

Definition 5.1 (Factored Dec-POMDP)**.** A *factored* Dec-POMDP is identical to a normal Dec-POMDP but has a factored state space $\mathcal{S} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_{|\mathcal{X}|}$. That is, $\mathcal{S}$ is spanned by $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_{|\mathcal{X}|}\}$ a set of state variables, or *factors.* A state corresponds to an assignment of values for all factors $s = \langle x_1, \ldots, x_{|\mathcal{X}|} \rangle$.

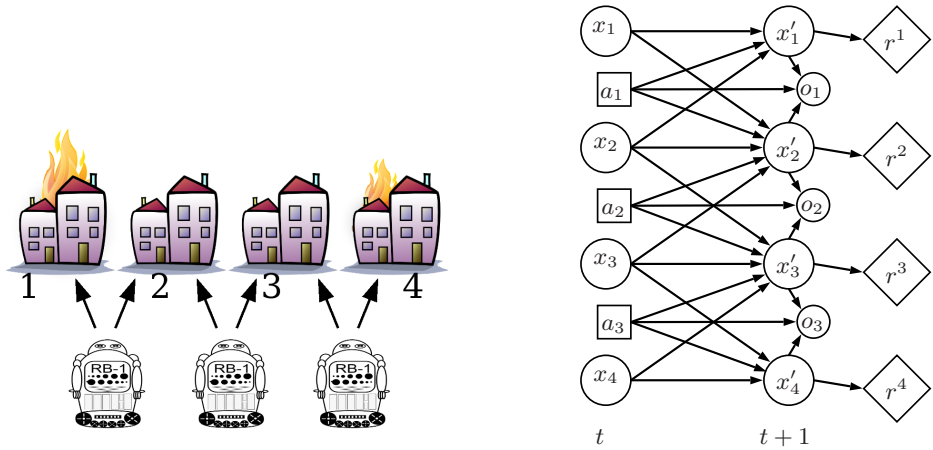
For instance, FireFighting presented in Section 2.3.1 has a factored state space and thus is a factored Dec-POMDP. This chapter will also consider actions, observations and policies for subsets of agents. For instance, $\mathbf{a}_{\mathcal{J}}$ denotes the joint action of a subset $\mathcal{J} \subseteq \mathcal{D}$ of the set of all agents $\mathcal{D}$.

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) (Boutilier et al., 1999). In such a DBN, arrows represent causal influence and each node with incoming edges has a conditional probability table (CPT) associated with it. Although the size of these CPTs is exponential in the number of parents, the parents are typically a subset of all state factors and actions, leading to a model exponentially smaller than a flat model.

Similarly, the reward function can often be compactly represented by exploiting *additive separability,* which means that it is possible to decompose the reward function into the sum of 'smaller' *local* reward functions:

$$R = R^1 + \cdots + R^\rho. \tag{5.1.1}$$

Here 'smaller' means that the functions are defined over a smaller number of state and action variables, i.e., the *scope* of these functions is smaller. We can represent the smaller functions in the DBN. Such a DBN that includes reward nodes is also called an *influence diagram (ID)* (Howard and Matheson, 1984/2005; Boutilier



(a) An illustration FFG.

(b) The DBN that represents FFG.

Figure 5.1: The FIREFIGHTINGGRAPH problem. A DBN can be used to represent the transition, observation and reward function.

et al., 1999). The reward nodes have conditional reward tables (CRTs) associated with them that represent the smaller reward functions. The local reward functions should not be confused with *individual* reward functions in a system with self-interested agents, such as partially observable stochastic games (Hansen et al., 2004) and (graphical) BGs (Singh et al., 2004b; Soni et al., 2007). In such models agents compete to maximize their individual reward functions, while we consider agents that collaborate to maximize the sum of local payoff functions.

Decision trees (Boutilier et al., 1999; Boutilier, Dearden, and Goldszmidt, 2000) or algebraic decision diagrams (Hansen and Feng, 2000; Poupart, 2005) can be used to further reduce the size of representation of both CPTs and CRTs by exploiting *context specific independence*. E.g., the value of some factor x_i may be of no influence when some other factor x_j has a particular value. These further enhancements are not considered in this thesis.

5.1.2 An Example: The FireFightingGraph Problem

To illustrate the model we introduce the FIREFIGHTINGGRAPH (FFG) problem as a running example. It is a modification of FIREFIGHTING that restricts each agent to fight fire at one of the two houses in its vicinity. FFG models a team of n firefighters that have to extinguish fires in a row of $n_H = n+1$ houses. We assume $n = 3, n_H = 4$ as illustrated in Figure 5.1a. At every time step, each agent i can choose to fight fires at house i or $i+1$. For instance, agent 2's possible actions are H_2 and H_3. As in the regular FIREFIGHTING problem, each agent can observe only whether there are flames, $o_i = F$, or not, $o_i = N$, at its location. Each house H is characterized by a fire level x_H, an integer parameter in $[0,N_f)$, where a level of 0 indicates the house is not burning. A state in FFG is an assignment of fire

levels $s = \langle x_1,x_2,x_3,x_4 \rangle$. Initially, the fire level x_H of each house is drawn from a uniform distribution.

Figure 5.1b shows the DBN for the problem. The transition and observation model are similar to that of the regular FIREFIGHTING problem as described in Section 2.3.1, although the maximum number of agents at a house is now limited to 2.

The rewards are also similar, but here we retain the structure of the immediate reward function. The agents receive a reward of $-x_H$ for each house H. In particular, for each house $1 \leq H \leq 4$ the rewards are specified by the fire levels at the next time step $r^H(x'_H) = -x'_H$. We can reduce these rewards to ones of the form $R^H(s,\mathbf{a})$ as in Definition 2.6 by taking the expectation over x'_H. For instance, for house 1

$$R^1(\mathbf{x}_{\{1,2\}},a_1) = \sum_{x'_1} \Pr(x'_1|\mathbf{x}_{\{1,2\}},a_1) r^1(x'_1), \tag{5.1.2}$$

where $\mathbf{x}_{\{1,2\}}$ denotes $\langle x_1,x_2 \rangle$. This formulation is possible because, as Figure 5.1 shows, x_1,x_2 and a_1 are the only variables that influence the probability of x'_1. In a similarly way, the other local reward functions are given by $R^2(\mathbf{x}_{\{1,2,3\}},\mathbf{a}_{\{1,2\}})$, $R^3(\mathbf{x}_{\{2,3,4\}},\mathbf{a}_{\{2,3\}})$ and $R^4(\mathbf{x}_{\{3,4\}},a_3)$.

5.1.3 Independence Assumptions

Because of the negative complexity results for Dec-POMDPs, much research has considered special cases that are easier to solve (Becker et al., 2003; Becker, Zilberstein, and Lesser, 2004a; Becker et al., 2005; Shen, Becker, and Lesser, 2006; Nair et al., 2005; Wu and Durfee, 2006; Kim, Nair, Varakantham, Tambe, and Yokoo, 2006; Varakantham et al., 2007; Marecki, Gupta, Varakantham, Tambe, and Yokoo, 2008; Kumar and Zilberstein, 2009). This research has considered models related to factored Dec-POMDPs but that impose stricter assumptions. We will not consider these additional assumptions in our approach, but treat them here to provide the necessary background.

First let us introduce the notion of an *agent-wise factored state space*, that is, a state consists of an instantiation of an *individual* state s_i for each agent i. Also, there is an 'external' state factor s_0 that cannot be influenced by any of the agents: its transitions are specified as

$$\Pr(s'_0|s_0).$$

The complete state is given by $s = \langle s_0,s_1,\ldots,s_n \rangle$. Although s_0 cannot be influenced by the agents, it may influence the local transitions and observations of each agent. Formally, the local state $\hat{s}_i$ of an agent i is defined as $\hat{s}_i \equiv \langle s_0,s_i \rangle$. Now, a Dec-POMDP is said to be transition independent, when the transition probabilities are factored as

$$\Pr(s'|s,\mathbf{a}) = \Pr(s'_0|s_0) \prod_{i \in \mathcal{D}} \Pr(s'_i|\hat{s}_i,a_i,s'_0). \tag{5.1.3}$$

Similarly, observation independence is expressed by

$$\Pr(\mathbf{o}|\mathbf{a},s') = \prod_{i\in\mathcal{D}} \Pr(o_i|a_i,\hat{s}'_i) \tag{5.1.4}$$

and reward independence can be expressed by

$$R(s,\mathbf{a}) = \sum_{i\in\mathcal{D}} R^i(\hat{s}_i,a_i). \tag{5.1.5}$$

When all three types of independence hold, we are simply dealing with separate POMDPs (or MDPs in the case of a Dec-MDP[1]). However, if one type of independence is violated, the problem is non-trivial. Some special cases considered in literature are

- Event-driven Dec-MDPs (Becker et al., 2004a; Witwicki and Durfee, 2009),
- Transition- and observation-independent (TOI) Dec-MDPs (Becker et al., 2003, 2004b; Wu and Durfee, 2006), and
- ND-POMDPs (Nair et al., 2005; Kim et al., 2006; Varakantham et al., 2007; Marecki et al., 2008; Kumar and Zilberstein, 2009).

Event driven Dec-MDPs are *locally observable* (i.e., the agents can observe their local state $\hat{s}_i$) and reward independent. No full transition and observation independence are assumed, but the transitions are assumed to have special structure.

For the class of TOI-Dec-MDPs, Becker et al. (2004b) prove that the complexity class is NP-complete. Note, however, that the assumption of TOI is a strong assumption: it requires each agent to have its own local state space that cannot be influenced whatsoever by another agent. In effect, (because it can be shown that a TOI-Dec-MDP is locally observable) a TOI-Dec-MDP is equivalent to n separated MDPs that are coupled only through the joint reward function. Such assumptions preclude many interesting problems, such as two robots collaborating to transport an item.

Nair et al. (2005) and subsequent work consider a slightly more general model, the *networked distributed POMDP (ND-POMDP),* which can best be understood as a transition and observation independent Dec-POMDP with additively separated rewards. Using the structure of the additive immediate reward function Nair et al. define an interaction (hyper-)graph. In this graph, each local reward function is a (hyper-) edge $e \in \mathcal{E}$, connecting two or more agents (nodes) $e = \{i,j,\dots\} \subseteq \mathcal{D}$. Let $\mathcal{N}_i$ denote the neighbors of agent i, including i itself, in this graph. Nair et al. show that in this setting the value can be decomposed into local value functions

$$V(\boldsymbol{\pi}) = \sum_{e\in\mathcal{E}} V^e(\boldsymbol{\pi}_e), \tag{5.1.6}$$

[1] As explained in Section 2.8.1 Dec-MDP is a Dec-POMDP that is *jointly observable*, i.e., the joint observation identifies the state.

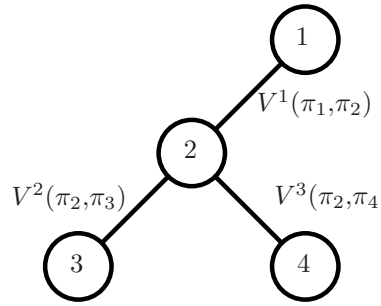


Figure 5.2: An example interaction graph with 4 agents. The planning problem can be interpreted as a DCOP. The structure can be exploited by, for instance, SPIDER (Varakantham et al., 2007) in which the agents cycle through their individual policies in a depth-first manner.

where $\boldsymbol{\pi}_e = \langle \pi_{e1}, \ldots, \pi_{e|e|} \rangle$ is the profile of individual policies of agents that participate in edge e, and define the *local neighborhood utility* of an agent i as the expected return for all the edges that contain agent i:

$$V(\boldsymbol{\pi}_{\mathcal{N}_i}) = \sum_{e \text{ s.t. } i \in e} V^e(\boldsymbol{\pi}_e). \tag{5.1.7}$$

Subsequently, they show that when an agent $j \notin \mathcal{N}_i$ changes its policy, $V(\boldsymbol{\pi}_{\mathcal{N}_i})$ is not affected, a property they refer to as *locality of interaction*.

It is this property that allows a reformulation of the planning problem as a *distributed constraint optimization problem (DCOP)* (Liu and Sycara, 1995; Yokoo, 2001; Modi, Shen, Tambe, and Yokoo, 2005), as is illustrated in Figure 5.2: in each of the nodes (representing agents) a policy is chosen, the goal is to select a joint policy such that the sum of local value functions is maximized. The sparsity in the graph can be exploited for more efficient computation. This chapter proposes to exploit independence between agents in a similar way as described above, but without assuming transition and observation independence.

Other special cases that have been considered are, for instance, goal oriented Dec-POMDPs (Goldman and Zilberstein, 2004), Dec-MDPs with time and resource constraints (Beynier and Mouaddib, 2005, 2006; Marecki and Tambe, 2007) and agent-wise factored Dec-MDPs with local interactions (Spaan and Melo, 2008). Like the rest of the models described in this subsection, though, these models also have limited applicability due to the assumptions they make. Therefore we will not further consider these models any further in this chapter.

5.2 Value Functions for Factored Dec-POMDPs

This section shows that even in the more general setting without transition and observation independence, value functions can still be decomposed by analyzing the independence that is present. In particular, the value can be decomposed into ρ local value functions (corresponding to the ρ local reward functions) involving

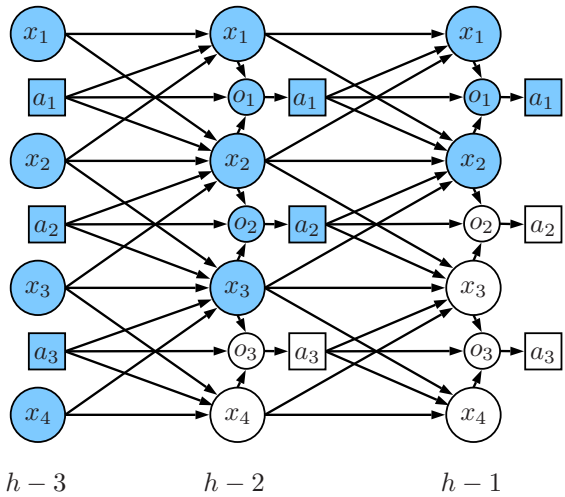


Figure 5.3: Illustration of the interaction between agents and environment over time in FFG. In contrast to Figure 5.1b, which represents the transition and observation model using abstract time steps t and $t+1$, this figure represents the last 3 stages of a decision problem. Also the rewards are omitted in this figure. The scope of Q^1, given by (5.1.2), is illustrated by shading and increases when going back in time.

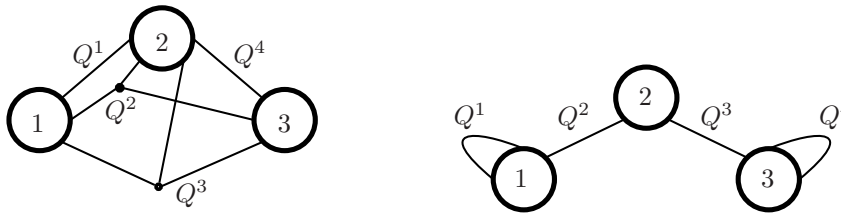
only a subset of agents (although this subset may grow with each stage further from the last stage $t = h-1$). Such a decomposed value function is called *factored.*

5.2.1 Scope, Scope Backup and Interaction Graphs

As an example of a factored Q-value function, we consider the last stage of FFG. By definition, Q^{h-1} is equal to the immediate reward function, and thus can be decomposed in 4 local Q-value functions $Q = Q^1 + \cdots + Q^4$, where each Q^e is defined over the same subset of variables as R^e. This subset of variables is called the *scope* of Q^e. In order to exploit independence between agents, we discriminate between the variables that pertain to state factors and those that pertain to agents (i.e., actions and observations).

Definition 5.2 (Scope)**.** Let $\mathcal{I} \subseteq \{1,\ldots,|\mathcal{X}|\}$ be a subset of state factor indices, and let $\mathcal{J} \subseteq \{1,\ldots,n\}$ be a subset of agent indices, and let us write $\mathcal{X}_\mathcal{I} = \times_{i\in\mathcal{I}}\mathcal{X}_i$, $\mathcal{A}_\mathcal{J} = \times_{i\in\mathcal{J}}\mathcal{A}_i$, $\vec{\boldsymbol{\Theta}}^t_\mathcal{J} = \times_{i\in\mathcal{J}}\vec{\Theta}^t_i$. Then a function $q : \mathcal{X}_\mathcal{I} \times \vec{\boldsymbol{\Theta}}^t_\mathcal{J} \times \mathcal{A}_\mathcal{J} \to \mathbb{R}$ has *state factor scope* $\mathbb{X}(q) = \mathcal{I}$ and *agent scope* $\mathbb{A}(q) = \mathcal{J}$. We also use $\mathbb{S}(q) \equiv \langle \mathbb{X}(q), \mathbb{A}(q)\rangle$ to denote the combined scope of q.

This definition has the most general form of value function considered in this chapter, involving state-factors, action-observation histories and actions. It also applies when not all of these elements are used. For example, the function $R^1(\mathbf{x}_{\{1,2\}}, a_1)$ has scope $\mathbb{X}(R^1) = \{1,2\}$ and $\mathbb{A}(R^1) = \{1\}$. Note also the difference between the scope and domain of a function: the domain of the same function R^1 is $\{\langle 0,0,H_0\rangle, \langle 0,0,H_1\rangle, \ldots, \langle N_f-1, N_f-1, H_1\rangle\}$, the set of tuples $\langle x_1, x_2, a_1\rangle$. So while



(a) $t = h - 2$. **(b)** $t = h - 1$.

Figure 5.4: Interaction graphs for the last two stages of the FFG problem.

the scope is the set of variables of influence, the domain is the set of (tuples of) values those variables can take.

Now we can formalize the e-th component of the Q-value function for joint policy $\boldsymbol{\pi}$ of the last stage $h-1$ as follows[1]

$$Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{h-1}_e,\vec{\boldsymbol{\theta}}^{h-1}_e,\mathbf{a}^{h-1}_e) \equiv R^e(\mathbf{x}^{h-1}_e,\mathbf{a}^{h-1}_e), \qquad e = 1,\ldots,\rho, \tag{5.2.1}$$

where, even though it is not needed, $\vec{\boldsymbol{\theta}}^{h-1}_e$ is included in $Q^e_{\boldsymbol{\pi}}$ because the agent scope is not empty. Moreover, as we will explain in a moment, the Q-value functions for earlier stages typically will be dependent on $\vec{\boldsymbol{\theta}}_e$. For instance, in the FFG problem $Q^1_{\boldsymbol{\pi}}(\mathbf{x}_{\{1,2\}},\vec{\theta}^{h-1}_1,a_1) \equiv R^1(\mathbf{x}_{\{1,2\}},a_1)$ of which the scope is shown in Figure 5.3 at $h-1$.

As in Section 5.1.3, this Q-value function can be used to define an *interaction hyper-graph* $G = \langle \mathcal{D},\mathcal{E}\rangle$. In this graph, the nodes correspond to agents and the ρ hyper-edges $e \in \mathcal{E}$ (edges that can involve more than two nodes) correspond to the ρ local Q-value functions $Q^{e,h-1}$ for stage $h-1$.[2] In particular, an edge e connects $\mathbb{A}(Q^{e,h-1})$, the agents involved in $Q^{e,h-1}$. For example, the interaction hyper-graph for the last two stages of the FFG problem is shown in Figure 5.4. It is the independence represented by sparsity of such interaction graphs that we will exploit to reduce computation costs later on.

Because we do not assume transition and observation independence, the scopes of Q^{h-2}, the Q-value function at $h-2$, will typically be larger than those for Q^{h-1} as given by (5.2.1). In particular, the scopes at $h-2$ should include all factors of influence, i.e., all state factors and agents' actions that can influence the variables in the scopes at $h-1$. We formalize this intuition by trying to write down the Q-value function of $\boldsymbol{\pi}$ at $t = h-2$ as follows. Let us write $\vec{\boldsymbol{\theta}}^{h-1}_e = (\vec{\boldsymbol{\theta}}^{h-2}_e,\mathbf{a}^{h-2}_e,\mathbf{o}^{h-1}_e)$ for the AOH-profile of the agents in $Q^{e,h-1}_{\boldsymbol{\pi}}$. We then have that $Q^{e,h-2}_{\boldsymbol{\pi}}$ is the sum of the immediate reward and the expected future reward:

$$Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{h-2}_{\mathcal{I}},\vec{\boldsymbol{\theta}}^{h-2}_{\mathcal{J}},\mathbf{a}^{h-2}_{\mathcal{J}}) = R^e(\mathbf{x}^{h-2}_e,\mathbf{a}^{h-2}_e) + \sum_{\mathbf{x}^{h-1}_e}\sum_{\mathbf{o}^{h-1}_e} \Pr(\mathbf{x}^{h-1}_e,\mathbf{o}^{h-1}_e|\cdot) Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{h-1}_e,\vec{\boldsymbol{\theta}}^{h-1}_e,\boldsymbol{\pi}_e(\vec{\boldsymbol{\theta}}^{h-1}_e)) \tag{5.2.2}$$

[1] We slightly abuse notation: e denotes both the concerning state factor scope $\mathbb{X}(R^e)$ and the agent scope $\mathbb{A}(R^e)$ of the e-th component of the function under concern. In general it should be clear from the context what scope is meant.

[2] We omit the stage index to Q-functions if it is implicit in their arguments, as in (5.2.1).

where $\vec{\boldsymbol{\theta}}_e^{h-2}$,$\mathbf{a}_e^{h-2}$ are specified by $\vec{\boldsymbol{\theta}}_{\mathcal{J}}^{h-2}$,$\mathbf{a}_{\mathcal{J}}^{h-2}$ and where $(\cdot)$ represents any factors of influence at stage $h-2$. Clearly, $\mathcal{I}$ and $\mathcal{J}$ should include those factors of influence and therefore these sets change over time. We now formalize these factors of influence by defining the scope backup.

Definition 5.3 (Scope backup)**.** Given a set $\mathcal{K}$ of particular state factors and observations of particular agents $\mathcal{K} \subseteq \left\{\mathcal{X}'_1,\ldots,\mathcal{X}'_{|\mathcal{X}|},\mathcal{O}_1,\ldots,\mathcal{O}_n\right\}$, the scope backup operator $\Gamma(\mathcal{K})$ returns the subset of variables $\left\{\mathcal{X}_1,\ldots,\mathcal{X}_{|\mathcal{X}|},\mathcal{A}_1\ldots,\mathcal{A}_n\right\}$ (i.e., from the left side of the DBN) that are ancestors of variables in $\mathcal{K}$. We discriminate the state-factor component and the agent component of the scope backup:

$$\Gamma^{\mathbb{X}}(\mathcal{K}) = \{j \mid \exists_{Y\in\mathcal{K}} \text{ s.t. } \mathcal{X}_j = \text{ancestor}(Y)\},$$

$$\Gamma^{\mathbb{A}}(\mathcal{K}) = \{j \mid \exists_{Y\in\mathcal{K}} \text{ s.t. } \mathcal{A}_j = \text{ancestor}(Y)\}.$$

For example, in FFG the probability of $\mathbf{x}_{\{1,2\}}^{h-1}$,$o_1^{h-1}$ depends only on $\mathbf{x}_{\{1,2,3\}}^{h-2}$ and actions $\mathbf{a}_{\{1,2\}}^{h-2}$ as shown at $h-2$ in Figure 5.3. When the policy is fixed, the probability of actions $\mathbf{a}_{\{1,2\}}^{h-2}$ is determined by $\mathbf{o}_{\{1,2\}}^{h-2}$. In turn, it is necessary to determine $\Pr(\mathbf{x}_{\{1,2,3\}}^{h-2},\mathbf{o}_{\{1,2\}}^{h-2}|\cdot)$. For this, we need to consider all variables of influence at stage $h-3$. This set encompasses all state factors and agents, and therefore the scope of $Q_{\boldsymbol{\pi}}^{1,h-3}$ includes all of these, as indicated in Figure 5.3.

5.2.2 Decomposition of Value Functions

This section shows that without assuming transition and observation independence, we can still decompose the value function. Recall that (2.5.6) expresses $\Pr(s^t,\vec{\boldsymbol{\theta}}^t|\boldsymbol{b}^0,\boldsymbol{\varphi}^t)$ the probability over states and AOHs given a followed (past) joint policy. When interested in some arbitrary scopes of state factors (f) and agents (g), it is possible to marginalize as follows:

$$\Pr(\mathbf{x}_f^t,\vec{\boldsymbol{\theta}}_g^t|\boldsymbol{b}^0,\boldsymbol{\varphi}^t) = \sum_{\mathbf{x}_{\bar{f}}^t}\sum_{\vec{\boldsymbol{\theta}}_{\bar{g}}^t} \Pr(\mathbf{x}_f^t,\mathbf{x}_{\bar{f}}^t,\vec{\boldsymbol{\theta}}_g^t,\vec{\boldsymbol{\theta}}_{\bar{g}}^t|\boldsymbol{b}^0,\boldsymbol{\varphi}^t), \tag{5.2.3}$$

Lemma 5.1 (Decomposition last-stage V)**.** *The value function $V^{h-1}(\boldsymbol{\pi})$ for the last stage $t=h-1$ of a finite-horizon factored Dec-POMDP with additive rewards is factored. It can be decomposed as*

$$V^{h-1}(\boldsymbol{\pi}) = \sum_{e\in\mathcal{E}} V^{e,h-1}(\boldsymbol{\pi}) = \sum_{e\in\mathcal{E}}\sum_{\mathbf{x}_e^{h-1}}\sum_{\vec{\boldsymbol{\theta}}_e^{h-1}} \Pr(\mathbf{x}_e^{h-1},\vec{\boldsymbol{\theta}}_e^{h-1}|\boldsymbol{b}^0,\boldsymbol{\pi}) Q_{\boldsymbol{\pi}}^e(\mathbf{x}_e^{h-1},\vec{\boldsymbol{\theta}}_e^{h-1},\boldsymbol{\pi}_e(\vec{\boldsymbol{\theta}}_e^{h-1})). \tag{5.2.4}$$

Proof. Similar to (2.5.5), the expected value for the last stage can be written as

$$V^{h-1}(\boldsymbol{\pi}) = \sum_{\vec{\boldsymbol{\theta}}^{h-1}}\sum_{s} \Pr(s,\vec{\boldsymbol{\theta}}^{h-1}|\boldsymbol{b}^0,\boldsymbol{\pi}) R(s,\boldsymbol{\pi}(\vec{\boldsymbol{\theta}}^{h-1})).$$

Filling out the definition of additive rewards (5.1.1), using (5.2.1) and swapping the summations yields (5.2.4). □

Such a decomposition is possible for every stage t, as illustrated by the following theorem. The remainder of this section uses more explicit notation for scopes than other parts of the chapter: the state factor and agent scope of the Q-value function under concern are denoted $\mathbb{X}_e$,$\mathbb{A}_e$. That is, $\mathbb{X}_e \equiv \mathbb{X}(Q^{e,t})$ and $\mathbb{A}_e \equiv \mathbb{A}(Q^{e,t})$. Similar shorthands $\mathbb{X}'_e$,$\mathbb{A}'_e$ are used for the next-stage Q-values. For the local immediate reward functions the scope should be clear, so they are simply written as $R^e(\mathbf{x}^t_e,\mathbf{a}_e)$.

Theorem 5.1 (Decomposition of $V^t(\boldsymbol{\pi})$)**.** *Given an additively factored immediate reward function, the value $V^t(\boldsymbol{\pi})$ of a finite-horizon factored Dec-POMDP is decomposable for any t. That is, for any joint policy $\boldsymbol{\pi}$ the value function is factored. $V^t(\boldsymbol{\pi})$ is defined as*

$$V^t(\boldsymbol{\pi}) = \sum_{e\in\mathcal{E}} V^{e,t}(\boldsymbol{\pi}) = \sum_{e\in\mathcal{E}} \sum_{\mathbf{x}^t_{\mathbb{X}_e}} \sum_{\vec{\boldsymbol{\theta}}^t_{\mathbb{A}_e}} \Pr(\mathbf{x}^t_{\mathbb{X}_e},\vec{\boldsymbol{\theta}}^t_{\mathbb{A}_e}|\boldsymbol{b}^0,\boldsymbol{\pi}) Q^e_{\boldsymbol{\pi}}(\mathbf{x}^t_{\mathbb{X}_e},\vec{\boldsymbol{\theta}}^t_{\mathbb{A}_e},\boldsymbol{\pi}_{\mathbb{A}_e}(\vec{\boldsymbol{\theta}}^t_{\mathbb{A}_e})) \tag{5.2.5}$$

where, using shorthand notation $\Gamma^{\mathbb{X}} = \Gamma^{\mathbb{X}}(\mathbf{x}^{t+1}_{\mathbb{X}'_e} \cup \mathbf{o}^{t+1}_{\mathbb{A}'_e})$ and $\Gamma^{\mathbb{A}} = \Gamma^{\mathbb{A}}(\mathbf{x}^{t+1}_{\mathbb{X}_e} \cup \mathbf{o}^{t+1}_{\mathbb{A}_e})$ to denote the backup scopes and $\mathbb{X}_e \equiv \mathbb{X}(R^e) \cup \Gamma^{\mathbb{X}}$ and $\mathbb{A}_e \equiv \mathbb{A}(R^e) \cup \Gamma^{\mathbb{A}} \cup \mathbb{A}'_e$ to denote the scopes of $Q^{e,t}_{\boldsymbol{\pi}}$,

$$Q^e_{\boldsymbol{\pi}}(\mathbf{x}^t_{\mathbb{X}_e},\vec{\boldsymbol{\theta}}^t_{\mathbb{A}_e},\mathbf{a}_{\mathbb{A}_e}) = R^e(\mathbf{x}^t_e,\mathbf{a}_e) + \sum_{\mathbf{x}^{t+1}_{\mathbb{X}'_e}} \sum_{\mathbf{o}^{t+1}_{\mathbb{A}'_e}} \Pr(\mathbf{x}^{t+1}_{\mathbb{X}'_e},\mathbf{o}^{t+1}_{\mathbb{A}'_e}|\mathbf{x}^t_{\Gamma^{\mathbb{X}}},\mathbf{a}_{\Gamma^{\mathbb{A}}}) Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{t+1}_{\mathbb{X}'_e},\vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e},\boldsymbol{\pi}_{\mathbb{A}'_e}(\vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e})). \tag{5.2.6}$$

Proof. The proof is listed in the appendix. □

This theorem is important because it implies that an optimal joint policy $\boldsymbol{\pi}^*$ induces a factored optimal value function. And, as shall be explained in Section 5.3, such a factored optimal value function can in turn be used to construct $\boldsymbol{\pi}^*$.

Theorem 5.1 also states that the scope of $Q^{e,t}_{\boldsymbol{\pi}}$ should contain $\mathbf{x}^t_{\Gamma^{\mathbb{X}}}$,$\mathbf{a}_{\Gamma^{\mathbb{A}}}$ such that the probability of $\mathbf{x}^{t+1}_{\mathbb{X}'_e}$,$\mathbf{o}^{t+1}_{\mathbb{A}'_e}$ at the next stage is defined. This means that the scopes of $Q^{e,t}_{\boldsymbol{\pi}}$ as given by (5.4.1) typically grow, as is clear from the definition of $\mathbb{X}_e$,$\mathbb{A}_e$. The definition of $\mathbb{A}(Q^{e,t})$ implies that the agent scope is non-decreasing when going back in time.[1] This means that the interaction graphs for earlier stages are denser, as shown in Figure 5.4. In contrast, the state scope is not necessarily non-decreasing. However, it does also depend on the agent scope through the backup $\Gamma^{\mathbb{X}}(\mathbf{x}^{t+1}_{\mathbb{X}'_e} \cup \mathbf{o}^{t+1}_{\mathbb{A}'_e})$. As such the value functions for earlier stages will typically have larger state factor scopes too.

[1] Note that it is necessary that $\mathbb{A}'_e \subseteq \mathbb{A}_e$, otherwise $\vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e}$ can be undefined.

Example 5.1 (Scopes of FFG). For instance, in FFG for stage $h-2$, the scope of the value function corresponding to reward function 1 is given by

$$\begin{aligned}\mathbb{X}(Q_{\boldsymbol{\pi}}^{1,h-2}) &= \mathbb{X}(R^1) \cup \Gamma^{\mathbb{X}}(\{x_1,x_2,o_1\}) \\ &= \{x_1,x_2\} \cup \{x_1,x_2,x_3\} = \{x_1,x_2,x_3\}\end{aligned}$$

$$\begin{aligned}\mathbb{A}(Q_{\boldsymbol{\pi}}^{1,h-2}) &= \mathbb{A}(R^1) \cup \Gamma^{\mathbb{A}}(\{x_1,x_2,o_1\}) \cup \mathbb{A}(Q_{\boldsymbol{\pi}}^{1,h-1}) \\ &= \{1\} \cup \{1,2\} \cup \{1\} = \{1,2\}\end{aligned}$$

We can identify the scopes for the other 3 ($e = 2,\ldots,4$) value functions $Q_{\boldsymbol{\pi}}^{e,h-2}$ in a similar way. The agent scopes define a new interaction graph for stage $h-2$ of the FFG problem, as illustrated in Figure 5.4. The figure clearly shows that Q^2 and Q^3 both involve all agents, which means that at this stage there is no independence between agents.

When one or more components becomes fully coupled, technically, the value function still is factored. However, at this point the factorization will no longer provide any benefits and therefore at this point the components can be collapsed to form a non-factored value function.

5.2.3 Locality of Interaction

When assuming transition and observation independence, as discussed in Section 5.1.3, the probability in (5.2.6) reduces easily: each variable s_i^{t+1} (representing a local state for agent i) is dependent only on s_i^t (its own value at the previous state), and each o_i is dependent only on s_i^{t+1}. This means that the scope does not expand and, as a result, the interaction graph for such settings is stationary.

In our more general case, such a notion of locality of interaction over full-length policies is not properly defined, simply because the interaction graph and hence agent i's neighborhood can be different at every stage. However, at a particular stage t, we can define such a notion. Let us decompose the joint policy $\boldsymbol{\pi} = (\boldsymbol{\varphi}^t,\boldsymbol{\delta}^t,\boldsymbol{\psi}^t)$ in the past joint policy $\boldsymbol{\varphi}^t$ that has been followed, the joint decision rule $\boldsymbol{\delta}^t$ for stage t and a future joint policy $\boldsymbol{\psi}^t$.

Corollary 5.1 (Locality of interaction for factored Dec-POMDPs)**.** *The value of the local neighborhood of agent i can be defined as*

$$V^t(\boldsymbol{\delta}_{\mathcal{N}_i}^t) = \sum_{e \ s.t. \ i \in \mathbb{A}(Q^{e,t})} \sum_{\mathbf{x}_e^t} \sum_{\vec{\boldsymbol{\theta}}_e^t} \Pr(\mathbf{x}_e^t,\vec{\boldsymbol{\theta}}_e^t|\boldsymbol{b}^0,\boldsymbol{\varphi}^t) Q_{\boldsymbol{\psi}_e^t}^e(\mathbf{x}_e^t,\vec{\boldsymbol{\theta}}_e^t,\boldsymbol{\delta}_e^t(\vec{\boldsymbol{\theta}}_e^t)). \qquad (5.2.7)$$

Consequently, locality of interaction holds within each stage.

Proof. We can rewrite (5.2.5) to (5.2.7) by restricting the summation to the edges in which agent i participates and subsequently observing that 1) the probability depends only on the past joint policy $\boldsymbol{\varphi}^t$, and 2) $Q_{\boldsymbol{\pi}}^e$ actually depends only on $\boldsymbol{\psi}_e^t$ the future joint policy of the agents in its scope $\mathbb{A}(Q_{\boldsymbol{\pi}}^e)$. (Note that the agent scopes are non-*increasing* when going forward in time.) Now because (5.2.7) defines the

value of the local neighborhood $V^t(\boldsymbol{\delta}^t_{\mathcal{N}_i})$ without any dependence on decision rules δ^t_j of agents $j \notin \mathcal{N}_i$, locality of interaction holds within stage t. □

5.2.4 Approximation of Value Functions

Even when the scope includes all factors and agents with just one backup, the decomposed value function for the last stage still has limited scope. In the optimal solution of a Dec-POMDP, the overwhelming majority of time is spent on the last stage, since it is exponentially harder than the next-to-last stage (Szer et al., 2005).

Therefore we show how this last-stage independence can be exploited in an optimal algorithm, allowing for a significant speedup of the last stage, and thus for the entire computation. Still, we expect that the number of applications for which optimal solutions are feasible is limited. As such, the main emphasis of the remainder of this chapter is on efficient approximations.

There are two main problems in applying the optimal value function identified in this section. First, computing the optimal value function itself is intractable. Second, even if we could compute it, computing an optimal policy with it (as will be explained in the next section), would be intractable because it is fully coupled for earlier stages of the Dec-POMDP. Therefore, Section 5.4 concentrates on finding approximate factored value functions with *predetermined* scopes. The underlying idea is as follows. In the TOI case the optimal scopes are fixed throughout: they are identical to the scopes of the factored immediate reward function. Now if there is no TOI, but the amount of interaction is limited, we may still be able to determine a suitable reduced set of scopes for each of the stages such that it is possible to compute a good approximate solution. Such a predetermined set of scopes may for example simply specify the immediate reward scopes as the scopes for each stage. However, one could use any scheme, e.g., the optimal scopes for the last two stages (i.e., immediate reward scopes for $t = h - 1$ and the back-projection of the immediate reward scopes for $t = h - 2$) and then for the remaining stages also using the scopes specified for $h - 2$.

The basic assumption that underlies this approach is that is is possible to find an approximate factored value function, whose scopes are such that some independence remains. A similar assumption is made by for instance Guestrin et al. (2003), who assume that although the value function of a factored MDP is not factored, it is 'close to factored', and therefore may be approximated well by a factored value function. Similar assumptions have also been made in the context of POMDPs (Guestrin, Koller, and Parr, 2001b).

5.3 Factored Dec-POMDPs via Collaborative Graphical Bayesian Games

This section provides the tools to exploit the independence between agents as represented by the interaction graphs (e.g., Figure 5.4), thereby reducing the agent component n in the complexity of solving a BG for a stage as expressed by Equation (5.0.1). Specifically, it shows that a factored Dec-POMDP with additively

separable rewards can be modeled using a series of *collaborative graphical BGs (CGBGs),* which are much more compact and can be solved more efficiently.

5.3.1 Collaborative Graphical Bayesian Games

A collaborative graphical BG is a graphical BG (Singh et al., 2004b; Soni et al., 2007), in which the agents try to optimize the sum of local rewards, rather than an individual payoff function.

Definition 5.4 (Collaborative graphical BG)**.** A *collaborative graphical Bayesian game (CGBG),* is a tuple $\langle \mathcal{D}, \mathcal{A}, \boldsymbol{\Theta}, \Pr(\boldsymbol{\Theta}), \langle u_1, ..., u_\rho \rangle \rangle$, with:

- $\mathcal{D} = \{1, \ldots, n\}$ is the set of agents.
- $\mathcal{A} = \times_{i=1}^{n} \mathcal{A}_i$ is the set of joint actions. $\mathcal{A}_i$ is the set of actions of agent i.
- $\boldsymbol{\Theta} = \times_i \Theta_i$ is the set of joint types over which a probability function $\Pr(\boldsymbol{\Theta})$ is given (and is assumed common knowledge).
- $u_1, ..., u_\rho$ are *local* payoff functions.

A CGBG is collaborative, which means that all agents try to maximize the same payoff. It is also graphical, which means that the payoff can be decomposed into a sum of local payoff functions, each of which can depend on only a subset of agents, just like in factored Dec-POMDPs. Formally, a local payoff function is specified as a mapping from the types and actions of a subset of agents to real numbers: $u_e : \boldsymbol{\theta}_e \times \mathbf{a}_e \to \mathbb{R}$, where again we abuse e to denote $\mathbb{A}(u_e)$ the subset of agent indices that participate in payoff function e. We also index the local payoff functions using e, because, as for factored Dec-POMDPs, it is possible to construct a hyper-graph $G = \langle \mathcal{D}, \mathcal{E} \rangle$ from the set of payoff functions such that the agent scope of each local payoff function corresponds to a hyper-edge $e \in \mathcal{E}$.

Finally, the goal is to maximize the expected sum of rewards:

$$\boldsymbol{\beta}^* = \arg\max_{\boldsymbol{\beta}} \sum_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \Pr(\boldsymbol{\theta}) \sum_{e \in \mathcal{E}} u_e(\boldsymbol{\theta}_e, \boldsymbol{\beta}_e(\boldsymbol{\theta}_e)), \tag{5.3.1}$$

where $\boldsymbol{\beta}_e(\boldsymbol{\theta}_e)) = \langle \beta_i(\theta_i) \rangle_{i \in e}$ is the joint action of the subset of agents in edge e, resulting from application of their individual BG-policies β_i.

5.3.2 A Dec-POMDP Stage as a CGBG

In a similar fashion to how a BG can model a stage of a (non-factored) Dec-POMDP, a CGBG can model a stage of a factored Dec-POMDP. For each stage t, we want to find a joint decision rule which is specified by $\boldsymbol{\delta}^t \equiv \boldsymbol{\beta}^{t,*}$ the solution of the CGBG for that stage. The actions the agents can take are the same in the Dec-POMDP and the CGBG. The private information each agent i holds is its action-observation history $\vec{\theta}_i^t$, which therefore naturally corresponds to agent i's type: $\theta_i \equiv \vec{\theta}_i^t$. The past joint policy $\boldsymbol{\varphi}^t$ is available when planning, therefore the probability of joint types is specified by $\Pr(\vec{\boldsymbol{\theta}}^t | \boldsymbol{b}^0, \boldsymbol{\varphi}^t) = \sum_{s^t} \Pr(s^t, \vec{\boldsymbol{\theta}}^t | \boldsymbol{b}^0, \boldsymbol{\varphi}^t)$ which is given by (2.5.6). As such, $\mathcal{D}, \mathcal{A}, \boldsymbol{\Theta}, \Pr(\boldsymbol{\Theta})$ have a clear correspondence to the Dec-POMDP.

$Q^{e=1,t=1}(\vec{\theta}_1^1, a_1^1)$

$\vec{\theta}_1^1$		
(H_1,F)	H_1	-0.25
	H_2	-1.10
(H_1,N)	H_1	-0.14
	H_2	-0.79

$Q^{e=2,t=1}(\vec{\boldsymbol{\theta}}^1_{\{1,2\}}, \mathbf{a}^1_{\{1,2\}})$

$\vec{\theta}_1^1$		$\vec{\theta}_2^1$ (H_2,F)		(H_2,N)	
		H_2	H_3	H_2	H_3
(H_1,F)	H_1	-0.55	-1.60	-0.50	-1.50
	H_2	0	-0.55	0	-0.50
(H_1,N)	H_1	-0.16	-1.10	-0.14	-1.00
	H_2	0	-0.16	0	-0.14

$Q^{e=4,t=1}(\vec{\theta}_3^1, a_3^1)$

$\vec{\theta}_3^1$		
(H_4,F)	H_3	-1.50
	H_4	-0.51
(H_4,N)	H_3	-1.10
	H_4	-0.15

$Q^{e=3,t=1}(\vec{\boldsymbol{\theta}}^1_{\{2,3\}}, \mathbf{a}^1_{\{2,3\}})$

$\vec{\theta}_3^1$		$\vec{\theta}_2^1$ (H_2,F)		(H_2,N)	
		H_2	H_3	H_2	H_3
(H_4,F)	H_3	-1.10	0	-0.71	0
	H_4	-1.90	-1.10	-1.70	-0.71
(H_4,N)	H_3	-1.00	0	-0.58	0
	H_4	-1.90	-1.00	-1.60	-0.58

Figure 5.5: The CGBG for stage $t = 1$ of the horizon $h = 2$ FFG problem for past joint policy $\boldsymbol{\varphi}^1 = \langle H_1,H_2,H_4\rangle$. The shading indicates an arbitrary BG-policy for agent 2.

What is left to be determined is the set of payoff functions. If we assume that we can find local Q-functions of the general $Q^e(\mathbf{x}_e^t,\vec{\boldsymbol{\theta}}_e^t,\mathbf{a}_e)$-form as in (5.4.1), we can, for some past joint policy $\boldsymbol{\varphi}^t$, define

$$u_e(\boldsymbol{\theta}_e,\mathbf{a}_e) \equiv Q^e_{\boldsymbol{\varphi}^t}(\vec{\boldsymbol{\theta}}_e^t,\mathbf{a}_e) = \sum_{\mathbf{x}_e^t} \Pr(\mathbf{x}_e^t|\vec{\boldsymbol{\theta}}_e^t,\boldsymbol{b}^0,\boldsymbol{\varphi}^t) Q^e(\mathbf{x}_e^t,\vec{\boldsymbol{\theta}}_e^t,\mathbf{a}_e). \tag{5.3.2}$$

As such, the structure of the CGBG is induced by the structure of the used Q-value function. In Section 5.4, computation of such Q-value functions will be further discussed.

Example 5.2 (FFG via CGBGs). In Figure 5.5, a CGBG for the last stage $t = 1$ of the horizon $h = 2$ FFG problem is shown. The past joint policy $\boldsymbol{\varphi}^1$ is a joint decision rule for the first stage, which is equivalent to a joint action (in this case $\langle H_1,H_2,H_4\rangle$). The figure shows the four local payoff function components. The two on the left ($e = 1,4$) involve only one agent (resp. agent 1 and 3), while the two on the right involve two agents. When two agents go to the same house, the reward is zero as they are sure the fire is extinguished. Also, the entries corresponding to histories with no-flame 'N' observations, typically have a higher payoff than those corresponding to the flame 'F' observations. The figure also has some shaded entries which indicate an arbitrary β_2, a BG-policy for agent 2. This provides some intuition as to why CGBGs are easier to solve than regular BGs: given such a fixed β_2, agents 1,3 can *independently* determine a best response to β_2.

Modeling factored Dec-POMDPs using CGBGs in principle is exact, because it is possible to use an optimal payoff function for the BGs. Theorem 5.1 states that an optimal joint policy $\boldsymbol{\pi}^*$ induces an optimal factored Q-value function,

$Q^e_{\boldsymbol{\pi}^*}(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e,\mathbf{a}_e)$, as follows

$$V^t(\pi^*) = \sum_{e\in\mathcal{E}}\sum_{\mathbf{x}^t_e}\sum_{\vec{\boldsymbol{\theta}}^t_e} \Pr(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e|\boldsymbol{b}^0,\pi^*) Q^e_{\pi^*}(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e,{\pi^*}_e(\vec{\boldsymbol{\theta}}^t_e)) \tag{5.3.3}$$

with $Q^e_{\pi^*}(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e,\mathbf{a}_e)$ as given by (5.4.1). Now it is possible to define an optimal Q-value function of the form of (5.3.2). We can split $\Pr(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e|\boldsymbol{b}^0,\pi^*)$ in a marginal and a conditional. Also $\pi^* = \left(\boldsymbol{\varphi}^{t,*},\boldsymbol{\delta}^{t,*},\boldsymbol{\psi}^{t,*}\right)$ with $\boldsymbol{\varphi}^{t,*},\boldsymbol{\psi}^{t,*}$ the optimal past and future joint policy. This allows us to write:

$$V^t(\pi^*) = \sum_{e\in\mathcal{E}}\sum_{\vec{\boldsymbol{\theta}}^t_e} \Pr(\vec{\boldsymbol{\theta}}^t_e|\boldsymbol{b}^0,\boldsymbol{\varphi}^{t,*}) Q^e_{\pi^*}(\vec{\boldsymbol{\theta}}^t_e,\boldsymbol{\delta}^{t,*}_e(\vec{\boldsymbol{\theta}}^t_e)) \tag{5.3.4}$$

with

$$Q^e_{\pi^*}(\vec{\boldsymbol{\theta}}^t_e,\mathbf{a}_e) = \sum_{\mathbf{x}^t_e} \Pr(\mathbf{x}^t_e|\vec{\boldsymbol{\theta}}^t_e,\boldsymbol{b}^0,\pi^*) Q^e_{\pi^*}(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e,\mathbf{a}_e) \tag{5.3.5}$$

the Q-value function of optimal policy π^*.

Theorem 5.2. *When using* (5.3.5) *as the payoff function for the series of CGBGs representing a Dec-POMDP, forward-sweep policy computation yields an optimal policy.*

Proof. Following the reasoning from Chapter 3, we only have to show that, given that an optimal past policy $\boldsymbol{\varphi}^{t,*}$ is followed, using (5.3.5) as the payoff function of a CGBG yields an optimal joint decision rule $\boldsymbol{\delta}^{t,*}$.

First note that through (5.3.2) we can reformulate the solution of the CGBG (5.3.1) as

$$\boldsymbol{\beta}^{t,*} = \arg\max_{\boldsymbol{\beta}^t} \sum_{e\in\mathcal{E}}\sum_{\vec{\boldsymbol{\theta}}^t_e} \Pr(\vec{\boldsymbol{\theta}}^t_e|\boldsymbol{b}^0,\boldsymbol{\varphi}^t) \sum_{\mathbf{x}^t_e} \Pr(\mathbf{x}^t_e|\vec{\boldsymbol{\theta}}^t_e,\boldsymbol{b}^0,\boldsymbol{\varphi}^t) Q^e(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e,\boldsymbol{\beta}^t_e(\vec{\boldsymbol{\theta}}^t_e)) \tag{5.3.6}$$

$$= \arg\max_{\boldsymbol{\beta}^t} \sum_{e\in\mathcal{E}}\sum_{\vec{\boldsymbol{\theta}}^t_e}\sum_{\mathbf{x}^t_e} \Pr(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e|\boldsymbol{b}^0,\boldsymbol{\varphi}^t) Q^e(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e,\boldsymbol{\beta}^t_e(\vec{\boldsymbol{\theta}}^t_e)). \tag{5.3.7}$$

When an optimal past policy $\boldsymbol{\varphi}^{t,*}$ is followed and using (5.3.5) as the payoff function, applying the same transformation results in an equation similar to (5.3.3):

$$\boldsymbol{\beta}^{t,*} = \arg\max_{\boldsymbol{\beta}^t} \sum_{e\in\mathcal{E}}\sum_{\vec{\boldsymbol{\theta}}^t_e}\sum_{\mathbf{x}^t_e} \Pr(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e|\boldsymbol{b}^0,\boldsymbol{\varphi}^{t,*}) Q^e_{\boldsymbol{\pi}^*}(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e,\boldsymbol{\beta}^t_e(\vec{\boldsymbol{\theta}}^t_e)).$$

An optimal joint decision rule $\boldsymbol{\delta}^{t,*}$ per definition maximizes the expected value V^t, and because $\boldsymbol{\beta}^{t,*}$ does maximize this value, it is an optimal decision rule. □

The implication of this theorem is that modeling a factored Dec-POMDP using CGBGs is exact. Finding $Q^e_{\boldsymbol{\pi}^*}(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e,\mathbf{a}_e)$, however, is intractable in the same way as the non-factored setting described in Section 3.1.5. Therefore we propose to use approximate factored Q-value functions that are easier to compute.

5.3.3 Efficiently Constructing Collaborative Graphical BGs

Regular BGs need exponential space with respect to the number of agents. CGBGs, however, can be represented much more compactly, and therefore are an important step in scaling up to more agents. Still, there is the matter of how CGBGs can be constructed efficiently, which is addressed in this section. In particular, the idea is that each component of the CGBG (i.e., the part corresponding to a local payoff function e) can be constructed independently. Let us assume at the moment that a factored Q-value function of the form $Q^e(\mathbf{x}_e^t,\vec{\boldsymbol{\theta}}_e^t,\mathbf{a}_e)$ is available. As illustrated by (5.3.7) the only other component required for the solution of the CGBG are the probabilities $\Pr(\mathbf{x}_e^t,\vec{\boldsymbol{\theta}}_e^t|\boldsymbol{b}^0,\boldsymbol{\varphi}^t)$ for each edge e.

These probabilities can be computed exactly by application of (2.5.6) and (5.2.3), but become costly for larger problems. Alternatively, it is possible to build upon existing literature and perform exact or approximate inference (filtering) on a DBN model defined on the joint $(s,\vec{\boldsymbol{\theta}}^t)$-space that we call $\vec{\boldsymbol{\theta}}$-DBN:

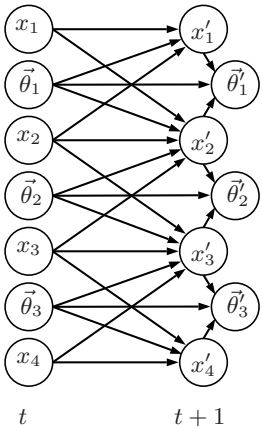
Definition 5.5 ($\vec{\boldsymbol{\theta}}$-DBN)**.** The $\vec{\boldsymbol{\theta}}$-DBN for a past joint policy $\boldsymbol{\varphi}^t$ of a factored Dec-POMDP is the dynamic Bayesian network for stages $0,\ldots,t$ that has state factors and action-observation histories as its nodes.

Figure 5.6a shows two stages of the $\vec{\boldsymbol{\theta}}$-DBN for the FFG problem. The dynamics of the $\vec{\boldsymbol{\theta}}$-DBN follow from the dynamics of the original DBN given $\boldsymbol{\varphi}^t$. For instance, the probability of a particular action-observation history $\vec{\theta}_i^{t+1} = (\vec{\theta}_i^t,a_i^t,o_i^{t+1})$ for agent i is given by $\Pr(\vec{\theta}_i^{t+1}|\mathbf{x}_{\mathcal{I}}^{t+1},\vec{\theta}_i^t) = \Pr(o_i^{t+1}|a_i^t,\mathbf{x}_{\mathcal{I}}^{t+1})\Pr(a_i^t|\boldsymbol{\varphi}^t,\vec{\theta}_i^t)\Pr(\vec{\theta}_i^t)$, where $\mathcal{I}\subseteq\mathcal{X}'$ denotes the subset of factors that has influence on the observation of agent i. This formulation can be further generalized to allow influence of other agents' actions and observations on o_i^{t+1} and thus $\vec{\theta}_i^{t+1}$. For ease of explanation, however, we restrict ourselves to the above simplified setting.

5.3.3.1 Approximate Inference

Exact inference on the $\vec{\boldsymbol{\theta}}$-DBN could be performed by algorithms such as the junction tree or the frontier algorithm (see Murphy (2002) for a review). When both the number of states and the number of agents is low, exact inference is feasible. However, for a larger number of agents and states, exact inference becomes impractical. In general, exact inference requires the representation of the probability distributions over all $|\mathcal{S}|\cdot|\vec{\boldsymbol{\Theta}}^t|$ (state, joint action-observation history)-pairs. The number $|\vec{\boldsymbol{\Theta}}^t|$ scales exponentially with the number of agents, while $|\mathcal{S}|$ scales exponentially with $|\mathcal{X}|$ the number of state factors (of which there are usually more in systems with more agents).

Therefore, we propose to use approximate inference to compute, for all components e, the quantities $\Pr(\mathbf{x}_e^t,\vec{\boldsymbol{\theta}}_e^t|\boldsymbol{b}^0,\boldsymbol{\varphi}^t)$ needed to construct the CGBG for stage t. There are several algorithms for approximate inference on DBNs (Murphy, 2002), such as the algorithm by Boyen and Koller (1998), loopy belief propagation (Kschischang et al., 2001; Mooij, 2008a), the factored frontier algorithm (Murphy and Weiss, 2001), etc. We choose to use the factored frontier (FF) algorithm



(a) The $\vec{\boldsymbol{\theta}}$-DBN.

x_1	0				1			
x_2	0		1		0		1	
$\pi_1(\vec{\theta}_1)$	H_1	H_2	H_1	H_2	H_1	H_2	H_1	H_2
$x'_1=0$	1	1	1	0.2	1	0	0.6	0
$x'_1=1$	0	0	0	0.8	0	1	0.4	1

(b) A compact representation of the CPT associated with x'_1.

x'_1	0				1			
x'_2	0		1		0		1	
$\pi_1(\vec{\theta}_1)$	H_1	H_2	H_1	H_2	H_1	H_2	H_1	H_2
$(\vec{\theta}_1,\pi_1(\vec{\theta}_1),N)$	0.8	0.8	0.8	0.5	0.5	0.8	0.5	0.5
$(\vec{\theta}_1,\pi_1(\vec{\theta}_1),F)$	0.2	0.2	0.2	0.5	0.5	0.2	0.5	0.5

(c) A compact representation of the CPT associated with $\vec{\theta}'_1$.

Figure 5.6: Given a joint policy $\boldsymbol{\pi}$ the standard DBN from Figure 5.1b can be transformed to a $\vec{\boldsymbol{\theta}}$-DBN as illustrated here for FireFightingGraph with $N_f = 2$.

because it is simple and allows computation of some useful intermediate representations (see next). Other, approximate inference algorithms can also be considered (Murphy, 2002; Mooij, 2008a).

The FF algorithm works as follows. Consider a general DBN with state factors $X = \langle X_1, \ldots, X_k \rangle$.[1] FF represents the distribution over states in a fully factored form. That is, if we write p to denote the maintained probability distributions, we have that

$$p(\langle X_1, \ldots, X_k \rangle) \equiv \prod_{i=1}^{k} p(X_i) \tag{5.3.8}$$

Given such a distribution for a stage t, the next-stage distribution can be approximately computed by directly computing the new marginals on each node X_i^{t+1}: the node's CPT is multiplied by the marginals of its parents $X_{P_1}^t \ldots X_{P_l}^t$ and the parents are marginalized out directly:

$$p(X_i^{t+1}) = \sum_{X_{P_1}^t \ldots X_{P_l}^t} \Pr(X_i^{t+1} | X_{P_1}^t \ldots X_{P_l}^t) p(X_{P_1}^t) \ldots p(X_{P_l}^t). \tag{5.3.9}$$

Given the completely factored distribution as computed by FF, we can approximately compute the desired quantities

$$\forall_e \qquad \Pr(\mathbf{x}_e^t, \vec{\boldsymbol{\theta}}_e^t | \boldsymbol{b}^0, \boldsymbol{\varphi}^t) \approx \prod_{i \in \mathbb{X}(Q^e)} p(x_i) \prod_{i \in \mathbb{A}(Q^e)} p(\vec{\theta}_i^t). \tag{5.3.10}$$

Subsequently it is possible to compute both $\Pr(\vec{\boldsymbol{\theta}}_e^t | \boldsymbol{b}^0, \boldsymbol{\varphi}^t)$ and a compact payoff function $u_e(\boldsymbol{\theta}_e, \mathbf{a}_e)$ (through (5.3.2)) for each edge e, thereby defining the CGBG.

[1] There are two types of nodes in the $\vec{\boldsymbol{\theta}}$-DBN, but from a DBN perspective these are all (hidden) state factors. In the following we write X_i for such more abstract state factors.

5.3.3.2 Using Intermediate Results

The FF algorithm can be used to compute approximate probabilities for the construction of CGBGs if a payoff function of the form $Q^e(\mathbf{x}_e^t,\vec{\boldsymbol{\theta}}_e^t,\mathbf{a}_e^t)$ is available. However, such payoff functions will be very large and difficult to compute. As such, Section 5.4 details how to compute payoff functions of the form $Q^e(\mathbf{x}_e^t,\mathbf{a}_e^t)$. However, looking at (5.3.2) we see that using a fully factored distribution means that for all $\vec{\boldsymbol{\theta}}_e^t$ the specified payoff $u_e(\boldsymbol{\theta}_e,\mathbf{a}_e)$ will be the same.

This problem can be overcome by making use of the intermediate results of the FF algorithm. It is possible to perform the computation of the new marginals in a fixed order: first the marginals of all state factors x_j, then the marginals of all $\vec{\theta}_i$. These latter $\vec{\theta}_i$ marginals, are computed in two phases. The first phase marginalizes only over the parents in the *previous* time slice. This leads to a *nearly* completely factored distribution p over states and joint action-observation histories. That is, the state probabilities are represented in a completely factored manner, but the history probabilities are conditionals of the relevant state-factors:

$$p(s,\vec{\boldsymbol{\theta}}) = \prod_{j=1}^{|\mathcal{X}|} p(x_j) \cdot \prod_{i=1}^{n} p(\vec{\theta}_i|\mathbf{x}_{\mathcal{I}}). \tag{5.3.11}$$

Here, $\mathbf{x}_{\mathcal{I}}$ denotes the set of state factors *from the same time slice* that influence the probability of $\vec{\theta}_i$. In the second phase also the $\mathbf{x}_{\mathcal{I}}$ are marginalized out. Now the problem of constructing a CGBG with a payoff function of the form $Q^e(\mathbf{x}_e^t,\mathbf{a}_e^t)$ is solved by using the nearly completely factored distribution (5.3.11).

Example 5.3. An example for FFG starts with a fully factored distribution:

$$p(s,\vec{\boldsymbol{\theta}}) = p(x_1)p(x_2)p(x_3)p(x_4)p(\vec{\theta}_1)p(\vec{\theta}_2)p(\vec{\theta}_3). \tag{5.3.12}$$

First we directly compute the next-stage state factors. For instance

$$p(x_2') = \sum_{x_1,x_2,x_3,\vec{\theta}_1,\vec{\theta}_2} \Pr(x_2'|x_1,x_2,x_3,\vec{\theta}_1,\vec{\theta}_2)p(x_1)p(x_2)p(x_3)p(\vec{\theta}_1)p(\vec{\theta}_2). \tag{5.3.13}$$

Subsequently, the next-stage $\vec{\theta}_i'$ marginals are computed in two phases. In the first phase, we directly marginalize over the previous-stage parents, for instance the probability of $\vec{\theta}_1' = (\vec{\theta}_1,a_1,o_1)$ is computed as

$$p(\vec{\theta}_1'|x_1',x_2') = \Pr(\vec{\theta}_1'|x_1',x_2',\vec{\theta}_1)p(\vec{\theta}_1). \tag{5.3.14}$$

Doing this for all agents yields the nearly factored distribution

$$p(s,\vec{\boldsymbol{\theta}}) = p(x_1)p(x_2)p(x_3)p(x_4)p(\vec{\theta}_1|x_1,x_2)p(\vec{\theta}_2|x_2,x_3)p(\vec{\theta}_3|x_3,x_4), \tag{5.3.15}$$

which can be used to construct the CGBG for the next stage. Finally, the second phase finishes the computation of the $\vec{\theta}_i$-marginals, by marginalizing over the intra-stage parents, e.g. $p(\vec{\theta}_1') = \sum_{x_1',x_2'} p'(\vec{\theta}_1'|x_1',x_2')p(x_1')p(x_2')$, thereby defining a new completely factored distribution, which prepares for a new iteration.

In the more general case, the next-stage observations and thus $\vec{\theta}_i'$ may depend

on the actions of multiple agents. In such a case computing the phase-one $\vec{\theta}'_i$ conditionals would require summing over the histories of the subset of other agents, say G, that are of influence. I.e., in this more general case (5.3.14) becomes

$$p(\vec{\theta}'_1|x'_1,x'_2) = \sum_{\vec{\boldsymbol{\theta}}_G} \Pr(\vec{\theta}'_1|x'_1,x'_2,\vec{\theta}_1,\vec{\boldsymbol{\theta}}_G)p(\vec{\theta}_1)p(\vec{\boldsymbol{\theta}}_G). \tag{5.3.16}$$

We note that these formulations can also be extended to include observation dependencies, although we do not explicitly consider those in this chapter.[1]

A final note is that, except in (5.3.14), all the influence of the previous-stage $\vec{\theta}_i$'s is through the actions that they induce (given the past joint policy $\boldsymbol{\varphi}$). As such, we can gain in efficiency by first computing for all agents, the distributions over their actions

$$p(a_i) = \sum_{\substack{\vec{\theta}_i \text{ s.t.} \\ \varphi_i(\vec{\theta}_i)=a_i}} p(\vec{\theta}_i). \tag{5.3.17}$$

Subsequently the summations over histories in (5.3.13) and (5.3.16) can be replaced by summations over actions.

5.4 Approximate Factored Q-Value Functions

So far we have seen that factored Dec-POMDPs with additive rewards can be represented by CGBGs that can be constructed efficiently using approximate inference. This section covers the computation of factored Q-value functions that can be used as the payoff functions for the CGBGs.

For non-factored Dec-POMDPs, Chapter 4 discussed how the Q-value functions of the 'underlying MDP' ($\mathrm{Q_{MDP}}$) and 'underlying POMDP' ($\mathrm{Q_{POMDP}}$) can be used as heuristic payoff functions for the (non-factored) BGs. However, in order to exploit independence between agents using CGBGs, approximate *factored* Q-value functions *with restricted scopes* are needed. Unfortunately, the underlying MDP and POMDP solution are fully coupled: they are based on the full state s^t and on the full joint belief $\boldsymbol{b}^t$ respectively as will be explained in Subsection 5.4.1.[2]

However, many researchers such as Koller and Parr (1999) have considered factored approximations. Therefore, Subsection 5.4.2 proposes to approximate $Q_{\mathrm{M}}(s,\mathbf{a})$, the fully coupled underlying $\mathrm{Q_{MDP}}$ solution, with a factored value function, by applying linear regression to find the least-squares factored approximation. Still, this approach is problematic for larger domains: the underlying MDP of a factored Dec-POMDP has a number of joint actions that grows exponentially with

[1] Observation dependencies can be modeled by arrows between observations in the original DBN of the factored Dec-POMDP (i.e., Figure 5.1b). As long as these dependencies do not form cycles (and one can always formulate the dependencies in this way) there is an implied ordering in which the $\vec{\theta}'_i$-marginals can be computed.

[2] An exception is presented by Meuleau, Hauskrecht, Kim, Peshkin, Kaelbling, Dean, and Boutilier (1998), who heuristically employ the independent value functions of a set of nearly-independent MDPs (with separate state and action spaces), that are coupled only through the number of available resources, which acts as an external state variable.

the number of agents and a number of states that grows exponentially with the number of state factors. This means that both the solution of the underlying MDP and the regression problem become intractable. Subsection 5.4.3 shows how the complete solution of the underlying MDP can be avoided by bootstrapping in an *approximate dynamic programming (ADP)* algorithm and how the techniques introduced by Koller and Parr can be extended to allow for efficient regression.

Finally, a third and completely new way of coming up with a factored value function is considered. This approach makes use of the solution of smaller *source* problems that involve fewer agents. It is referred to as *transfer planning* and is introduced in Subsection 5.4.4**.**

5.4.1 Nearly Factored Underlying MDP Solutions

Even though Subsection 5.2.2 shows that the value function of a Dec-POMDP is factored (albeit with increasing scopes when moving away from the last stage), the value function for the underlying MDP is not: it becomes fully coupled in just one step. This can be explained by considering (5.2.6), repeated here in a slightly altered form, which gives a description of how the value is being back-propagated:

$$Q^e_{\boldsymbol{\pi}}(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e,\mathbf{a}_e) = R^e(\mathbf{x}^t_e,\mathbf{a}_e) + \sum_{\mathbf{x}^{t+1}_{e'}} \sum_{\mathbf{o}^{t+1}_{e'}} \Pr(\mathbf{x}^{t+1}_{e'},\mathbf{o}^{t+1}_{e'}|\mathbf{x}^t_{\Gamma^{\mathbb{X}}},\mathbf{a}_{\Gamma^{\mathbb{A}}}) Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{t+1}_{e'},\vec{\boldsymbol{\theta}}^{t+1}_{e'},\mathbf{a}^{t+1}_{e'}). \quad (5.4.1)$$

When considering the underlying MDP or POMDP of the Dec-POMDP, value is propagated in exactly the same way; the only difference is in the next stage action $\mathbf{a}^{t+1}_{e'}$ determined by the policy we assume for the next stage. In particular, in the Dec-POMDP case, the value of the next stage that is propagated is determined by the Dec-POMDP policy $\mathbf{a}^{t+1}_{e'} = \boldsymbol{\pi}_{e'}(\vec{\boldsymbol{\theta}}^{t+1}_{e'})$. In contrast, for the optimal policy $\boldsymbol{\pi}^*_{\text{MDP}}$ of the underlying MDP, we assume that at the next stage, the problem is fully observable. As, a result this policy can condition on the nominal state at the next stage, and $\mathbf{a}^{t+1}_{e'}$ is the restriction to e' of

$$\mathbf{a}^{t+1} = \boldsymbol{\pi}^*_{\text{MDP}}(s') = \arg\max_{\mathbf{a}} \sum_{e\in\mathcal{E}} Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{t+1}_{e'},\mathbf{a}^{t+1}_{e'}).$$

(in the MDP setting $\forall_e \forall_{\vec{\boldsymbol{\theta}}^{t+1}_{e'}} Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{t+1}_{e'},\vec{\boldsymbol{\theta}}^{t+1}_{e'},\mathbf{a}^{t+1}_{e'}) = Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{t+1}_{e'},\mathbf{a}^{t+1}_{e'})$). Thus, the 'simplifying' assumption of observing the nominal state at the next stage effectively introduces a dependency on this nominal state. The result is that the value function becomes non-factored at stage $h-2$. A similar argument can be made for the underlying POMDP.

Still, Koller and Parr (1999) argue that, while the exact value function for a factored MDP is non-factored, it might be close to factored: i.e., it may be approximated well by a factored value-function. In particular, Koller and Parr show that

1. using a linear factored value function—defined as the weighted sum of a number of predefined basis functions with restricted (state factor) scopes—a least squares approximation of V^*_{MDP} over the entire state space can be found,

2. this least-squares approximation can be computed efficiently, because the projection of a factored function with restricted scopes w.r.t. the Euclidean norm can be computed efficiently, and

3. the resulting approximation can be close to the exact value function V^*_{MDP}.

Koller and Parr (2000) propose an *approximate policy iteration (API)* algorithm based on these ideas. In subsequent work, Guestrin, Koller, and Parr (2001a); Guestrin et al. (2003) show how also the max-norm $\mathcal{L}_\infty$ can be used for efficient projections, by compactly representing the constraints of a linear program, and adapt API to make use of it. The max-norm projection brings theoretical advantages, because convergence guarantees of policy iteration are based on the max-norm. Szita and Lörincz (2008) propose a similar modification of *approximate value iteration (AVI)*. Both Schuurmans and Patrascu (2002) and Guestrin et al. (2003) improved a related technique called *approximate linear programming (ALP)*, first introduced by Schweitzer and Seidman (1985), that directly approximates the factored MDP with a linear program. Performance bounds for this approach are given by de Farias and Van Roy (2003). Similar ideas were also transferred to the solution of multiagent MDPs (Guestrin, Koller, and Parr, 2002a) and POMDPs (Guestrin et al., 2001b).

While this shows that there are a number of papers that address efficient solutions for factored MDPs, they all consider the approximate solution of factored MDPs over an *infinite horizon* and, to this end, compute *stationary* factored value functions (V). In contrast, the problem considered in this section is different: it is to determine a *non-stationary* (finite-horizon) factored *Q-value* function (Q) for a factored multiagent MDP. The rest of this section uses the tools developed in the mentioned papers, and applies them to the computation of a factored finite-horizon Q-value function. First, we introduce a straightforward approach and introduce some necessary concepts, subsequently we discuss a second approach that is based on the same concepts but more efficient, because it builds on the techniques developed by Koller and Parr (1999).

5.4.2 Factored Q_{MDP}: A Naive Approach using Linear Regression

The simplest approach to computing a non-stationary factored Q-value function considered in this chapter is called *naive regression (NR)*. It first computes the full non-stationary *non-factored* Q-value function, $\text{Q}_{\text{MDP}} = (Q^0_{\text{M}}, \ldots, Q^{h-1}_{\text{M}})$, by solving the underlying MDP with standard dynamic programming (DP) techniques (Puterman, 1994). Next, it computes the approximate *factored* Q-value function for each stage separately by using the found Q^t_{M} as targets in regression.

Naive regression is outlined in Algorithm 5.1. As input it requires for each stage $t = 0, \ldots, h-1$ the desired scopes of the local Q-functions. The scopes of the e-th local Q-function at stage t are denoted $\mathbb{X}(e,t)$,$\mathbb{A}(e,t)$ here. As mentioned, Step 1 can be performed by standard dynamic programming. Step 2 performs the regression for each stage t separately. This involves defining, given some scope, a set of basis functions, one for each 'local state-action pair'.

Algorithm 5.1 Factored $\mathrm{Q}_{\mathrm{MDP}}$ through Naive Regression

Input: The underlying MDP $\mathcal{M}$ of the Dec-POMDP.

Input: The desired scopes for each stage $\mathbb{X}(e,t)$,$\mathbb{A}(e,t)$.

1: Solve $\mathcal{M}$: compute $Q^t_{\mathrm{M}}(s,\mathbf{a})$ for all t and $s = \langle x_1,\ldots,x_{|\mathcal{X}|}\rangle$.

2: For each stage t, solve the following linear regression problem:

$$\forall_s \forall_{\mathbf{a}} \quad Q^t_{\mathrm{M}}(s,\mathbf{a}) \quad \approx \quad \sum_{e\in\mathcal{E}} Q^{e,t}(\mathbf{x}_{\mathbb{X}(e,t)},\mathbf{a}_{\mathbb{A}(e,t)}). \tag{5.4.2}$$

The found factored value function is employed by setting $\forall_{\vec{\boldsymbol{\theta}}^t_e} Q^e(\mathbf{x}^t_e,\vec{\boldsymbol{\theta}}^t_e,\mathbf{a}^t_e) = Q^e(\mathbf{x}^t_e,\mathbf{a}^t_e)$. Subsequently (5.3.2) is used to combine the values into a heuristic factored payoff function for the CGBGs.

5.4.2.1 Local State-Action Pairs and Indicator Functions

Formalizing the decomposition of the $\mathrm{Q}_{\mathrm{MDP}}$ function into a factored Q-value function through regression requires the formalization of several concepts. In particular, linear regression is effectively a projection onto a set of basis functions, so these basis functions on which $Q^t_{\mathrm{M}}(s,\mathbf{a})$ is projected need to be defined in such a way that the resulting approximation is of the desired form: a factored Q-value function with the desired scopes. To this end we propose to use *induced indicator basis functions*. Here we provide only the intuition, for a rigorous formalization, please consult Appendix C.

Let us write N for the number of state-(joint)action pairs $(s,\mathbf{a})$. The desired factored Q-value function has components $Q^{e,t}(\mathbf{x}_{\mathbb{X}(e,t)},\mathbf{a}_{\mathbb{A}(e,t)})$ that are defined over what we call *local* state-action pairs $(\mathbf{x}_{\mathbb{X}(e,t)},\mathbf{a}_{\mathbb{A}(e,t)})$. The total number of such local state-action pairs is denoted with $\hat{N}$ and is the sum of the number of local state-action pairs for each component e. Note that for a typical problem $N \gg \hat{N}$. Let us use $1 \leq l \leq \hat{N}$ as an index over such a local state-action pair for all components e. An *induced indicator basis function* h_l for each *local* state-action pair l takes global state-action pairs $z = (s,\mathbf{a})$ as its argument and indicates whether the supplied z is consistent with l. That is, $h_l(z) = 1$ if and only if z is consistent with local state action pair indicated by l, and 0 otherwise.

5.4.2.2 Formulation of the Regression Problem

Using the notation from the previous section we can rephrase the regression problem (5.4.2) in terms of basis functions: we want to find a weight vector w for which

$$\begin{pmatrix} h_1(z_1) & \ldots & h_l(z_1) & \ldots & h_{\hat{N}}(z_1) \\ \vdots & & \vdots & & \vdots \\ h_1(z_N) & \ldots & h_l(z_N) & \ldots & h_{\hat{N}}(z_N) \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_l \\ \vdots \\ w_{\hat{N}} \end{pmatrix} \approx \begin{pmatrix} Q(z_1) \\ \vdots \\ Q(z_N) \end{pmatrix} \tag{5.4.3}$$

is a good approximation. In matrix notation we write:

$$Hw \approx Q \tag{5.4.4}$$

where H is the matrix of the $\hat{N}$ basis functions:

$$H = \begin{pmatrix} | & & | \\ h_1 & \dots & h_{\hat{N}} \\ | & & | \end{pmatrix}. \tag{5.4.5}$$

In the setting considered here, there is no particular need for max-norm projections (Guestrin et al., 2001a, 2003). Also, although a weighted norm can improve the approximation of a value-function, it may be unsuitable to use such a value function to define a policy as discussed by Koller and Parr (1999).[1] Moreover, it is unclear how to select such weights, which is why we use an unweighted least-squares approximation:

$$\min_w ||Hw - Q||_2 \tag{5.4.6}$$

This is accomplished by selecting w such that Hw is the orthogonal projection of Q into subspace $\mathcal{H} = \text{span}(H)$, the subspace spanned by H. This projection is standard linear algebra (Hefferon, 2008) and is given by

$$\text{proj}_{\mathcal{H}}(Q) = H(H^T H)^{-1} H^T Q, \tag{5.4.7}$$

where ^{T} denotes transpose and $^{-1}$ inverse. Equating the desired result to the projection

$$Hw = H(H^T H)^{-1} H^T Q, \tag{5.4.8}$$

derives

$$w = (H^T H)^{-1} H^T Q. \tag{5.4.9}$$

as the w that minimizes the error. This is exactly the operation as it is implemented by any linear regression package.

5.4.2.3 Scalability of Naive Regression

The outlined approach works well for small problems, but it scales poorly with the number of agents and number of state factors, as computing all the non-factored Q_{MDP} values and performing linear regression—which requires the explicit construction of H—becomes intractable. Fortunately, because all basis functions have a restricted induced scope, construction of H can be avoided and $(H^T H)^{-1}$ can be computed efficiently, because the inner products of basis functions can be computed efficiently (Koller and Parr, 1999). The work of Koller and Parr is extended by showing that this computation is particularly efficient for induced indicator basis functions, as detailed in Appendix C.2. Still computation of the full Q and the inner product $H^T Q$ cannot be avoided in naive regression.

[1] When selecting a greedy policy from a value function, it is more important that the relative values of all states are preserved than that the absolute values are approximated accurately. However, weighting means that the error can be quite different for different states, and therefore the relative values can be perturbed.

Algorithm 5.2 Approximate DP.

Input: The underlying factored MDP $\mathcal{M}$ of the Dec-POMDP.
Input: The desired scopes for each stage $\mathbb{S}(e,t) = \langle \mathbb{X}(e,t), \mathbb{A}(e,t) \rangle$.
Output: The factored approximate value function for $\mathcal{M}$.

1: **if** immediate reward scopes do not match desired scopes $\exists_e \ \mathbb{S}(R^e) \neq \mathbb{S}(e,t)$ **then**
2: compute Q^{h-1} through projection:

$$\forall_s \quad \sum_{e \in \mathcal{E}} Q^{e,h-1}(\mathbf{x}_{\mathbb{X}(e,t)}, \mathbf{a}_{\mathbb{A}(e,t)}) \approx \sum_{e \in \mathcal{E}^R} R^e(\mathbf{x}_e, \mathbf{a}_e). \tag{5.4.10}$$

3: **else**
4: $Q^{h-1} \leftarrow R$
5: **end if**
6: **for** $t = h-1$ to 1 **do**
7: $Q^{t-1} \leftarrow \text{ComputePreviousStageQ}(Q^t, \mathcal{M}, \mathbb{S}(t-1))$
8: **end for**

A solution would be to consider a sample-based approach. Rather than computing the full non-factored Q_{MDP}-function, it generates a set of targets by sampling state-action pairs and estimates their value by performing backups (also based on sampling) from the factored Q_{MDP} values for stage $t+1$. That is, the factored approximation for t is estimated directly, or *bootstrapped,* from the approximation for $t+1$. One difficulty in the setting of approximate value function for a Dec-POMDP is how to select these samples. Drawing them uniformly might be quite wrong, since the true distribution may be quite different: it is based on the past policy, which is unknown. Note that such a sample-based method would be closely related to reinforcement learning (Sutton and Barto, 1998), and it may be possible to reuse methods explicitly designed to work with factored Q-value functions (Kok and Vlassis, 2006). Section 5.4.3 adopts a different approach building upon the work by Guestrin et al. (2003).

5.4.3 Factored Q_{MDP}: Approximate Dynamic Programming

For large problems, it is more efficient and elegant to interleave the projection steps with backup steps. That is, the approximate Q-value function of a stage t is bootstrapped from the approximation for stage $t+1$. This means that errors may accumulate. However, when the state space gets very large there seems to be no other option for the computation of Q_{MDP} than to resort to such a scheme, because computation of the flat Q_{MDP} solution is no longer possible.

To implement this idea of bootstrapping to compute a non-stationary factored Q-value function, we propose a straightforward algorithm, which we refer to as *approximate dynamic programming (ADP).*[1] The algorithm is closely related to

[1] Arguably, approximate dynamic programming is an inconvenient name since in the operations research community (e.g., see Powell, 2007) it is used to describe the techniques that the

the factored value iteration approach of Szita and Lörincz (2008), but computes non-stationary value functions and materializes the desired factored Q-value function for each step. ADP is shown in Algorithm 5.2. Lines 1–5 correspond to the initialization and are run only once. Its main component is a subroutine `ComputePreviousStageQ`, shown in Algorithm 5.3, that computes Q^t from Q^{t+1}.

ADP starts by checking whether the scope of the immediate reward function is the same as the scope desired for the factored Q-value function of the last stage. Algorithm 5.2 uses $\mathcal{E}$ to denote the set of components (or edges of the corresponding interaction hyper-graph) of Q^{h-1}. The ρ components of the immediate reward are denoted by $\mathcal{E}^R$. If the scopes for Q^{h-1} are not equal to the immediate reward scopes, a projection is first performed using the equivalent of (5.4.9)

$$w = (H^T H)^1 H^T R, \tag{5.4.11}$$

which results in a factored value function $Q^{h-1} = Hw$ that is the the least-squares approximation of R. Because H is the matrix of induced indicator basis functions and R is factored, (5.4.11) can be computed efficiently as detailed in Appendix C.

Subsequently, ADP computes the factored Q-value function for the remaining stages $h-2,\ldots,0$ by calling `ComputePreviousStageQ`. This function is shown in Algorithm 5.3. Note that the set of components of Q^{t+1} is denoted using $\mathcal{E}^{t+1}$ while the set of components of Q^t (induced by the desired scopes $\mathbb{S}(e,t)$) are denoted using $\mathcal{E}^t$. The function consists of three steps.

In the first step, Q^{t+1} is converted to a factored value function V^{t+1}. In general, this is a complicated procedure that requires a maximization for each state and that results in a non-factored value function V^{t+1} which consequently has to be factored by projection. This would mean that again the resulting algorithm will not scale well. To overcome this problem, we propose to make a fast approximation by maximizing each component independently, similar in spirit to the approximation used by Kumar and Zilberstein (2009). That is we use

$$\forall_{e\in\mathcal{E}^{t+1}} \quad V^{e,t+1}(\mathbf{x}_{\mathbb{X}(e,t+1)}) \leftarrow \max_{\mathbf{a}_{\mathbb{A}(e,t+1)}} Q^{e,t+1}(\mathbf{x}_{\mathbb{X}(e,t+1)},\mathbf{a}_{\mathbb{A}(e,t+1)}). \tag{5.4.12}$$

Clearly this is a crude approximation and will result in an overestimation of the value. However, it may be the case that the relative values will be largely preserved, as a similar overestimation is made for each local state. Another option would be to perform this step in a sample-based fashion, as for instance proposed by Szita and Lörincz (2008).[1] Still, such sample-based approaches typically require a lot of samples and are therefore slow in practice, which is why we have not considered this option further. Note that in such a case, the state factor scopes of the components $V^{e,t+1}(\mathbf{x}_{\mathbb{X}(e,t+1)})$ need not be the same as those of $Q^{e,t+1}$.

The second step back-projects each of the components $V^{e,t+1}$ through the transition model, constructing the components $g^{e,t}$. In the algorithm $\Gamma^{\mathbb{X}}$, $\Gamma^{\mathbb{A}}$ are

AI community calls Reinforcement Learning. Nevertheless we choose the term ADP in order to be consistent with the naming for API and ALP.

[1] Szita and Lörincz (2008) show PAC-bounds for performing uniform sampling for this step: using only polynomially many samples (in the representation of the factored MDP) with high probability the approximate solution of this step has low error.

Algorithm 5.3 ComputePreviousStageQ(Q^{t+1},$\mathcal{M}$,$\mathbb{S}(t)$)

Input: Q^{t+1}, the factored Q-value of the next stage.
Input: The underlying factored MDP $\mathcal{M}$ of the Dec-POMDP.
Input: The desired scopes for stage t $\mathbb{S}(e,t) = \langle \mathbb{X}(e,t), \mathbb{A}(e,t) \rangle$.
Output: Q^t, the factored Q-value for stage t.

1: Compute a restricted scope factored value function V^{t+1} from Q^{t+1} such that

$$\forall_s \quad \sum_{e \in \mathcal{E}^{t+1}} V^{e,t+1}(\mathbf{x}_{\mathbb{X}(e,t+1)}) \approx \max_{\mathbf{a}} \sum_{e \in \mathcal{E}^{t+1}} Q^{e,t+1}(\mathbf{x}_{\mathbb{X}(e,t+1)}, \mathbf{a}_{\mathbb{A}(e,t+1)}). \quad (5.4.13)$$

2: Compute $g^{e,t}$ the back-projection of $V^{e,t+1}$ for all $e \in \mathcal{E}^{t+1}$:

$$\forall_{\mathbf{x}_{\Gamma^{\mathbb{X}}}, \mathbf{a}_{\Gamma^{\mathbb{A}}}} \quad g^{e,t}(\mathbf{x}_{\Gamma^{\mathbb{X}}}, \mathbf{a}_{\Gamma^{\mathbb{A}}}) = \gamma \sum_{\mathbf{x}'_e} \Pr(\mathbf{x}'_e | \mathbf{x}_{\Gamma^{\mathbb{X}}}, \mathbf{a}_{\Gamma^{\mathbb{A}}}) V^{e,t+1}(\mathbf{x}'_e). \quad (5.4.14)$$

3: Compute Q^t as the least-squared error approximation:

$$\forall_s \quad \sum_{e \in \mathcal{E}^t} Q^{e,t}(\mathbf{x}_{\mathbb{X}(e,t)}, \mathbf{a}_{\mathbb{A}(e,t)}) \approx \sum_{e \in \mathcal{E}^R} R^e(\mathbf{x}_e, \mathbf{a}_e) + \sum_{e \in \mathcal{E}^{t+1}} g^{e,t}(\mathbf{x}_{\Gamma^{\mathbb{X}}}, \mathbf{a}_{\Gamma^{\mathbb{A}}}). \quad (5.4.15)$$

shorthand for the state factor and agent backup of the next-stage state scope $\Gamma^{\mathbb{X}}(\mathbb{X}(V^{e,t+1}))$, $\Gamma^{\mathbb{A}}(\mathbb{X}(V^{e,t+1}))$.

Step 3 requires another projection, but this time no maximization is involved. Again, because both the left-hand and the right-hand side of (5.4.15) are factored functions with restricted induced scopes, this projection can be computed efficiently, as described in Appendix C. Note that, when using a different method for step 1, it may be possible to choose the scopes of $V^{e,t+1}$ in such a way that the back-projected $g^{e,t}(\mathbf{x}_{\Gamma^{\mathbb{X}}}, \mathbf{a}_{\Gamma^{\mathbb{A}}})$ can be immediately combined with immediate reward components R^e to form Q-functions of the desired form. This would make the projection in Step 3 unnecessary.

5.4.4 Transferring Q-value Functions

The high-level goal of entire Section 5.4 is to come up with some meaningful heuristic for each component e of the CGBG that has the form $Q^e_{\boldsymbol{\varphi}^t}(\vec{\boldsymbol{\theta}}^t_e, \mathbf{a}_e)$. Subsections 5.4.2 and 5.4.3 did this by searching for a heuristic of the form $Q^e(\mathbf{x}^t_e, \vec{\boldsymbol{\theta}}^t_e, \mathbf{a}_e)$ and transforming it to the former form using (5.3.2). This subsection proposes a different approach that directly tries to find heuristic values

$$Q^e_{\boldsymbol{\varphi}^t}(\vec{\boldsymbol{\theta}}^t_e, \mathbf{a}_e) \equiv Q^{\mathrm{s}}(\vec{\boldsymbol{\theta}}^t, \mathbf{a}) \quad (5.4.16)$$

by solving smaller, but similar, *source tasks* and using their value functions Q^{s}. These source tasks and their correspondence to the original problem are manually specified as will be detailed in Subsection 5.4.4.1, but this may be automated in the future.

This idea is closely related to *inductive transfer* or *transfer learning,* in which a set of tasks is used to bias the learning process on a new task (Thrun, 1996; Thrun and Pratt, 1998; Baxter, 2000; Torrey, Walker, Shavlik, and Maclin, 2005; Rosenstein, Marx, Kaelbling, and Dietterich, 2005). In this line of research learning a target task is bootstrapped from a simpler source task. This way the target task can be learned more efficiently. Also, specific source tasks may be designed such that the total time needed to first learn the source task and then the target task is less than when learning the target task directly. Many such methods have been proposed within a more general machine learning context. Within the field of sequential decision making, inductive transfer has mostly been applied to reinforcement learning (Selfridge, Sutton, and Barto, 1985; Singh, 1992; Wilson, Fern, Ray, and Tadepalli, 2007; Taylor, Stone, and Liu, 2007; Taylor, Kuhlmann, and Stone, 2008; Taylor and Stone, 2009). Typically, a policy or value function learned on a source task is used to construct an initial policy for the target task, which is then refined by further learning. In contrast, the method proposed here considers a planning task and uses the value function of multiple source tasks as a heuristic. We refer to this method as *transfer planning (TP)* and the resulting approximate Q-value function as Q_{TP}.

5.4.4.1 Formalization of Multiagent Transfer Planning

The goal of transfer planning is to compute an approximate factored Q-value function by identifying a source problem for each component, or edge. The basic idea is that it is possible to directly use the Q-value function of a source task as a component Q^e. As before, we assume that the desired structure of the factored Q-value function in terms of scopes is known, i.e., for all stages and all edges $\mathbb{S}(e,t) = \langle \mathbb{X}(e,t), \mathbb{A}(e,t) \rangle$ is specified. To simplify the discussion, the following assumes some particular stage t.

TP is defined using the following components:

- S—the set of source problems $\mathrm{s} \in \mathrm{S}$
- $\mathcal{D}^{\mathrm{s}} = \{1^{\mathrm{s}}, \ldots, n^{\mathrm{s}}\}$ —the set of agents of source problem s.
- E—a function that maps each edge e to a source problem $E(e) = \mathrm{s} \in \mathrm{S}$. We require that $\mathbb{A}(e)$ the agent scope of e for the considered stage contains at least as many agents as the assigned source problem: $|\mathbb{A}(e)| \geq n^{E(e)}$.
- A^e—for each edge e we define a mapping $A^e : \mathbb{A}(e) \to \mathcal{D}^{E(e)}$ that maps agent indices in the target task $i \in \mathbb{A}(e)$ to indices $A^e(i) = j^{\mathrm{s}} \in \mathcal{D}^{\mathrm{s}}$ in the source task s for that particular edge $\mathrm{s} = E(e)$. Note that A^e is surjective, such that all $j^{\mathrm{s}} \in \mathcal{D}^{\mathrm{s}}$ are pointed to.

With some abuse of notation A^e is overloaded to also work on profiles of agents, actions and histories. E.g.,

$$A^e(\vec{\boldsymbol{\theta}}^t_e) = A^e(\langle \ldots, \vec{\theta}^t_i, \ldots \rangle_{i \in e}) = \langle \ldots, \vec{\theta}^t_{A^e(i)}, \ldots \rangle_{i \in e} = \vec{\boldsymbol{\theta}}^t_{A^e(e)}. \tag{5.4.17}$$

This allows us to formally define the transfer we apply as follows.

e	$\mathbb{A}(R^e)$	$\mathbb{X}(R^e)$
1	{1}	{1,2}
2	{1,2}	{1,2,3}
3	{2,3}	{2,3,4}
4	{3,4}	{3,4,5}
5	{4,5}	{4,5,6}
6	{5}	{5,6}

(a) Original Scopes.

e	$\mathbb{A}(e)$	$\mathbb{X}(e)$
1	{1,2}	{1,2,3}
2	{2,3}	{2,3,4}
3	{3,4}	{3,4,5}
4	{4,5}	{4,5,6}

(b) Reduced Scopes.

Table 5.1: The scopes of the immediate reward functions of 5-agent FFG.

Definition 5.6 ($\mathrm{Q_{TP}}$)**.** Given a set of source problems that satisfy the above requirements, and some (heuristic) Q-value functions for them,

$$Q^e_{\mathrm{TP}}(\vec{\boldsymbol{\theta}}^t_e,\mathbf{a}_e) \equiv Q^{\mathrm{s}}(\vec{\boldsymbol{\theta}}^t_{A^e(e)},\mathbf{a}_{A^e(e)}), \qquad \mathrm{s} = E(e). \tag{5.4.18}$$

This means that it is only necessary to define source problems and the corresponding mapping A^e for each edge e and to find a good heuristic Q-value function $Q^{\mathrm{s}}(\vec{\boldsymbol{\theta}}^t,\mathbf{a})$ for each of the source problems. Since the source problems are typically selected to be small, it is possible to treat them as non-factored and use the heuristic $\mathrm{Q_{MDP}}$, $\mathrm{Q_{POMDP}}$ and $\mathrm{Q_{BG}}$ value functions as discussed in Chapter 4.

Example 5.4 ($\mathrm{Q_{TP}}$ for the FFG problem.). This illustrates the application of $\mathrm{Q_{TP}}$ to the 5-agent FFG problem. In this problem there are 6 houses. However, the immediate reward scopes of the first and last house are sub-scopes of other scopes as illustrated in Table 5.1a and as such we can reduce them such that the desired Q-value function will be factored as shown in Table 5.1b.
Now, for each of the 4 components e it is necessary to 1) define a source task $E(e)$, 2) define a mapping A^e and 3) compute a (heuristic) Q-value function for the source task such that we can transfer using (5.4.18). We propose to use 2-agent FFG as the source task for each of the four components. The agent mapping is defined such that the agent with the lower index is mapped to 1^{s} and the agent with the higher index to 2^{s}. E.g., for $e = 3$, $A^e(3) = 1^{\mathrm{s}}$,$A^e(4) = 2^{\mathrm{s}}$. Finally, computation of the underlying centralized $\mathrm{Q_{MDP}}$ function for 2-agent FFG provides the values to define Q^e_{TP} using (5.4.18).

5.5 Solution of Collaborative Graphical BGs

The previous sections discussed how a factored Dec-POMDP can be represented as a series of CGBGs, how such CGBGs can be efficiently constructed and how we can compute approximate payoff functions for them. This section explains how CGBGs can be solved efficiently. Since the results for this section are not specific to CGBGs used to model Dec-POMDPs, we abstract away the Dec-POMDP context.

The solution of a CGBG is given by

$$\begin{aligned}\boldsymbol{\beta}^{t,*} &= \arg\max_{\boldsymbol{\beta}^t} \sum_{e\in\mathcal{E}} \sum_{\vec{\boldsymbol{\theta}}_e^t} \Pr(\vec{\boldsymbol{\theta}}_e^t|\boldsymbol{\varphi}^t,\boldsymbol{b}^0) Q^{e,t}(\vec{\boldsymbol{\theta}}_e^t,\boldsymbol{\beta}_e^t(\vec{\boldsymbol{\theta}}_e^t)) \\ &= \arg\max_{\boldsymbol{\beta}} \sum_{e\in\mathcal{E}} \sum_{\boldsymbol{\theta}_e} \Pr(\boldsymbol{\theta}_e) u_e(\boldsymbol{\theta}_e,\boldsymbol{\beta}_e(\boldsymbol{\theta}_e)) \qquad (5.5.1)\end{aligned}$$

where $e = \mathbb{A}(Q^{e,t})$. The second equation ignores the Dec-POMDP context.

5.5.1 Nonserial Dynamic Programming

Optimal solution of regular BGs requires evaluating all joint BG policies. In contrast, CGBG can be optimally solved more efficiently using *non-serial dynamic programming (NDP)* (Bertele and Brioschi, 1972; Rosenthal, 1977), also known as *variable elimination* (Guestrin et al., 2002a; Vlassis, 2007). This technique works by iteratively eliminating agents (variables) from the maximization. Let us define, for all edges e, the value for all local BG-policies $\boldsymbol{\beta}_e$

$$u_e(\boldsymbol{\beta}_e) \equiv \sum_{\boldsymbol{\theta}_e} \Pr(\boldsymbol{\theta}_e) u_e(\boldsymbol{\theta}_e,\boldsymbol{\beta}_e(\boldsymbol{\theta}_e)). \qquad (5.5.2)$$

Using this formulation, (5.5.1) can be rewritten as

$$\boldsymbol{\beta}^{t,*} = \arg\max_{\boldsymbol{\beta}^t} \sum_{e\in\mathcal{E}} u_e(\boldsymbol{\beta}_e). \qquad (5.5.3)$$

Example 5.5 (NDP for the FFG example of Figure 5.5). The maximization we are performing in this case is

$$\max_{\boldsymbol{\beta}} \Big[u_1(\beta_1) + u_2(\boldsymbol{\beta}_{\{1,2\}}) + u_3(\boldsymbol{\beta}_{\{2,3\}}) + u_4(\beta_3) \Big]. \qquad (5.5.4)$$

It is possible to isolating the functions in which agent 1 participates

$$\max_{\boldsymbol{\beta}_{\{2,3\}}} \Big[u_3(\boldsymbol{\beta}_{\{2,3\}}) + u_4(\beta_3) + \max_{\beta_1} \Big[u_1(\beta_1) + u_2(\boldsymbol{\beta}_{\{1,2\}}) \Big] \Big],$$

This inner maximization can now be written as $f_1(\beta_2)$ a function representing the best-response value contributed by agent 1 when agent 2 selects β_2. This allows us to isolate the maximization for the policy of agent 2:

$$\begin{aligned} &= \max_{\boldsymbol{\beta}_3} \Big[u_4(\beta_3) + \max_{\beta_2} \Big[u_3(\boldsymbol{\beta}_{\{2,3\}}) + f_1(\beta_2) \Big] \Big] \\ &= \max_{\boldsymbol{\beta}_3} \Big[u_4(\beta_3) + f_{\{1,2\}}(\beta_3) \Big] \Big]. \end{aligned}$$

Here $f_{\{1,2\}}$ is the function that will compute the maximizing (best-response) contribution of agents 1 and 2, given β_3. Now it is easy to determine β_3^* the maximizing β_3 by just looping over all possible options. Subsequently, $f_{\{1,2\}}$ can be used to determine β_2^* and f_1 can be used to determine β_1^*. At this point we have found $\boldsymbol{\beta}^* = \langle \beta_1^*,\beta_2^*,\beta_3^* \rangle$ without looping over the space of joint BG-policies.

Conceptually, the u_e functions from (5.5.4) define a graphical normal-form game, which is then solved by NDP. This normal-form game does not need to be constructed explicitly, however.

The time needed by NDP is exponential in the induced width of the interaction graph (Guestrin et al., 2002a), i.e., the maximum number of agents that are participating in any function u_e or f. Let us denote this width w, then we have that the complexity of solving the last stage CGBG is

$$O\Big(n \cdot |\mathcal{A}_*|^{w(|\mathcal{O}_*|^{h-1})}\Big), \tag{5.5.5}$$

yielding an exponential speedup over (5.0.1) as long as $n \gg w$.

5.5.2 CGBGs as Factor Graphs

The agents in a CGBG are collaborative and try to optimize the same global payoff function. As such it is possible to interpret a CGBG in the more general paradigm of factor graphs (Kschischang et al., 2001; Loeliger, 2004). This section introduces two factor graph representations of CGBGs.

5.5.2.1 Policy Factor Graphs

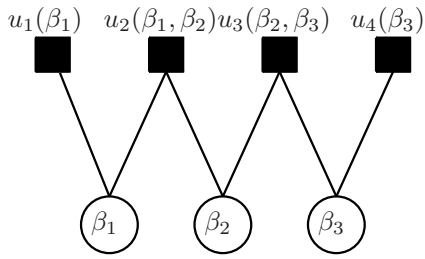
Section 5.1 introduced interaction graphs and how they model independence between agents. In particular, two agents in an interaction graph are connected when there is a component of the value function that involves the policies of both those agents. Like many graphical models, interaction graphs can be seen as a special case of factor graphs. In particular, the factor graph makes explicit the components of the value functions as *factors* and the variables that influence them.

Definition 5.7 (Factor Graph)**.** A *factor graph (FG)* is a bipartite graph with a set of factors $\mathcal{F} = \{F_1, \ldots, F_{|\mathcal{F}|}\}$ and variables $\mathcal{V} = \{V_1, \ldots, V_{|\mathcal{V}|}\}$. Each factor is a function F_i defined over a subset $S \subseteq \mathcal{V}$ of the variables: $F_i(\mathbf{v}_S)$. A factor F_i and variable V_j are connected if and only if the variable is in the scope of the factor $V_j \in \mathbb{S}(F_i)$.

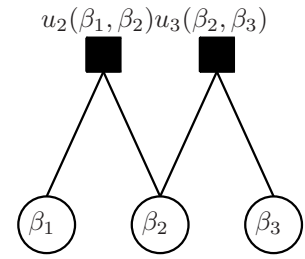
A CGBG can be represented as a factor graph with individual BG-policies as variables, and a factored payoff function defined over these policies. In particular, the description of the solution of the CGBG as given by (5.5.3) corresponds to a factor graph, dubbed the *policy factor graph (PFG).* The PFG for the CGBG from Figure 5.4b is illustrated in Figure 5.7a. Because this CGBG contains payoff components of which the agent scopes are sub-scopes of others, it can be reduced as in Exampe 5.4. The PFG for this reduced CGBG is shown in Figure 5.7b. In the Dec-POMDP context, $u_e(\boldsymbol{\beta}_e)$ should be interpreted as $V^{e,t}_{\boldsymbol{\varphi}^t}(\boldsymbol{\beta}^t_e)$, a heuristic for the future value as expressed by (5.2.7).

5.5.2.2 Type-Action Factor Graphs

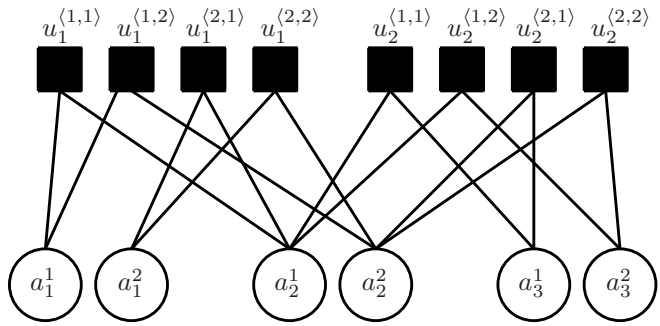
The policy factor graph introduced above is a very intuitive formulation and relates directly to interactions graphs and the discussion of locality of interaction in



(a) A policy factor graph for a CGBG of the FFG problem. The variables correspond to the BG-policies β_i for the agents. Note the correspondence to the interaction graph of Figure 5.4b.



(b) The PFG for the same, but reduced CGBG.



(c) The type-action factor graph (TAFG) representation of the same (reduced) CGBG.

Figure 5.7: Factor graph representations of CGBGs. The factors are displayed as black squares. The variables are circles.

Section 5.1 and 5.2. However, there is some independence that this representation does not model.

In particular, an individual BG policy specifies an action for each type. Now, as illustrated by Figure 5.7c, such a chosen action a_i for some type θ_i of agent i does not directly influence the contribution of a different action a'_i chosen for a different type θ'_i of the same agent. This section introduces a second factor graph formulation, dubbed the *type-action factor graph (TAFG),* that models this independence.

In the following a_i^k will denote the action agent i selects for its k-th type θ_i^k. I.e.,

$$a_i^k \equiv \beta_i(\theta_i^k) \equiv \delta_i^t(\vec{\theta}_{i,k}^t). \tag{5.5.6}$$

For each *local* joint type $\boldsymbol{\theta}_e$, a payoff function is defined as follows:

$$u_{\boldsymbol{\theta}_e}(\mathbf{a}_e) \equiv \Pr(\boldsymbol{\theta}_e) u_e(\boldsymbol{\theta}_e, \mathbf{a}_e) \tag{5.5.7}$$

We also write $u_e^{\langle 1,1 \rangle}$ for $u_{\boldsymbol{\theta}_e = \langle 1,1 \rangle}$(in Figure 5.7c). Using this notation, it is possible to express the solution of the CGBG as

$$\boldsymbol{\beta}^* = \arg\max_{\boldsymbol{\beta}} \sum_{e \in \mathcal{E}} \sum_{\boldsymbol{\theta}_e} u_{\boldsymbol{\theta}_e}(\boldsymbol{\beta}_e(\boldsymbol{\theta}_e)) \tag{5.5.8}$$

$$= \arg\max_{\boldsymbol{\beta}} \sum_{e \in \mathcal{E}} \sum_{\boldsymbol{\theta}_e} u_{\boldsymbol{\theta}_e}(\langle a_{e_1}^{k_1}, \ldots, a_{e_{|e|}}^{k_{|e|}} \rangle) \tag{5.5.9}$$

where e_i denotes the i-th agent in e and k_i denotes the index of its individual type as specified by $\boldsymbol{\theta}_e = \langle \theta_{e_1}^{k_1}, \ldots, \theta_{e_{|e|}}^{k_{|e|}} \rangle$.

In effect, we are trying to maximize a function which contains a factor for each local joint type $\boldsymbol{\theta}_e$ and this function can be represented by a type-action factor graph, as illustrated in Figure 5.7c.

5.5.3 Maximization over a Factor Graph using Max-Plus

In the previous section it was shown that a CGBG can be represented as a factor graph in two ways. In order to find the solution for a CGBG, however, it is necessary to find the configuration of variables that maximizes the sum of the factors. This can be done using the Max-Plus algorithm (Pearl, 1988; Wainwright, Jaakkola, and Willsky, 2004; Kok and Vlassis, 2005; Vlassis, 2007), which is a distributed algorithm based on message passing. It was originally proposed to compute the maximum a posteriori probability configurations in Bayesian networks and is a special case of the *sum-product algorithm* (Kschischang et al., 2001)—also referred to as belief propagation in probabilistic domains.

On an intuitive level, Max-Plus works by iteratively sending messages between factors and variables. These messages encode how much payoff the sender F_i expects to be able to contribute to the total payoff, given each of the possible values j the receiving variable V_j can take on. To estimate this value, the sending factor F_i makes use of the incoming messages sent by all variables to which it is connected except V_j.

In the following these messages are formalized. To ease notation we use uppercase for factors and lowercase for variables and write $\mu_{i\to I} = \mu_{V_i\to F_I}$ for the message sent from variable i to factor I, etc. We use i to denote both the variable and the value it can take. The context should make clear what is meant in each case. The messages sent from variables to factors are defined as

$$\mu_{i\to I}(i) = \sum_{F_J\in\mathcal{N}(V_i)\setminus F_I} \mu_{J\to i}(i), \tag{5.5.10}$$

where $\mathcal{N}(V_i)$ is the set of neighbors of variable V_i. Effectively, variable V_i performs an element-wise addition of the incoming messages from other factors F_J and sends the resulting vector $\mu_{i\to I}$ to F_I.

Now we consider a factor $F_I(i,j_1,\ldots,j_k)$ which, per definition, is connected to variables $V_i,V_{j_1},\ldots,V_{j_k}$. The message $\mu_{I\to i}$ that F_I sends to V_i should be a summary of the expected reward it can contribute and depends on the incoming messages from $V_{j_1},\ldots,V_{j_k}$. These messages however can have different sizes, which is resolved by introducing $\oplus$ to add vector messages to the consistent entries of a factor.

Definition 5.8 (Factor-message addition $\oplus$)**.** For a factor $F_K(a,b,c)$ we have that

$$\forall_{a,b,c} \quad (F_K \oplus \mu_{b\to K})(a,b,c) \;\equiv\; F_K(a,b,c) + \mu_{b\to K}(b). \tag{5.5.11}$$

Example 5.6. For instance, if we have a factor $F_K(x,y)$ that depends on two variables x and y with respectively 3 and 2 values:

$$F_K = \begin{pmatrix} F_K(1,1) & F_K(1,2) \\ F_K(2,1) & F_K(2,2) \\ F_K(3,1) & F_K(3,2) \end{pmatrix}.$$

Suppose all F_K's entries are 0 and we have a message $\mu_{x\to F_K}(x) = (1,2,3)^T$, then

$$F_K \oplus \mu_{x\to F_K} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{pmatrix}.$$

Similarly, for a message $\mu_{y\to F_K}(y) = (4,5)^T$, the result is

$$F_K \oplus \mu_{y\to F_K} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \oplus \begin{pmatrix} 4 \\ 5 \end{pmatrix} = \begin{pmatrix} 4 & 5 \\ 4 & 5 \\ 4 & 5 \end{pmatrix}.$$

Combining above expressions yields

$$F_K \oplus \mu_{x\to F_K} \oplus \mu_{y\to F_K} = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{pmatrix} \oplus \begin{pmatrix} 4 \\ 5 \end{pmatrix} = \begin{pmatrix} 5 & 6 \\ 6 & 7 \\ 7 & 8 \end{pmatrix}.$$

Now the message $\mu_{I\to i}$ is defined as

$$\mu_{I\to i}(i) = \max_{j_1,\ldots,j_l} \Big((F_I \oplus \mu_{j_1\to I} \oplus \cdots \oplus \mu_{j_k\to I})(i,j_1,\ldots,j_k) \Big). \tag{5.5.12}$$

MAX-PLUS proceeds to iteratively send these messages over the edges of the FG. Within each iteration, the messages can be assumed to be sent in parallel, or sequentially with some fixed or random ordering. When run on a FG without any cycles (i.e., a tree), the algorithm is guaranteed to converge to an optimal fixed point (Pearl, 1988; Wainwright et al., 2004). In FGs with cycles, such as the ones defined in the Section 5.5.2, there are no guarantees that MAX-PLUS will converge. However, experimental results have demonstrated that it works very well in practice (Kschischang et al., 2001; Kok and Vlassis, 2006; Kuyer, Whiteson, Bakker, and Vlassis, 2008). Also, it is possible to use more recent variants of message passing that guarantee convergence (Globerson and Jaakkola, 2008). However, regular MAX-PLUS suffices for the setting considered in this chapter and therefore incorporating such modifications is deferred to future work.

To get regular MAX-PLUS to work in FGs with cycles, one needs to normalize the messages to prevent them from growing bigger and bigger. This is typically done by subtracting from each sent message the average of the incoming messages. Another technique that is applied to increase convergence rates is *damping:* taking a weighted sum of the newly calculated messages and the old ones.

5.5.4 Other Solution Methods for CGBGs

In essence the maximization over a factor graph as introduced in this section is a distributed constraint optimization (DCOP) problem (Modi et al., 2005). As such, any (approximate) algorithm for DCOPs can be used to find an (approximate) solution for the CGBG (Liu and Sycara, 1995; Yokoo, 2001; Modi et al., 2005; Pearce and Tambe, 2007). In particular, methods that are employed for transition and observation independent Dec-POMDPs may be reused for the solution of CGBGs (Nair et al., 2005; Varakantham et al., 2007).

Alternatively, it is possible to convert a CGBG G to a regular graphical BG (GBG) G' (Singh et al., 2004b) by defining an *individual* payoff function for each agent analogous to (5.1.7):

$$\forall_{i \in \mathcal{D}} \quad u_i(\theta_i, \mathbf{a}_{\mathcal{N}_i}) \equiv \sum_{e \text{ s.t. } i \in e} \sum_{\boldsymbol{\theta}_e} \Pr(\boldsymbol{\theta}_e | \theta_i) u_e(\boldsymbol{\theta}_e, \mathbf{a}_e). \tag{5.5.13}$$

The resulting GBG G' is non-collaborative and each agent tries to optimize its individual payoff function. However, each Bayes-Nash equilibrium will correspond to a local optimum of the original CGBG G. As such any solution methods for regular GBGs (Singh et al., 2004b; Soni et al., 2007) may also be used to find locally optimal solutions.

5.6 Algorithms

In this section, all the different components treated in this chapter are tied together. The CGBG approach to Dec-POMDPs matches seamlessly with the method for policy search in Dec-POMDPs using regular BGs as was described in Chapter 4. As such, this new work defines not a single algorithm, but a family of algorithms

Algorithm 5.4 `Expand`($\boldsymbol{\varphi}^t$)—Exploit last stage independence ELSI

if $t < h-1$ **then**
  {Perform regular MAA* for $t = 0,...,h-2$ }
  Φ^{t+1} = construct all policies $\boldsymbol{\varphi}^{t+1}$ as in Algorithm 4.2.
  return Φ^{t+1}
else
  {For $t = h-1$ exploit independence }
  G = ConstructCGBG($\boldsymbol{\varphi}^{h-1}$,$\boldsymbol{b}^0$)
  $\boldsymbol{\delta}^{h-1}$ = NDPSolve(G)
  $\boldsymbol{\pi} = (\boldsymbol{\varphi}^{h-1},\boldsymbol{\delta}^{h-1})$
  return $\boldsymbol{\pi}$
end if

to which we refer as FACTORED GMAA*. Within this family we discriminate between an optimal algorithm and approximate algorithms.

5.6.1 Optimal Methods: Exploiting Last-stage Independence

Computation of an optimal Q-value function is intractable. As such it is not practical to use FSPC for an optimal algorithm. Rather we will use k-GMAA* with $k = \infty$ (i.e., MAA*) to compute optimal policies.

This chapter did not prove for any of the proposed factored value functions that they are guaranteed overestimates, and as such, applying them in a MAA*-setting makes no immediate sense. Moreover, in order to guarantee finding the optimal joint policy it is required that *all* joint policies are expanded for intermediate stages, so efficiently solving the CGBGs for those stages also does not make direct sense: all joint BG policies will have to be constructed anyway.

However, for the last—and most computationally demanding—stage:

1. the factored optimal Q-value function is available: it is given by the immediate reward function.

2. only the best solution of the CGBG is needed (i.e., there is no need to expand all full-length joint policies).

Therefore we can modify GMAA* to exploit the last-stage independence (ELSI). The resulting method, GMAA*-ELSI, is described by Algorithms 4.1 and 5.4 jointly. For stages $0,\ldots,h-2$ regular MAA* is used with a non-factored Q-value function as described in Chapter 4. For the last stage $t = h-1$, a CGBG is constructed rather than a regular BG by using the factored immediate reward function as the factored payoff function. This CGBG is then optimally solved using NDP and the solution of this CGBG used to construct a full-length joint policy $\boldsymbol{\pi}$.

Even though GMAA*-ELSI may provide significant speed-up for some problems, it is not likely to scale to much higher number of agents, because in the intermediate stages no independence is exploited. It is non-trivial to extend the algorithm to also exploit independence in those stages because MAA* depends on

the generation of all intermediate policies in order to guarantee optimality. It might be possible to circumvent this problem by incrementally constructing the set of expanded policies. I.e., rather than *only* expanding the k heuristically best-ranked policies, it may be possible to *first* expand the k best-ranked policies and expand the rest later only when needed. This idea remains subject to further investigation.

5.6.2 Approximate Methods

Factored FSPC and Factored k-GMAA* are the approximate algorithms examined in this chapter. They are direct extensions of FSPC and k-GMAA* from Chapter 4, modified to construct and solve CGBGs rather than BGs. In particular, Factored k-GMAA* repeatedly solves CGBGs and returns the k best-ranked policies to be expanded in new partial policies. Factored FSPC is the case where $k = 1$.

Apart from k there are three other main dimensions that can be changed: the used scopes, the CGBG solver and the heuristic Q-value function. The desired scopes can be determined before-hand and determine the form of the Q-value function that will be used as a heuristic and thus the form of the corresponding CGBGs. Changing CGBG solvers is fairly trivial: any CGBG solution method, as well as regular BG methods may be used. For these methods we can use any of the methods to compute a factored Q-value function discussed in Section 5.4: we can compute a factored approximation of the $\mathrm{Q}_{\mathrm{MDP}}$ value function though naive regression (NR) and approximate dynamic programming (ADP), or compute an approximate Q_{TP} function through transfer planning.

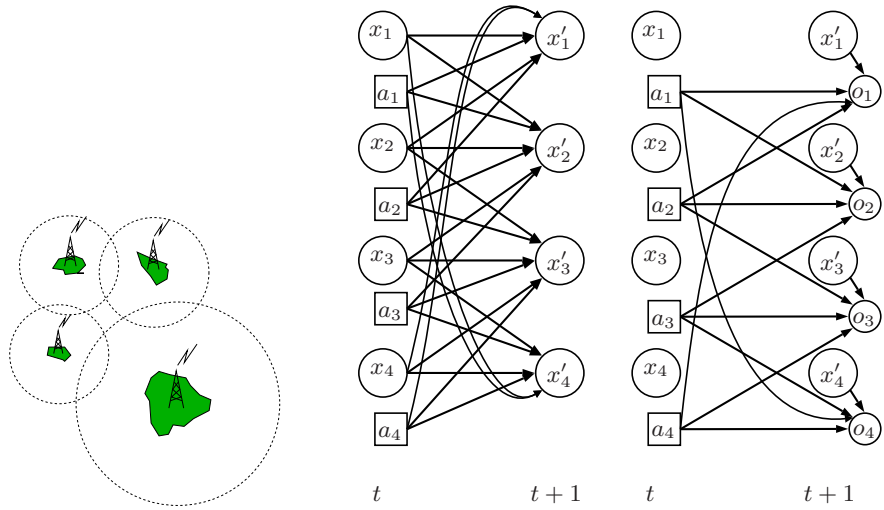
Since exact construction for CGBGs also becomes infeasible for larger problems, Factored FSPC and Factored k-GMAA* use approximate inference as detailed in Section 5.3.3. A side-effect of this, however, is that $V^{0\ldots t-1}(\boldsymbol{\varphi}^t)$, the past reward that a past joint policy $\boldsymbol{\varphi}^t$ achieves over stages $0, \ldots, t-1$ becomes an estimate.

5.7 Experiments

This section presents the results of an empirical evaluation of some members of the proposed Factored GMAA* family. First, Factored FSPC is compared against other state-of-the-art methods. Next, some of the approximations made in Factored GMAA* methods are analyzed.

5.7.1 Problem Domains and Experimental Setup

The empirical evaluation is performed on two problem domains: the FFG problem described in Section 5.1.2, and a new problem called Aloha, which is first introduced.



(a) An illustration of ALOHA.

(b) The DBN that represents ALOHA, split in two for clarity: left the transition model, right the observation model.

Figure 5.8: The ALOHA problem, showing the problem instance with four islands arranged in a square.

5.7.1.1 The Aloha Problem

The ALOHA problem can be seen as a generalization of the BROADCAST CHANNEL benchmark problem (Hansen et al., 2004), and was inspired by a similar problem domain defined in the context of single-agent POMDPs (Cassandra, 1998). The problem consists of a number of islands, each equipped with a radio tower that is used to transmit messages to its local population. Figure 5.8a shows an illustration of the ALOHA problem with four islands in a square configuration. Each island has a backlog of messages that it needs to send, and at each time step it can decide to send one message or to not send. Even though the messages of a radio tower are only of interest to the residents of that same island, given the proximity of some islands, communications from neighboring islands interfere, as indicated by the transmission ranges (dashed circles in Figure 5.8a). This means that when in a particular time slot two neighboring islands attempt to send, a collision occurs (and the messages will have to be resent). The left DBN in Figure 5.8b shows that each island's backlog is affected by the island's action, as well as by the backlogs and actions of neighboring islands. Each island starts with no packages in its backlog, and with probability 0.6 a new packet arrives (i.e., the backlog increases by 1), until the maximum backlog has been reached (set to 2 in these experiments). In general the size of an island may influence the frequency with which messages arrive, but we do not consider this here. The DBN on the right in Figure 5.8b shows the structure of the observation model. Each island can observe noisily whether the channel it shares with its neighbors was idle, had a successful transmission or a

collision occurred (it has a 90% chance of getting the correct observation, the other 10% is uniformly divided between the two other observations). For each packet that is present in the backlog of an island at $t+1$, the system receives a penalty of 1, i.e., $R^i(x'_i) = -x_i$. Therefore the goal of the system can be described as trying to minimize the number of waiting packets. Apart from the square configuration shown in Figure 5.8a, also several 'in-line' variants, where a number of islands is connected in a line configuration, are considered. Because the immediate reward functions depend on a back-projection similar to (5.1.2), and because the transition model is more densely connected than FFG, the immediate reward scopes of the considered ALOHA problems contain 3 agents.

5.7.1.2 Experimental Setup

The empirical evaluation is performed on systems with a 3.4 GHz Xeon processor and 4GB memory, running 64bit Debian GNU/Linux. Reported timing results are CPU times with a resolution of 0.01s. For all the evaluations reported below, the different methods are given 2 hours of wall-clock time within which they have to compute solutions for all considered horizons (typically $h = 2, \ldots, 5$ or 6).

In the experiments reported here, immediate reward scopes are used unless mentioned otherwise. Before computing the factored Q-value functions, these scopes are reduced, i.e., scopes that form a proper sub-scope of another one were removed as in Exampe 5.4.

In the experiments using the factored Q_{MDP} function computed through naive regression (NR), the flat transition, reward and observation model are cached, because NR makes full sweeps over the state space. This caching is only possible for relatively small problems. For ADP and Q_{TP} we do not perform such caching. To solve the linear systems in the factored Q_{MDP} methods we use the UMFPACK library (Davis, 2004), exploiting the sparsity of our systems. For the computation of Q_{TP} for the FFG problem, the 2-agent FFG is used as the source problem. The agent mapping function maps the lower agent index in a scope $\mathbb{A}(e,t)$ to agent 1 and the higher to agent 2, as described in Section 5.4.4. For the ALOHA problem, the 3-island in-line variant is used as the source problem. For the other 'in line' variants, we perform a similar mapping as for FFG: i.e., the lowest agent index in a scope is mapped to agent 1, the middle index is mapped to agent 2 and the highest index is mapped to agent 3. For both FFG and ALOHA we used the Q_{MDP} heuristic for the source problems.

The CGBGs are solved by running MAX-PLUS on the corresponding type-action factor graph. The implementation of MAX-PLUS makes use of libDAI (Mooij, 2008b). In the experiments MAX-PLUS runs until convergence with a maximum of 25 iterations and using a damping factor of 0.5. MAX-PLUS uses a sequential random message passing scheme and performs 10 restarts for each CGBG, also the other BG solution methods are restarted 10 times in each stage. The reported statistics are means over 10 restarts of GMAA*. Finally, 10,000 simulation runs are performed to evaluate the true value of the joint policies found.

	V^*	T_{GMAA^*}	$T_{\text{GMAA}^*\text{-ELSI}}$
$h = 2$	-5.213685	≤ 0.01 s	0.03 s
$h = 3$	-6.654551	0.15 s	0.46 s
$h = 4$	-7.462685	4834.07 s	12.67 s

Table 5.2: Results for the FFG for horizon 2, 3, and 4. Column V^* indicates the value of an optimal policy, T_{GMAA^*} the computation time for the plain GMAA* algorithm, and $T_{\text{GMAA}^*\text{-ELSI}}$ the computation time for our proposed method.

5.7.2 Comparison to Other Methods

Here we compare some instantiations of the proposed family of Factored GMAA* solution methods against existing algorithms. In particular we evaluate the performance of the optimal GMAA*-ELSI algorithm and Factored FSPC, and compare the Max-Plus solution method using type-action factor graphs against performing alternating maximization (AM) (Emery-Montemerlo et al., 2004). AM iteratively selects one agent to improve (maximize) its individual policy while keeping the policies of the other agents fixed, thus climbing to a local maximum.

5.7.2.1 Exact Results: Last-stage Independence

In contrast to the other results presented, the results for the experiment reported here are obtained on 32bit machines using 2GB of memory. Table 5.2 compares results for GMAA*-ELSI with regular GMAA* (that solves a regular BG also for the last stage) for FFG, introduced in Section 5.1.2. We used Q_{BG} as the heuristic Q-value function for all time steps of GMAA*, and for time steps $0 \ldots h-2$ of GMAA*-ELSI. As both methods are optimal, they compute identical optimal policies, whose values are shown in the first column. The remaining two columns show computation time of both methods. The timing results displayed are for the GMAA*-phase and do not include the time needed to compute the heuristic (which is the same for both algorithms).

For the low horizons, we can see that GMAA* is faster, as the Bayesian games to be solved are small, and not worth the overhead caused by the construction and solution of CGBGs and the bookkeeping of the non-serial dynamic programming used in GMAA*-ELSI's last time step. However, for horizon 4 a dramatic speedup is provided by exploiting the local interactions in this factored Dec-POMDP.

5.7.2.2 Evaluation of Factored FSPC

In order to evaluate Factored GMAA* and its scaling behavior with respect to the number of agents, it is compared against some other methods for (approximately) solving Dec-POMDPs. In particular it is tested against two regular, non-factored, GMAA* methods: FSPC and MAA*. We compare with direct cross-entropy (Dice) policy search (Oliehoek et al., 2008a), the only other method that has been demonstrated to work on Dec-POMDPs that are not transition and observation independent (TOI) with more than 3 agents.

The performance of FACTORED FSPC is evaluated using the three factored Q-value heuristics described in Section 5.4. For all three, fixed immediate reward scopes are used for all stages. Note that the scopes of the local reward function associated with the first and last house are sub-scopes of those associated with the house next to them. The result is that the number of components of the factored Q-value functions is $N_H - 2$, and in each of them 2 agents participate.

The (non-factored) FSPC algorithm uses alternating maximization with 10 restarts to solve the BGs. For DICE two parameter settings are employed: one that gives good results according to Oliehoek et al. (2008a) (DICE-normal), and one that should be faster (DICE-fast).[1]

Finally, as a baseline we also included the performance of the random joint policy in which each agent selects each action with uniform probability and the performance of the best joint policy in which each agent always selects a same fixed action. That is, the result of a brute-force search over all joint policies in which each agent is restricted to always select the same action. So the agents do not have to select the same action, but one particular agent should always perform the same action for all its possible histories.

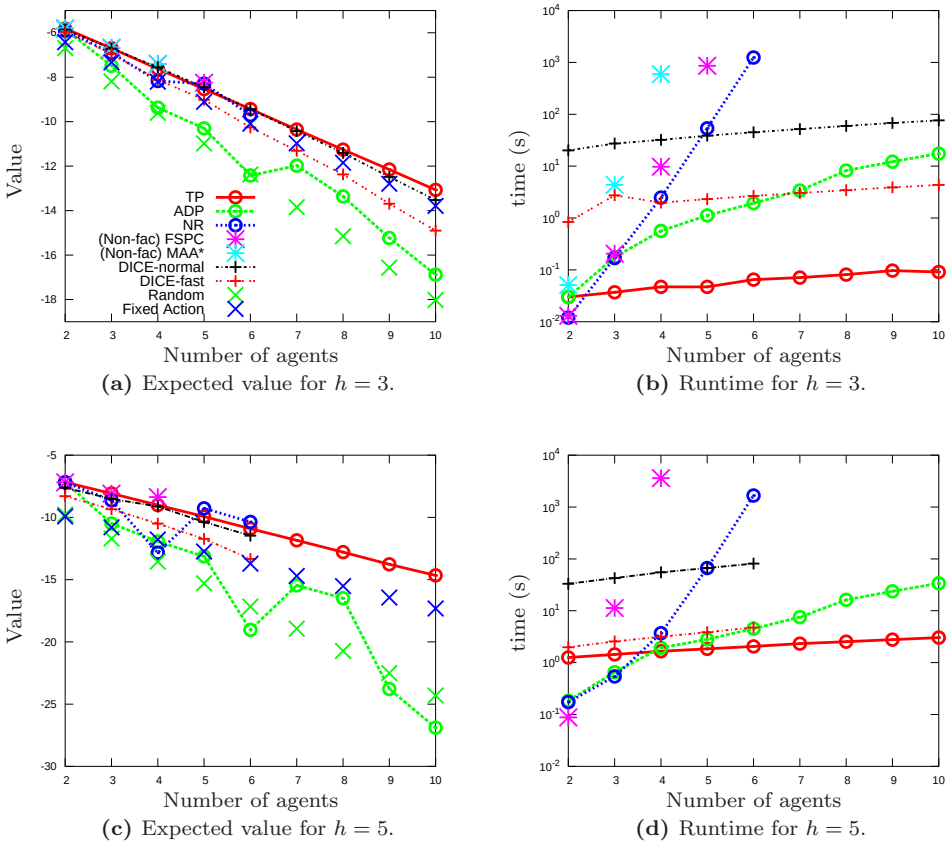
Figure 5.9 shows the results of the comparison on the FFG domain. We see that, although it does not always find the very best value, Q_{TP} performs very well: it achieves a value as good as or better than DICE-normal at a fraction of the time, and it achieves a value significantly higher than DICE-fast and is faster. The figure shows that ADP is also fast and has good scaling w.r.t. time, but finds bad policies, worse than random in some cases and generally not better than the fixed action baseline. Naive regression, on the other hand, generally achieves reasonable to very good values, except for 4 agents.[2] However, it scales bad w.r.t. the number of agents as confirmed in the runtime plots. MAA* is able to compute the optimal solution only for $h = 3$ with 2–4 agents. For these settings, we see that FSPC, DICE-normal and Q_{TP} also perform (near-)optimal. DICE-normal performs fairly well with respect to the achieved value, especially for $h = 3$, for $h = 5$ the quality of the found policies is somewhat less than the solutions found by Q_{TP}. However, in terms of runtime DICE-normal performs poorly. For DICE-fast we see the opposite: it has somewhat poor performance in terms of value, but much better runtime results. Note that it is outperformed by the fixed action baseline for $h = 3$ as shown in Figure 5.9a.[3] In general, runtime of the DICE methods scales well with respect to the number of agents. However they run out of memory for more than 7 agents for $h = 5$.

We ran similar experiments for the Aloha problems. Figure 5.10 shows the

[1] Both variants perform 10 restarts, use a learning rate of 0.2 and perform what Oliehoek et al. (2008a) refer to as 'approximate evaluation' of joint policies. DICE-normal performs 300 and DICE-fast 100 simulations per joint policy. DICE-normal performs $I = 50$ iterations, in each of which $N = 100$ joint policies are sampled of which $N_b = 5$ policies are used to update the maintained distribution. DICE-fast uses $I = 15$, $N = 40$, $N_b = 2$.

[2] Further investigation revealed that the resulting linear system is badly conditioned and the found solution is poor.

[3] Of course, for shorter horizons, the fixed action baseline is relatively good: only after been in the same house for some stages, the expected value of switching becomes higher, since in this house, the probability of having fire becomes near zero.

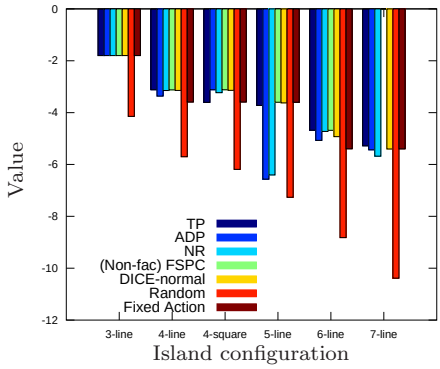


(a) Expected value for $h = 3$.
(b) Runtime for $h = 3$.
(c) Expected value for $h = 5$.
(d) Runtime for $h = 5$.

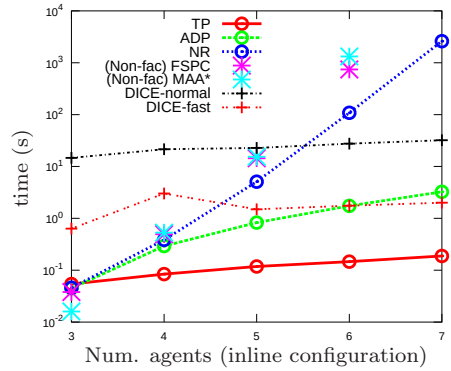
Figure 5.9: A comparison of FACTORED FSPC with different heuristics and other methods on the FFG problem.

results of this comparison. In the value plots we omit the results for MAA* and DICE-fast; FSPC achieved the same value for these experiments as the former and DICE-normal performed better than the latter. Again, we see that Q_{TP} performs very well, both in terms of quality and speed: it finds near-optimal solutions for all 'in-line' instances ($h = 2$, 3–6 islands) for which it was possible to compute the optimal policy (remember that the FSPC results illustrated in Figure 5.10a achieved the same value as MAA*). Also, these experiments confirm the results found for FFG in that ADP is fast but can result in bad quality policies, NR performs better but scales poorly and DICE performs adequately (for $h = 2$,3). However, we also see that DICE is the only method to complete the 5-island problem for $h = 4$. For more agents, it runs out of memory. For $h = 4$, FACTORED FSPC is quite slow. Even though the increase in runtime from 3 to 4 agents is less dramatic than for FSPC, the runtime for 3 agents is already quite high.

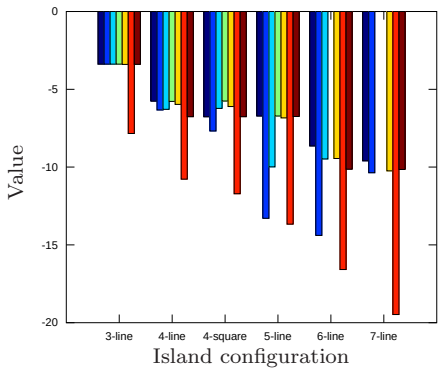
Further analysis revealed that the difficulty is that the number of local joint



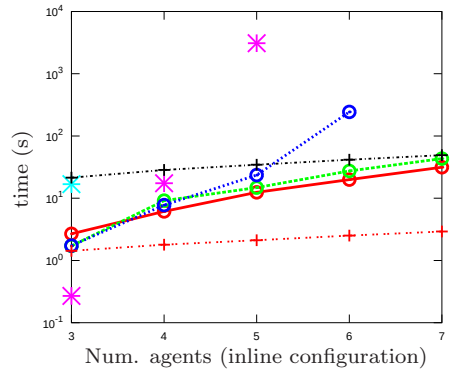
(a) Expected value for $h = 2$.



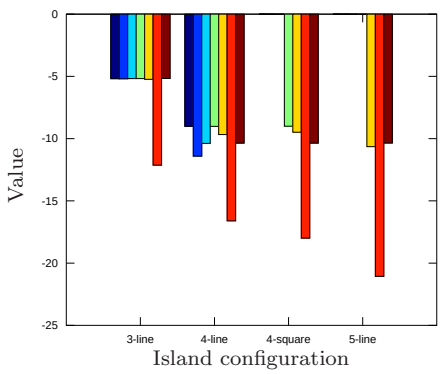
(b) Runtime for $h = 2$.



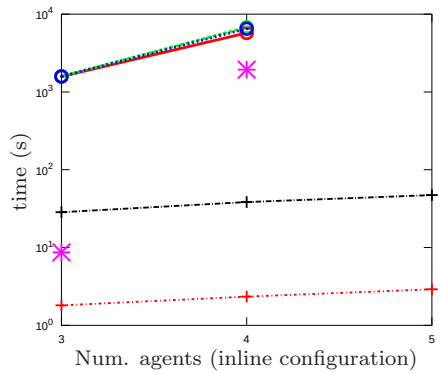
(c) Expected value for $h = 3$.



(d) Runtime for $h = 3$.

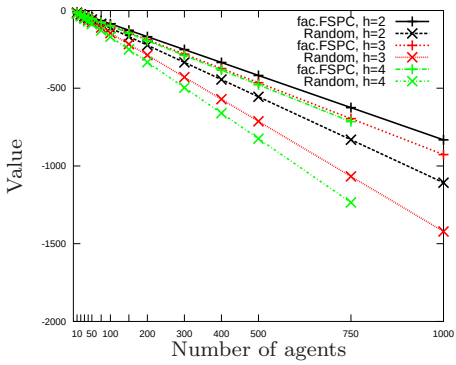


(e) Expected value for $h = 4$.

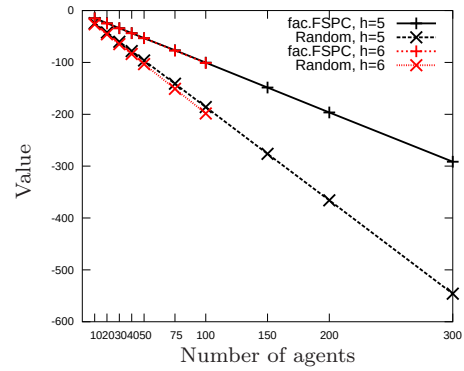


(f) Runtime for $h = 4$.

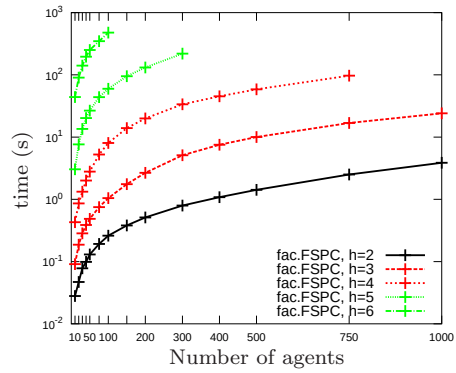
Figure 5.10: A comparison of FACTORED FSPC with different heuristics and other methods on the ALOHA problem. Plots on the left-hand side show expected values for all tested configurations. Plots on the right-hand side show runtime results for the 'in line' configurations.



(a) Expected value for $h = 2, \ldots, 4$.



(b) Expected value for $h = 5,6$.

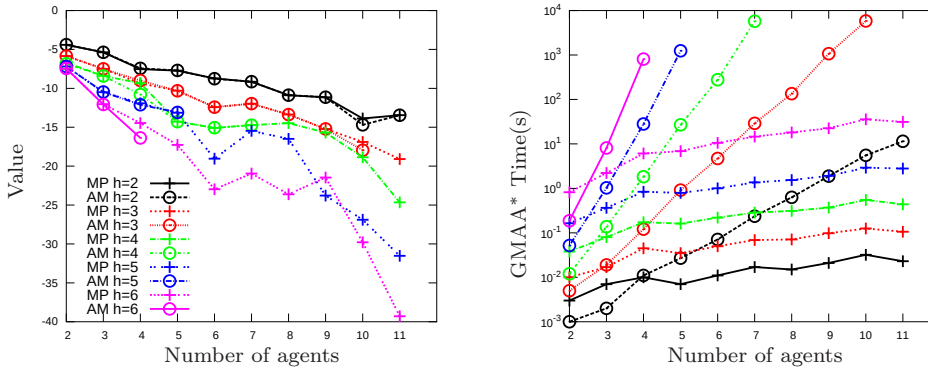


(c) Runtime for $h = 2, \ldots, 6$.

Figure 5.11: Factored FSPC results for large number of agents.

types for each edge grows quickly in the Aloha problem. In these problems the immediate reward scopes contain 3 agents and these agents have 3 observations. This leads to $3^3 = 27$ observation histories (and thus type-action variables) per agent and $27^3 = 19{,}683$ local joint types (and thus factors) associated with each edge of a CGBG for the last stage $t = 3$. So the fact that Factored FSPC performs slow, is the result of the combination of larger immediate reward scopes together with the fact that the proposed methods do not scale well with respect to the horizon, unless some other techniques (such as clustering of the types, see Emery-Montemerlo et al. (2005) and Chapter 6 of this thesis) are applied. A last note is that the fixed action baseline performs quite well for some instances.

Since Factored FSPC using Q_{TP} performed very well and the computation of the heuristic for the FFG problem does not scale with the number of agents, Q_{TP} is used to evaluate the results on much larger numbers of agents. The results are shown in Figure 5.11. In particular, Figure 5.11a shows that it is possible to compute solutions for up to 1000 agents for $h = 2{,}3$ and 750 agents for $h = 4$. For $h = 5$ the maximum number of agents is 300 and even for $h = 6$ is is possible



(a) Expected value. **(b)** Runtime.

Figure 5.12: MAX-PLUS compared against AM of the FFG problem.

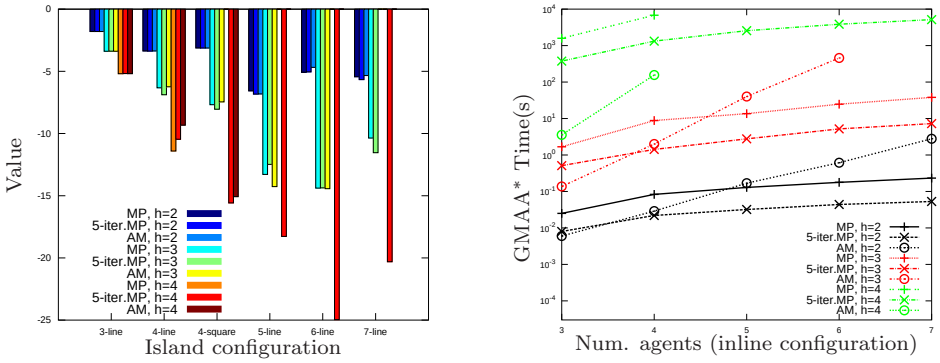
to compute a solution for the 100-agent problem, as shown in Figure 5.11b. An interesting detail is that for the computed entries for $h = 6$ the expected value is roughly equal to $h = 5$. This implies that probability of any fire remaining at stage $t = 5$ is close to zero. The reason the methods could not scale up any further is not computation time, but insufficient memory. The runtime results are shown in Figure 5.11c and increase linearly with respect to the number of agents. Note that the fixed action baseline is not included in Figure 5.11, as that becomes intractable: it performs simulation of all considered fixed action joint policies and these simulations become relatively expensive for many agents. Moreover, the number of fixed action joint policies grows exponentially with the number of agents.

5.7.2.3 Max-Plus vs. Alternating Maximization

In order to test the efficiency of the proposed MAX-PLUS algorithm to solve CGBGs, it is compared against AM, which has been the standard for approximately solving BGs for Dec-POMDPs (Emery-Montemerlo et al., 2004, 2005). The results reported in this section are obtained via FSPC using the ADP heuristic with immediate reward scopes. To present a clear comparison, the timing results are shown for the GMAA* phase in which the CGBGs are actually solved, excluding times for initialization and computation of the Q-value function.

For the FFG problem the results are shown in Figure 5.12. Figure 5.12a shows that the expected value of MAX-PLUS and AM is roughly the same. Figure 5.12b, however, shows that for $n > 2$, AM is significantly outperformed for all horizons but horizon 2. In contrast, for the 2-agent case, AM actually performs almost 10 times better than MAX-PLUS. This can be understood by the fact that in this case, there actually is only one component of the Q-function, and thus the problem is not factored at all. As such there is not enough independence to be exploited by MAX-PLUS.

The same comparison is performed for the ALOHA problems. Figure 5.13a



(a) Expected value. **(b)** Runtime.

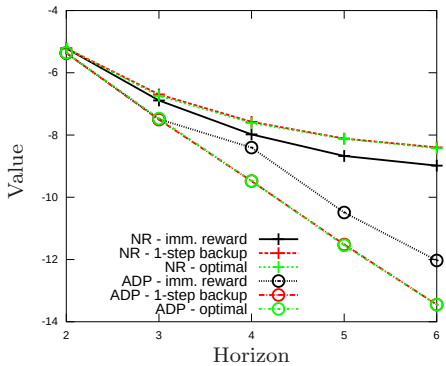
Figure 5.13: MAX-PLUS compared against AM on the 'in line' ALOHA problems.

shows that also for these problems, the solution methods are roughly equal in solution quality, although there are some differences at some points. The runtime results, shown in Figure 5.13b tell a more subtle story. We see that MAX-PLUS is significantly outperformed for three islands. This can be expected because, as in 2-agent FFG, this problem only has one Q-function component with all 3 agents in its agent scope and thus has no independence to exploit. However, also for the two 4-island problems, we see that MAX-PLUS is outperformed by AM, especially for $h = 4$. Apparently the amount of independence is not yet enough to pay off in MAX-PLUS. For the 5–7 islands problems of $h = 2{,}3$ however, MAX-PLUS performs better than AM, because its runtime grows more gradually with the number of agents. For $h = 4$, none of the methods was able to compute results for these larger problems.

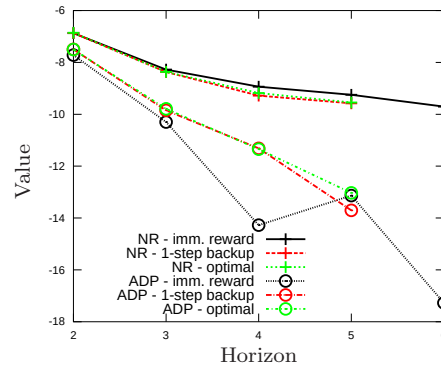
As mentioned before, the problem is that the number of local joint types (i.e, joint observation histories) grows very fast for the ALOHA problems. Further analysis revealed that for these large CGBGs, the messages of MAX-PLUS do not converge within 25 iterations. However, for these large CGBGs the maximizing configuration is usually found very fast: within a few iterations. As such, is possible to speed up MAX-PLUS by reducing the number of iterations. Figure 5.13 also includes results for MAX-PLUS with 5 iterations. This variant runs a constant factor faster than regular MAX-PLUS, which allows it to compute results for all problems, without paying a huge penalty in estimated value.

5.7.3 Analysis of Factored GMAA* Methods

Here we report experiments that perform an analysis of different choices one can make in FACTORED GMAA* methods.



(a) Expected value for varying scopes for 3-agent FFG.



(b) Expected value for varying scopes for 5-agent FFG.

Figure 5.14: A comparison of different scopes is non-conclusive.

t	e	$\mathbb{A}(e,t)$	$\mathbb{X}(e,t)$
$h-3$	1	{1,2,3,4,5}	{1,2,3,4,5,6}
$h-2$	1	{1,2,3,4}	{1,2,3,4}
	2	{2,3,4,5}	{2,3,4,5,6}
$h-1$	1	{1,2}	{1,2,3}
	2	{2,3}	{2,3,4}
	3	{3,4}	{3,4,5}
	4	{4,5}	{4,5,6}

Table 5.3: Optimal scopes for 5-agent FFG. For stages $t \leq h-3$ the problem becomes fully coupled.

5.7.3.1 Comparing Scopes

In order the establish the influence of using different scopes, using immediate reward scopes was compared to using larger scopes, but the results are inconclusive. The general findings are that, almost always the value found for optimal scopes (OS) and 1-step back projected scopes (1SB) are the same. This can be understood by realizing that for many problem instantiations OS and 1SB actually specify exactly the same scopes: starting from only 5 agents in FFG is 1SB not fully coupled. This means that only for $n \geq 5, h \geq 3$ OS and 1SB actually specify a different scopes structure, which allows for different values as illustrated by Table 5.3. Still Figure 5.14b shows that also for these settings, the performance is often the same. Furthermore, the performance of varying scopes under ADP was unpredictable. For NR using optimal or 1SB scopes usually performed better than immediate reward scopes, as in Figure 5.14a. However, in some cases we also found the opposite, for instance as shown in Figure 5.14b.

5.7.3.2 k-GMAA* Results

This subsection reports on experiments performed to investigate the added performance of FACTORED k-GMAA*. Figure 5.15 shows the expected value for $k = 10$ compared to FACTORED FSPC (i.e., $k = 1$) for different FFG problems. The conclusions that can be drawn from these plots is that in some cases using k-GMAA* can significantly improve the results, but not always, and the improvement usually is not enough to reach the best found value (i.e., the result of the best heuristic with $k = 1$). There are two reasons that can explain this behavior:

1. MAX-PLUS typically only finds a few solutions, especially for the smaller CGBGs (for earlier stages). Therefore some branches of the search space are never explored. I.e., MAX-PLUS is very fast and results in good policies when the heuristic is good. However, when the heuristic is bad, only limited back tracking will take place and chances are that the value is not improved substantially.

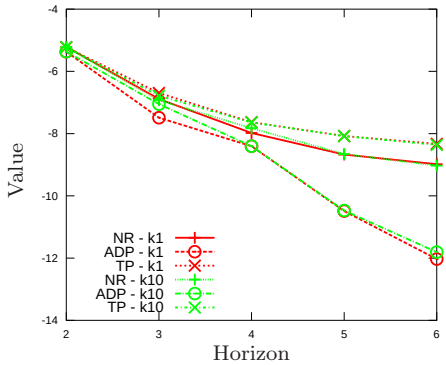
2. The used factored Q-functions are not guaranteed to be over-estimations (i.e., it is not certain they are admissible heuristics). As such parts of the search space containing (near-) optimal policies may be pruned.

A more detailed analysis per heuristic follows below.

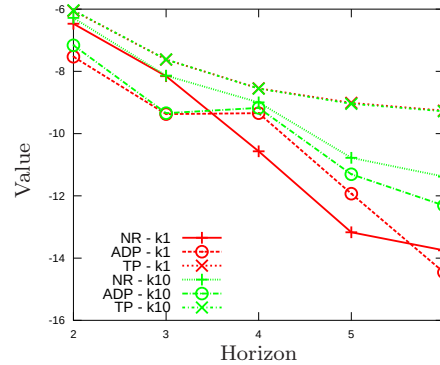
ADP Figure 5.16 shows the influence of k on FACTORED FSPC using ADP. Since the runtimes increase with k, it is clear that that more policies evaluated. As such the influence of reason 2 is perhaps minor. This is to be expected as the component-wise maximization typically renders the heuristic admissible, although we have not proven so. Since ADP still does not reach very good values in many cases, we suspect that the influence of reason 1 is very important for this heuristic. I.e., because MAX-PLUS finds only a few (perhaps just 1) solution for the CGBGs of the first stages, the heuristic search never considers parts of the joint policy space which contains much better policies.

NR The comparison for naive regression gives similar results to ADP. The runtimes are slightly lower, which indicates that more pruning takes place, which can be explained by two things. First, FACTORED FSPC with NR typically finds better policies, and thus a higher lower bound, which allows for more pruning. Second, the Q-function found by NR does not overestimate by performing maximization over the components. As such, this heuristic may give less of an overestimation, and in fact an underestimation of the value. I.e., reason 2 may also apply here.

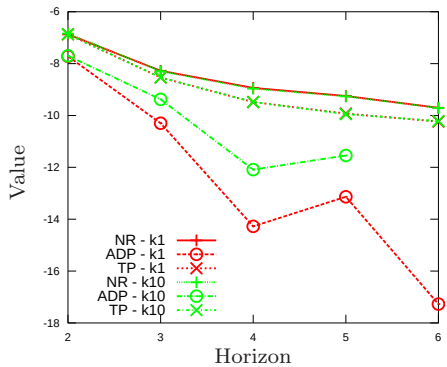
Q_{TP} A similar evaluation for Q_{TP} resulted in values that are (near-)identical, and no significant differences in run times. Q_{TP} is not guaranteed to be an admissible heuristic and in our experiments definitely is not. In particular, the effect of applying the underlying value function of a source problem is that the influence of reward function components is overcounted. Since the rewards in the problems considered are negative, this means that we are overcounting negative rewards, and



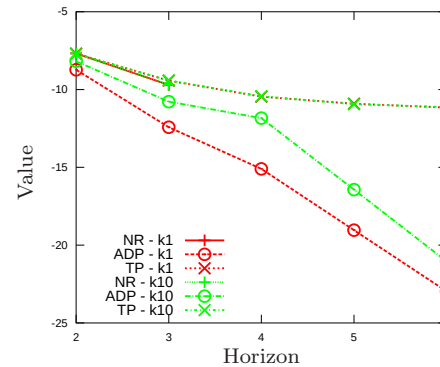
(a) Expected value for 3-agent FFG.



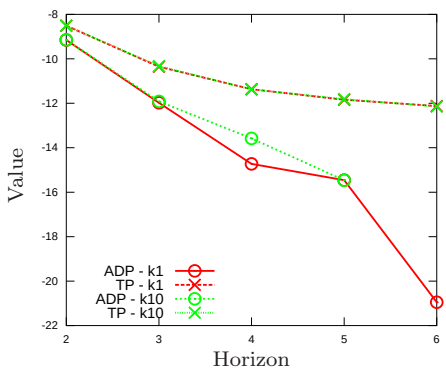
(b) Expected value for 4-agent FFG.



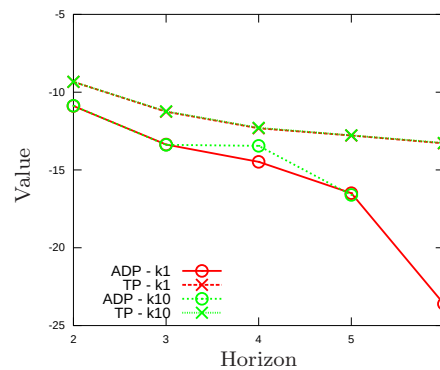
(c) Expected value for 5-agent FFG.



(d) Expected value for 6-agent FFG.

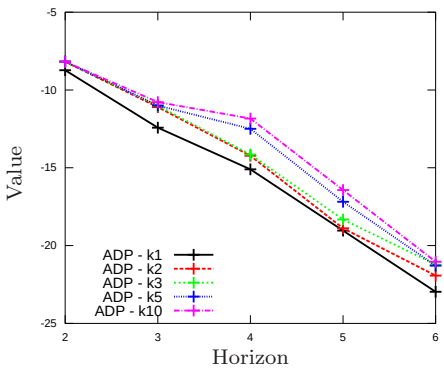


(e) Expected value for 7-agent FFG.

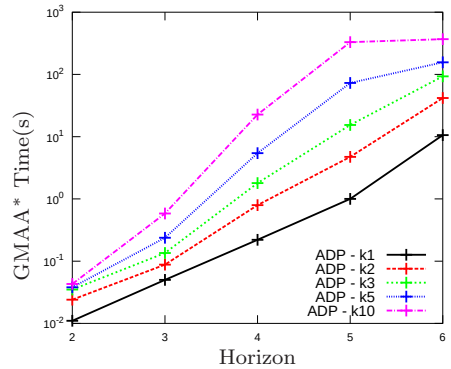


(f) Expected value for 8-agent FFG.

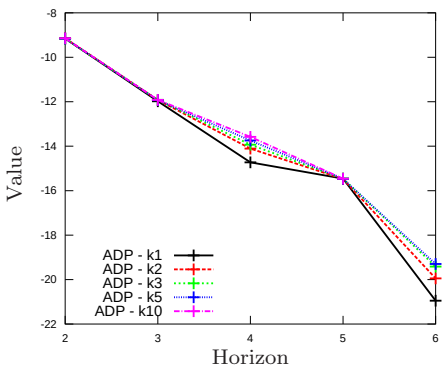
Figure 5.15: Comparison for different k on the FFG problem for different heuristics. Higher k may result in better policies, but not always so.



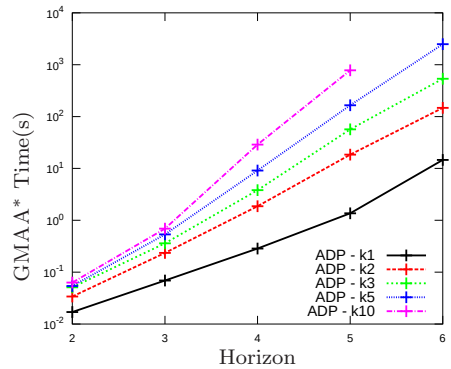
(a) Expected value for 6-agent FFG.



(b) Runtime for 6-agent FFG.

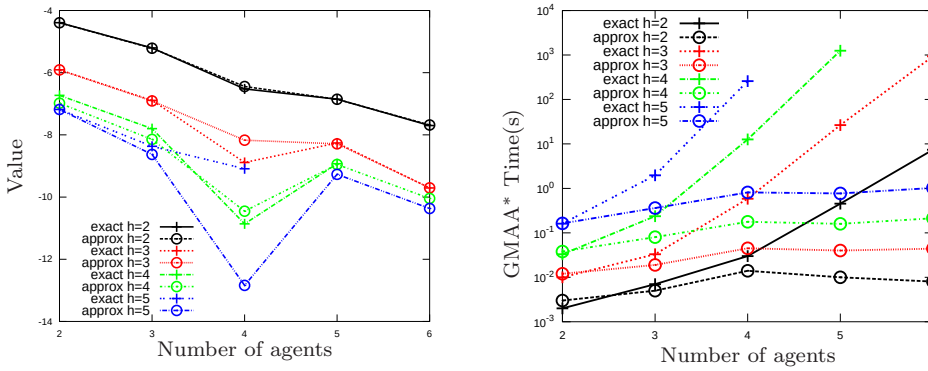


(c) Expected value for 7-agent FFG



(d) Runtime for 7-agent FFG.

Figure 5.16: A typical comparison for different k on the FFG problem when using ADP to compute the heuristic. Although higher k do results in searching more policies, and in some cases this also results in a significant better policy, this is not always the case.



(a) Expected value. **(b)** Runtime of GMAA* phase.

Figure 5.17: Comparing the influence of exact vs. approximate inference.

thus underestimating the value. As a result, many policies are pruned rather than expanded. This effect could be reduced by trying to apply some scaling to the Q_{TP} value function. Alternatively, it is possible to estimate how much the underestimation is and introduce a slack variable that indicates how much worse a policy needs to be than the best found policy before it is allowed to be pruned.

5.7.3.3 Exact vs. Approximate Inference

In order to test the influence of using approximate inference, we compare against a variant of FACTORED FSPC that uses exact inference. For this comparison the NR heuristic is used, because it gives fairly good results while depending on $\Pr(\mathbf{x}_e, \vec{\boldsymbol{\theta}}_e)$, i.e., on inference of *both* the states factors and histories to compute the resulting heuristic.[1]

Figure 5.17 shows the results of the comparison. Figure 5.17b clearly shows that exact inference indeed suffers from exponential scaling with respect to the runtime and that approximate inference behaves much better. Figure 5.17a shows that this increase in runtime efficiency comes at little cost: typically there is little or no loss in value, except for 4 agents. As mentioned earlier, for 4 agents the resulting linear systems become ill conditioned and the solution makes little sense: there are quite a few local Q-values that are really large. Thus, a change in inference can have a dramatic impact on the solution. However, Figure 5.17 also shows that this dramatic change can go in either direction: we see that for $h = 5$ exact inference is much better, but approximate inference actually performs better for $h = 3{,}4$.

[1] Q_{TP}, in contrast only uses the probabilities of local joint types $\Pr(\vec{\boldsymbol{\theta}}_e)$ and therefore we expect the influence of approximate inference to be less.

5.8 Summary and Conclusions

This chapter formalized the interaction of several agents under uncertainty as a factored Dec-POMDP with an additively factored immediate reward function. For such models, the structure of the immediate reward function is not preserved in the value function, unless assuming transition and observation independence (TOI). Without this assumption, the scopes of the factored value function grow when going back in time, and will become fully coupled at some point. Nevertheless, it was shown that locality of interaction holds at each stage *given* the scopes of the value function for that stage. This analysis strengthens our belief that general Dec-POMDPs should be tackled per stage, rather than trying to find individual full-length policies in an iterative fashion.

Subsequently, the chapter showed how a factored Dec-POMDP can be represented by a series of collaborative graphical Bayesian games (CGBG). These have a compact representation that can be exponentially smaller than that of regular Bayesian games. However, constructing them exactly requires the probabilities $\Pr(s,\vec{\boldsymbol{\theta}})$, which means that it is necessary to perform exact inference over the space of states and joint histories, whose size is exponential in the number of state factors and the number of agents respectively. To counter this source of intractability we proposed to construct them using approximate inference. In particular, intermediate steps of computation of the factored frontier algorithm (Murphy and Weiss, 2001) were exploiting to construct the CGBGs.

To use CGBGs in the solution of factored Dec-POMDPs, a factored Q-value function is needed as payoff function for the CGBGs. Using the optimal Q-value function is implausible for two reasons. First, it is intractable to compute. Second, because its scopes grow when moving away from the last stage it becomes fully coupled. This means that the CGBGs for these earlier stages reduce to regular BGs which are exponentially larger and therefore much harder to solve. To overcome these problems, we proposed to compute approximate value function *given a predetermined scope structure* that specifies the scopes for each stage for each component of the factored Q-value function. Two methods to compute a factored approximation of the $\mathrm{Q}_{\mathrm{MDP}}$ (underlying MDP) value function were considered: naive regression (NR) and approximate dynamic programming (ADP). The former first computes the non-factored $\mathrm{Q}_{\mathrm{MDP}}$ value function and then finds the Q-value of the desired factorization that is closest in Euclidean sense. ADP does not first compute the centralized $\mathrm{Q}_{\mathrm{MDP}}$ value function, but rather bootstraps its approximation for stage t from the approximation already computed for stage $t+1$. The projection to factored value functions of a predetermined scopes can be performed by defining *induced indicator basis functions,* and this projection is particularly efficient because the inner product can be computed very efficiently (shown in appendix C). A third way of computing an approximate Q-value function proposed in this chapter is *transfer planning* (TP). TP computes an approximate (e.g., $\mathrm{Q}_{\mathrm{MDP}}$) value function for a smaller source problem that involves a smaller number of agents and use this value function as a component of the factored Q-value function of the original problem (the target problem). We refer to the resulting approximate Q-value function as Q_{TP}.

Given the probabilities $\Pr(s,\vec{\boldsymbol{\theta}})$ and a payoff function for the CGBGs, the representation of a factored Dec-POMDP as a series of CGBGs is complete. To efficiently solve the CGBGs it is possible to employ non-serial dynamic programming that scales exponentially in the induced width of the interaction graph (w). In the worst case, w is equal to the number of agents (n), but in typical settings the interaction graphs are sparse and $n \gg w$ yielding an exponential speedup. Another option is to convert a CGBG to a *type-action factor graph* and to use Max-Plus—a method based on belief propagation that finds a maximum through message passing—to find an approximate solution.

Making use of the components defined and building upon Chapter 4, it is possible to define a family of algorithms, which we refer to as Factored GMAA*. GMAA*-ELSI is an optimal algorithm and approximations can be found using Factored FSPC and Factored k-GMAA* using the approximate Q-value functions mentioned above as heuristic.

An empirical evaluation showed that for 3 agents GMAA*-ELSI obtains a speedup of two orders of magnitude compared to regular GMAA* that does not exploit last-stage independence. To the author's knowledge, these were the first experimental results for general factored (i.e., non-TOI) Dec-POMDPs with more than 2 agents. Also, Factored FSPC was compared against the state-of-the-art methods for solving Dec-POMDPs.

The remainder of the evaluation analyzed some of the proposed approximations and also tested the influence of increasing k in Factored k-GMAA*. The results showed that Factored GMAA* methods are efficient and scale well with respect to the number of agents, but that the quality of the found solutions depends very much on the used heuristic. In particular, the ADP achieved disappointing results. Since, NR generally did perform much better, this is likely to be caused by the independent maximization of the components (perhaps in combination with bootstrapping). However, also NR was in some cases outperformed by transferring the Q_{MDP} value of smaller source problems. This suggests that for the task at hand there may be a better similarity measure than Euclidean distance. I.e., it might be better to have a factored Q_{MDP} value function, that is not globally closest to the non-factored Q_{MDP} value function in Euclidean sense, but that better preserves structure at a local level. In general, though, NR found good solutions, but the computation of this heuristic itself does not scale well. However, Q_{TP} consistently performed very well, finding (near-) optimal policies in little time. Comparing to Dice, the only other method that has been demonstrated on non-TOI Dec-POMDPs with more than 3 agents, Factored FSPC found equal or better policies in similar or less time. In particular using Q_{TP} as the heuristic it was possible to compute good policies for up to 1000 agents in a matter of seconds. A comparison between Max-Plus and alternating maximization to solve CGBGs revealed that Max-Plus scales much better with respect to the number of agents. However, it is very sensitive to the number of local joint types and therefore scales badly with the horizon and the size of the agent scopes. As a result, AM still outperforms Max-Plus for larger horizons with few agents. The influence of using approximate inference seems negligible, while it avoids an exponential dependence of the runtime on the number of agents and state factors. The influence of us-

ing larger scopes was also investigated, but the results were inconclusive. The investigation of k-GMAA* revealed that larger k typically finds better policies, but because the heuristics are no longer guaranteed over-estimations and because Max-Plus only finds few solutions, the amount of improvement is usually limited.

5.9 Discussion and Future Work

The work presented in this chapter provides scaling only with respect to the number of agents, not with respect to the horizon. In order to allow scaling to larger horizons, the proposed methods should be combined for instance with pruning (Emery-Montemerlo et al., 2004) or clustering of histories (Emery-Montemerlo et al., 2005; Oliehoek et al., 2009). Chapter 6 will further discuss such clustering methods for regular BGs and how they can be applied in GMAA*. Extending these methods to CGBGs and incorporating these clustering techniques in Factored GMAA* is non-trivial and an important branch of future work.

Other future work should investigate other methods to compute factored Q_{MDP} approximations. A first starting point would be to replace the independent maximization step in ADP by a sample-based projection similar to the approach of Szita and Lörincz (2008). Also, we hope to develop methods to compute factored Q_{POMDP}, Q_{BG} approximations. Since the use of tighter heuristics can be quite beneficial as was demonstrated in Chapter 4, the efficient computation of such value functions is especially important.

Another direction of future work would be to consider dynamically changing interaction graphs, as for instance considered by Guestrin, Venkataraman, and Koller (2002b); Kok and Vlassis (2006). An added difficulty in our setting is that the environment it partially observable. Perhaps this problem may be overcome by adding a particular state variable that determines the shape of the interaction graph and can be observed by all agents.

The theory of Dec-POMDPs could be extended by unification of the methods described here with the SPIDER algorithm (Varakantham et al., 2007). On the one hand, GMAA* and its variants search over joint policies which are partially specified with respect to time (e.g., a policy only specified for the first stage). On the other hand, SPIDER searches over joint policies that are partially specified w.r.t. the individual agents (e.g., a joint policy only specifying an individual policy for agent 1). Factored GMAA* takes a step towards this unification: intuitively the GMAA*-phase corresponds to the former, while solution of the CGBGs (e.g., with NDP) corresponds to the latter. Future algorithms may truly unify the two approaches.

Finally, the newly identified method of transfer planning seems to open up an exciting new direction of research. In particular, our approach as proposed is closely related to *transfer via inter-task mappings* by Taylor et al. (2007) and it may be possible to consider transferring from tasks where there are different action and/or observation spaces by building on their work. Another interesting idea is automatic identification of source tasks and investigating methods for automatically finding the mapping between agents in the source and target tasks, for instance

using qualitative DBNs (Liu and Stone, 2006). A last question in this direction is whether it is possible to perform some scaling or transforming of the Q_{TP} functions such that they become admissible.

Chapter 6

Lossless Clustering of Histories

The GENERALIZED MAA* (GMAA*) algorithm introduced in Chapter 4 can find (optimal) solutions for Dec-POMDPs by repeatedly solving Bayesian games (BGs) for different stages. However, the cost of solving these BGs grows exponentially with the number of agents n and doubly-exponentially with the horizon h. That is, for a BG for the last stage, the number of joint policies, and thus the cost of optimally solving it, is expressed by (5.0.1), repeated here for convenience:

$$O\Big(|\mathcal{A}_*|^{n(|\mathcal{O}_*|^{h-1})}\Big),$$

where $\mathcal{A}_*$ and $\mathcal{O}_*$ denote the largest individual action and observation sets. Chapter 5 introduced techniques to provide scaling with respect to the number of agents. This chapter addresses scaling with respect to the horizon.

In a BG for a stage t, an individual type corresponds to an action-observation history (AOH). However, given that a deterministic past joint policy $\boldsymbol{\varphi}^t$ is followed, it also corresponds to a single observation history (OH), because the actions can be inferred from $\boldsymbol{\varphi}^t$. The problem is that the number of individual types (i.e., the number of individual observation histories) grows exponentially over time and thus the BGs also grow exponentially.

To counter the exponential growth of the BGs this chapter proposes to cluster the individual AOHs in a way that does not compromise solution quality. It is straightforward to exploit this result in optimal policy search. In particular, Section 6.4 shows how to cluster histories within the GMAA* algorithm.

This chapter empirically demonstrates that the proposed technique can provide a speed-up of multiple orders of magnitude, allowing the optimal solution of significantly longer horizons. For instance, we solve the well-known benchmark decentralized tiger (DEC-TIGER) problem for horizon $h = 5$ (in which case there are $3.82e29$ joint policies) and the BROADCASTCHANNEL problem for horizons as large as $h = 20$. To the best of our knowledge, such results were not obtainable previously. Empirical analysis of the generality of the clustering method suggests that it may also be useful in other (approximate) Dec-POMDP solution methods.

6.1 Clustering Types in BGs

The idea of clustering histories in BGs is not new. Emery-Montemerlo et al. (2004) already proposed to prune types with low probabilities. In subsequent work Emery-Montemerlo et al. (2005) replaced this pruning by clustering types, based on the profiles of the payoff functions of the BGs, thereby increasing the quality of the found policies. However, since the payoff functions are heuristics, this method is somewhat ad-hoc: even when providing a bound on the error of clustering two types in a BG, as long as this bound is with respect to the heuristic payoff function, this will guarantee nothing with respect to the optimal solution of a Dec-POMDP. As such performing clustering based on the heuristic payoff function causes an increased dependence on heuristic, while providing no guarantees whatsoever.

This chapter also considers clustering of AOHs. In contrast, however, it does not consider a lossy clustering scheme based on the heuristic payoff function $\widehat{Q}$ of the BGs. Rather, a criterion for clustering AOHs is introduced based on the probability these histories induce over histories of the other agents and over states. The nice thing of this criterion, which we refer to as *probabilistic equivalence (PE)*, is that clustering histories that satisfy this criterion is *lossless*: the solution for the clustered BG can be used to construct the solution for the original BG and the values of the two BGs are identical. Thus, the criterion allows for clustering of AOHs in BGs that represent Dec-POMDPs without compromising solution quality, i.e., optimality is preserved.

The main contribution of this chapter is that it established that when two histories in a Dec-POMDP are PE, they can be clustered together without loss in value. In the proof, some other contributions are made. In the following we outline the proof and these other contributions.

Section 6.2 introduces two concepts: reduction of BGs through commitment and best-response equivalence for BGs. The former states that *if* an agent is committed to select the same action for two of its types, the BG can be reduced by clustering these types. The latter says *when* a rational agent is committed to select the same action for two of its types, namely when those types are guaranteed to have the same best-response action, i.e., if if the two types are *best-response equivalent* (BRE). These results, although applied in the context of Dec-POMDPs in this chapter, may be useful more generally.

Next, Section 6.3 applies these results in the Dec-POMDP context. In particular, it formally introduces probabilistic equivalence and demonstrates that if it holds for two histories, then they are BRE. This is proven by showing that if PE holds, the two conditions necessary for BRE, as will be identified by Lemma 6.1, hold.

6.2 Best-Response Equivalence for BGs

This section considers Bayesian games as introduced in Section 2.1.1.2 and investigates when it is possible to cluster individual types in BGs.

Theorem 6.1 (Reduction through commitment)**.** *Given that in a Bayesian game B agent i is committed to select a policy that assigns the same action for two of its types θ_i^a,θ_i^b, i.e., to select a policy β_i such that*

$$\beta_i(\theta_i^a) = \beta_i(\theta_i^b), \tag{6.2.1}$$

then the BG can be reduced to a smaller one without loss in value for any of the agents. I.e., the two types can be substituted by a new type θ_i^c such that

$$\forall_{\boldsymbol{\theta}_{\neq i}} \ \Pr(\theta_i^c,\boldsymbol{\theta}_{\neq i}) = \Pr(\theta_i^a,\boldsymbol{\theta}_{\neq i}) + \Pr(\theta_i^b,\boldsymbol{\theta}_{\neq i}) \tag{6.2.2}$$

$$\forall_j \forall_{\mathbf{a}} \quad u(\langle \theta_i^c,\boldsymbol{\theta}_{\neq i}\rangle,\mathbf{a}) = \frac{\Pr(\theta_i^a,\boldsymbol{\theta}_{\neq i})u(\langle \theta_i^a,\boldsymbol{\theta}_{\neq i}\rangle,\mathbf{a}) + \Pr(\theta_i^b,\boldsymbol{\theta}_{\neq i})u(\langle \theta_i^b,\boldsymbol{\theta}_{\neq i}\rangle,\mathbf{a})}{\Pr(\theta_i^a,\boldsymbol{\theta}_{\neq i}) + \Pr(\theta_i^b,\boldsymbol{\theta}_{\neq i})}. \tag{6.2.3}$$

The result is a new BG B' in which the expected value is the same as in the original BG: $V^{B'} = V^B$.

Proof. We show that the expected value of any joint policy $(\beta_i,\boldsymbol{\beta}_{\neq i})$ that satisfies condition (6.2.1) is the same in both B and B'. Using short-hand $\mathbf{a} = \langle \beta_i(\theta_i),\boldsymbol{\beta}_{\neq i}(\boldsymbol{\theta}_{\neq i})\rangle$,

$$\begin{aligned}
V^B(\beta_i,\boldsymbol{\beta}_{\neq i}) =& \sum_{\boldsymbol{\theta}_{\neq i}} \Bigg[\overbrace{\Pr(\theta_i^a,\boldsymbol{\theta}_{\neq i})u(\langle \theta_i^a,\boldsymbol{\theta}_{\neq i}\rangle,\mathbf{a}) + \Pr(\theta_i^b,\boldsymbol{\theta}_{\neq i})u(\langle \theta_i^b,\boldsymbol{\theta}_{\neq i}\rangle,\mathbf{a})}^{\left(\Pr(\theta_i^a,\boldsymbol{\theta}_{\neq i})+\Pr(\theta_i^b,\boldsymbol{\theta}_{\neq i})\right)u(\langle \theta_i^c,\boldsymbol{\theta}_{\neq i}\rangle,\mathbf{a})} \\
&\quad + \sum_{\theta_i \in \Theta_i \setminus \{\theta_i^a,\theta_i^b\}} \Pr(\theta_i,\boldsymbol{\theta}_{\neq i})u(\langle \theta_i,\boldsymbol{\theta}_{\neq i}\rangle,\mathbf{a}) \Bigg] \\
=& \sum_{\boldsymbol{\theta}_{\neq i}} \left[\Pr(\theta_i^c,\boldsymbol{\theta}_{\neq i})u(\langle \theta_i^c,\boldsymbol{\theta}_{\neq i}\rangle,\mathbf{a}) + \sum_{\theta_i \in \Theta_i' \setminus \{\theta_i^c\}} \Pr(\theta_i,\boldsymbol{\theta}_{\neq i})u(\langle \theta_i,\boldsymbol{\theta}_{\neq i}\rangle,\mathbf{a}) \right] \\
=& V^{B'}(\beta_i,\boldsymbol{\beta}_{\neq i})
\end{aligned}$$

which is the expected value of $(\beta_i,\boldsymbol{\beta}_{\neq i})$ as computed in the reduced BG. □

This theorem tells us that given that agent i is committed to taking the same action for its types θ_i^a,θ_i^b, we can reduce the Bayesian game B to a smaller one B' and translate the joint BG-policy $\boldsymbol{\beta}'$ found for B' back to a joint BG-policy $\boldsymbol{\beta}$ in B. This does not necessarily mean that $\boldsymbol{\beta} = (\beta_i,\boldsymbol{\beta}_{\neq i})$ also is a solution (Bayesian Nash-equilibrium) for B, because the best-response of agent i against $\boldsymbol{\beta}_{\neq i}$ may not select the same action for θ_i^a,θ_i^b. Rather β_i is the best-response against $\boldsymbol{\beta}_{\neq i}$ *given that the same action needs to be taken for* θ_i^a,θ_i^b. For instance, when θ_i^a,θ_i^b are BRE as we detail below.

The preceding showed that reduction through clustering is possible *if* an agent is committed to select the same action for two of its types. In the following we will

identify *when* an agent is committed to select the same action for two of its types through the notion of *best-response equivalence.* I.e., the following demonstrates when the best-response for two types is the same. In a general BG, a best-response β_i^* for agent i's type θ_i against some fixed policy profile $\boldsymbol{\beta}_{\neq i}$ is given by

$$\beta_i^* = \arg\max_{\beta_i} \sum_{\theta_i} \Pr(\theta_i) \sum_{\boldsymbol{\theta}_{\neq i}} \Pr(\boldsymbol{\theta}_{\neq i}|\theta_i) u_i(\langle \theta_i, \boldsymbol{\theta}_{\neq i} \rangle, \langle \beta_i(\theta_i), \boldsymbol{\beta}_{\neq i}(\boldsymbol{\theta}_{\neq i}) \rangle),$$

or, alternatively, we can compose β_i^* as the best response $\beta_i^*(\theta_i)$ for each type θ_i:

$$\beta_i^*(\theta_i) = \arg\max_{a_i} \sum_{\boldsymbol{\theta}_{\neq i}} \Pr(\boldsymbol{\theta}_{\neq i}|\theta_i) u_i(\langle \theta_i, \boldsymbol{\theta}_{\neq i} \rangle, \langle a_i, \boldsymbol{\beta}_{\neq i}(\boldsymbol{\theta}_{\neq i}) \rangle).$$

It is this latter formulation we use in the following lemma, identifying best-response equivalence.

Lemma 6.1 (Best-response equivalence)**.** *When for two types $\theta_{i,a}$,$\theta_{i,b}$ it holds that*

$$\forall_{\theta_{\neq i}} \quad \Pr(\theta_{\neq i}|\theta_{i,a}) = \Pr(\theta_{\neq i}|\theta_{i,b}) \tag{6.2.4}$$

and

$$\forall_{\mathbf{a}} \forall_{\theta_{\neq i}} \quad u(\theta_{i,a}, \theta_{\neq i}, \mathbf{a}) = u(\theta_{i,b}, \theta_{\neq i}, \mathbf{a}), \tag{6.2.5}$$

then the best-response policy for agent i will always select the same action for $\theta_{i,a}$,$\theta_{i,b}$.

Proof. We can simply derive

$$\begin{aligned} \beta_i^*(\theta_{i,a}) &= \arg\max_{a_i} \sum_{\theta_{\neq i}} \Pr(\theta_{\neq i}|\theta_{i,a}) u(\theta_{i,a}, \theta_{\neq i}, a_i, \mathbf{a}_{\neq i}) \\ &= \arg\max_{a_i} \sum_{\theta_{\neq i}} \Pr(\theta_{\neq i}|\theta_{i,b}) u(\theta_{i,b}, \theta_{\neq i}, a_i, \mathbf{a}_{\neq i}) \end{aligned}$$

which is equal to $\beta_i^*(\theta_{i,b})$. □

Remark. This lemma states sufficient, but not necessary conditions for best response equivalence. This is easy to understand by considering a randomly generated BG with many types, but few actions per agent. Because the probabilities and utilities are randomly generated, conditions (6.2.4) and (6.2.5) typically will not hold. However, as there are many types and few actions, any policy (so also a best-response policy) will need to select the same action for many types.

By combination of Theorem 6.1 and Lemma 6.1, it is clear that individual types $\theta_{i,a}$,$\theta_{i,b}$ can be clustered if the conditions in the lemma are satisfied. Note that although these results are presented in the context of identical payoff BGs, it is trivial to generalize them to BGs with individual payoff functions. As such, these results may perhaps also be used in more general methods for solving BGs.

6.3 Lossless Clustering in Dec-POMDPs

This section makes the bridge to the Dec-POMDP context. In particular it shows when two histories in a Dec-POMDP are BRE and can therefore be clustered together in a BG representing a stage of a Dec-POMDP.

6.3.1 Probabilistic Equivalence Criterion

A particular stage t of a Dec-POMDP can be represented as a BG. For such a BG we can cluster two individual histories $\vec{\theta}_{i,a}$,$\vec{\theta}_{i,b}$ when they satisfy the *probabilistic equivalence criterion* as we define here.

Criterion 6.1 (Probabilistic Equivalence)**.** *Two AOHs $\vec{\theta}_{i,a}$,$\vec{\theta}_{i,b}$ for agent i are probabilistically equivalent (PE) when the following holds:*

$$\forall_{\vec{\boldsymbol{\theta}}_{\neq i}} \forall_s \quad \Pr(s,\vec{\boldsymbol{\theta}}_{\neq i}|\vec{\theta}_{i,a}) = \Pr(s,\vec{\boldsymbol{\theta}}_{\neq i}|\vec{\theta}_{i,b}). \tag{6.3.1}$$

Remark. Alternatively, the criterion can be rewritten to the following two:

$$\forall_{\vec{\boldsymbol{\theta}}_{\neq i}} \quad \Pr(\vec{\boldsymbol{\theta}}_{\neq i}|\vec{\theta}_{i,a}) = \Pr(\vec{\boldsymbol{\theta}}_{\neq i}|\vec{\theta}_{i,b}), \tag{6.3.2}$$

$$\forall_{\vec{\boldsymbol{\theta}}_{\neq i}} \forall_s \quad \Pr(s|\vec{\boldsymbol{\theta}}_{\neq i},\vec{\theta}_{i,a}) = \Pr(s|\vec{\boldsymbol{\theta}}_{\neq i},\vec{\theta}_{i,b}). \tag{6.3.3}$$

These equations give a natural interpretation: the first says that the probability distribution over the other agents' AOHs must be identical for both $\vec{\theta}_{i,a}$,$\vec{\theta}_{i,b}$. The second demands that the resulting joint beliefs are identical.

Remark. The above probabilities are not well defined without the initial state distribution $\boldsymbol{b}^0$ and past joint policy $\boldsymbol{\varphi}^t$. However, since we consider clustering of histories within a particular BG (for some stage t) and because this BG is constructed for a particular $\boldsymbol{b}^0$,$\boldsymbol{\varphi}^t$, they are implicitly specified. Therefore we drop these arguments, clarifying the notation.

Remark. Probabilities as defined in (6.3.1) appear somewhat similar to beliefs in POMDPs, but are substantially different. In a Dec-POMDP it is not possible for an agent to maintain beliefs as in POMDPs. The probabilities here are not sufficient statistics. Only a 'multiagent belief' specified over states and *future policies* of other agents has been shown to be a sufficient statistic (Hansen et al., 2004). Our notion of PE is specified over states and AOHs given only a *past* joint policy. Thus establishing conditions for equivalence in Dec-POMDPs is a non-trivial extension over the POMDP case.

Probabilistic equivalence has a convenient property: if it holds for a particular pair of histories, then it will also hold for all *identical extensions* of those histories, i.e., the property propagates forwards regardless of the policies the other agents use.

Definition 6.1 (Identical extensions)**.** Given two AOHs $\vec{\theta}^t_{i,a}$,$\vec{\theta}^t_{i,b}$, their respective extensions $\vec{\theta}^{t+1}_{i,a} = (\vec{\theta}^t_{i,a},a_i,o_i)$ and $\vec{\theta}^{t+1}_{i,b} = (\vec{\theta}^t_{i,b},a'_i,o'_i)$ are called *identical extensions* if and only if $a_i = a'_i$ and $o_i = o'_i$.

Lemma 6.2 (Propagation of PE)**.** *Given $\vec{\theta}^t_{i,a}$,$\vec{\theta}^t_{i,b}$ that are PE, regardless of $\boldsymbol{\beta}^t_{\neq i}$ the policy the other agents use, identical extensions are also PE:*

$$\forall_{a_i^t}\forall_{o_i^{t+1}}\forall_{\boldsymbol{\beta}^t_{\neq i}}\forall_{s^{t+1}}\forall_{\vec{\boldsymbol{\theta}}^{t+1}_{\neq i}} \quad \Pr(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i}|\vec{\theta}^t_{i,a},a^t_i,o^{t+1}_i,\boldsymbol{\beta}^t_{\neq i}) = \Pr(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i}|\vec{\theta}^t_{i,b},a^t_i,o^{t+1}_i,\boldsymbol{\beta}^t_{\neq i}) \quad (6.3.4)$$

Proof. Assume an arbitrary a^t_i,o^{t+1}_i, $\boldsymbol{\beta}^t_{\neq i}$,$s^{t+1}$ and $\vec{\boldsymbol{\theta}}^{t+1}_{\neq i} = (\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}^t_{\neq i},\mathbf{o}^{t+1}_{\neq i}))$. We have that

$$\begin{aligned}
&\Pr(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i},o^{t+1}_i|\vec{\theta}^t_{i,a},a^t_i,\boldsymbol{\beta}^t_{\neq i})\\
&= \sum_{s^t} \Pr(o^{t+1}_i,\mathbf{o}^{t+1}_{\neq i}|a^t_i,\mathbf{a}^t_{\neq i},s^{t+1}) \Pr(s^{t+1}|s^t,a^t_i,\mathbf{a}^t_{\neq i}) \Pr(\mathbf{a}^t_{\neq i}|\vec{\boldsymbol{\theta}}^t_{\neq i},\boldsymbol{\beta}^t_{\neq i}) \Pr(s^t,\vec{\boldsymbol{\theta}}^t_{\neq i}|\vec{\theta}^t_{i,a})\\
&= \sum_{s^t} \Pr(o^{t+1}_i,\mathbf{o}^{t+1}_{\neq i}|a^t_i,\mathbf{a}^t_{\neq i},s^{t+1}) \Pr(s^{t+1}|s^t,a^t_i,\mathbf{a}^t_{\neq i}) \Pr(\mathbf{a}^t_{\neq i}|\vec{\boldsymbol{\theta}}^t_{\neq i},\boldsymbol{\beta}^t_{\neq i}) \Pr(s^t,\vec{\boldsymbol{\theta}}^t_{\neq i}|\vec{\theta}^t_{i,b})\\
&= \Pr(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i},o^{t+1}_i|\vec{\theta}^t_{i,b},a^t_i,\boldsymbol{\beta}^t_{\neq i})
\end{aligned}$$

Because we assumed an arbitrary s^{t+1},$\vec{\boldsymbol{\theta}}^{t+1}_{\neq i}$,$o^{t+1}_i$, we have that

$$\forall_{s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i},o^{t+1}_i} \quad \Pr(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i},o^{t+1}_i|\vec{\theta}^t_{i,a},a^t_i,\boldsymbol{\beta}^t_{\neq i}) = \Pr(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i},o^{t+1}_i|\vec{\theta}^t_{i,b},a^t_i,\boldsymbol{\beta}^t_{\neq i}) \quad (6.3.5)$$

In general we have that

$$\begin{aligned}
\Pr(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i}|\vec{\theta}^t_i,a^t_i,o^{t+1}_i,\boldsymbol{\beta}^t_{\neq i}) &= \frac{\Pr(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i},o^{t+1}_i|\vec{\theta}^t_i,a^t_i,\boldsymbol{\beta}^t_{\neq i})}{\Pr(o^{t+1}_i|\vec{\theta}^t_i,a^t_i,\boldsymbol{\beta}^t_{\neq i})}\\
&= \frac{\Pr(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i},o^{t+1}_i|\vec{\theta}^t_i,a^t_i,\boldsymbol{\beta}^t_{\neq i})}{\sum_{s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i}} \Pr(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i},o^{t+1}_i|\vec{\theta}^t_i,a^t_i,\boldsymbol{\beta}^t_{\neq i})}
\end{aligned}$$

Now, because of (6.3.5), both the nominator and denominator are the same when substituting $\vec{\theta}^t_{i,a}$,$\vec{\theta}^t_{i,b}$ in this equation. This means we can conclude

$$\Pr(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i}|\vec{\theta}^t_{i,a},a^t_i,o^{t+1}_i,\boldsymbol{\beta}^t_{\neq i}) = \Pr(s^{t+1},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i}|\vec{\theta}^t_{i,b},a^t_i,o^{t+1}_i,\boldsymbol{\beta}^t_{\neq i})$$

Finally, because a^t_i,o^{t+1}_i,$\boldsymbol{\beta}^t_{\neq i}$,$s^{t+1}$,$\vec{\boldsymbol{\theta}}^{t+1}_{\neq i}$ were all arbitrarily chosen we can conclude (6.3.4). □

6.3.2 Identical Values Allow Lossless Clustering of Histories

Since we want to show that two PE histories can be clustered under the optimal policy, we need to show (6.2.5) holds and thus that their optimal Q-values are the same.

Lemma 6.3 (Q^* equivalence)**.** *When two histories in a BG for a Dec-POMDP $\vec{\theta}_{i,a}$, $\vec{\theta}_{i,b}$ satisfy Criterion 6.1, then they have equal Q-values according any joint policy $\boldsymbol{\pi}$*

$$\forall_{\vec{\boldsymbol{\theta}}^t_{\neq i}} \forall_{\mathbf{a}} \quad Q_{\boldsymbol{\pi}}(\vec{\theta}^t_{i,a},\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}) = Q_{\boldsymbol{\pi}}(\vec{\theta}^t_{i,b},\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}). \tag{6.3.6}$$

Proof. The proof is by induction backwards in time (i.e., from the last time step $t = h-1$ to the first $t = 0$). However, to prove the induction step we employ Lemma 6.2, which ensures propagation forward through time of the PE criterion on identical extensions.

The base case is given by the last stage $t = h-1$ of the Dec-POMDP. In this case we have that

$$\forall_{\mathbf{a}} \forall_{\vec{\boldsymbol{\theta}}^t_{\neq i}} \quad Q_{\boldsymbol{\pi}}(\vec{\theta}_{i,a},\vec{\boldsymbol{\theta}}_{\neq i},\mathbf{a}) = \sum_{s\in\mathcal{S}} R(s,\mathbf{a}) \Pr(s|\vec{\theta}_{i,a},\vec{\boldsymbol{\theta}}_{\neq i}) = \sum_{s\in\mathcal{S}} R(s,\mathbf{a}) \Pr(s|\vec{\theta}_{i,b},\vec{\boldsymbol{\theta}}_{\neq i}) = Q_{\boldsymbol{\pi}}(\vec{\theta}_{i,b},\vec{\boldsymbol{\theta}}_{\neq i},\mathbf{a})$$

because of (6.3.3) in Criterion 6.1. For stages $0 \leq t < h-1$ the $Q_{\boldsymbol{\pi}}$ is given by

$$Q_{\boldsymbol{\pi}}(\vec{\boldsymbol{\theta}}^t,\mathbf{a}) = R(\vec{\boldsymbol{\theta}}^t,\mathbf{a}) + \sum_{\mathbf{o}^{t+1}\in\mathcal{O}} \Pr(\mathbf{o}^{t+1}|\vec{\boldsymbol{\theta}}^t,\mathbf{a}) Q_{\boldsymbol{\pi}}(\vec{\boldsymbol{\theta}}^{t+1},\boldsymbol{\pi}(\vec{\boldsymbol{\theta}}^{t+1})).$$

The induction hypothesis is as follows: If at $t+1$ the criteria hold for any two $\vec{\theta}^{t+1}_{i,a},\vec{\theta}^{t+1}_{i,b}$, then they have equal Q-values:

$$\forall_{\vec{\boldsymbol{\theta}}^{t+1}_{\neq i}} \forall_{\mathbf{a}^{t+1}} \quad Q_{\boldsymbol{\pi}}(\vec{\theta}^{t+1}_{i,a},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i},\mathbf{a}^{t+1}) = Q_{\boldsymbol{\pi}}(\vec{\theta}^{t+1}_{i,b},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i},\mathbf{a}^{t+1}). \tag{6.3.7}$$

Assume: some stage $0 \leq t < h-1$, that the criteria hold for $\vec{\theta}^t_{i,a},\vec{\theta}^t_{i,b}$ and an arbitrary $\mathbf{a} = \langle a_i,\mathbf{a}_{\neq i}\rangle$ and $\vec{\boldsymbol{\theta}}^t_{\neq i}$. Now we need to show that

$$Q_{\boldsymbol{\pi}}(\vec{\theta}^t_{i,a},\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}) = Q_{\boldsymbol{\pi}}(\vec{\theta}^t_{i,b},\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}) \tag{6.3.8}$$

I.e.:

$$R(\vec{\theta}^t_{i,a},\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}) + \sum_{\mathbf{o}^{t+1}\in\mathcal{O}} \Pr(\mathbf{o}^{t+1}|\vec{\theta}^t_{i,a},\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}) Q_{\boldsymbol{\pi}}(\vec{\boldsymbol{\theta}}^{t+1}_a,\boldsymbol{\pi}(\vec{\boldsymbol{\theta}}^{t+1}_a)) = R(\vec{\theta}^t_{i,b},\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}) + \sum_{\mathbf{o}^{t+1}\in\mathcal{O}} \Pr(\mathbf{o}^{t+1}|\vec{\theta}^t_{i,b},\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}) Q_{\boldsymbol{\pi}}(\vec{\boldsymbol{\theta}}^{t+1}_b,\boldsymbol{\pi}(\vec{\boldsymbol{\theta}}^{t+1}_b)) \tag{6.3.9}$$

where

$$\vec{\boldsymbol{\theta}}^{t+1}_a = (\langle\vec{\theta}^t_{i,a},\vec{\boldsymbol{\theta}}^t_{\neq i}\rangle,\mathbf{a},\mathbf{o}^{t+1}) = \langle(\vec{\theta}^t_{i,a},a_i,o^{t+1}_i),(\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}_{\neq i},\mathbf{o}^{t+1}_{\neq i})\rangle = \langle\vec{\theta}^{t+1}_{i,a},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i}\rangle$$

$$\vec{\boldsymbol{\theta}}^{t+1}_b = (\langle\vec{\theta}^t_{i,b},\vec{\boldsymbol{\theta}}^t_{\neq i}\rangle,\mathbf{a},\mathbf{o}^{t+1}) = \langle(\vec{\theta}^t_{i,b},a_i,o^{t+1}_i),(\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}_{\neq i},\mathbf{o}^{t+1}_{\neq i})\rangle = \langle\vec{\theta}^{t+1}_{i,b},\vec{\boldsymbol{\theta}}^{t+1}_{\neq i}\rangle$$

To prove the equality of (6.3.9), we have to show that:

1. The immediate rewards are equal: $R(\vec{\theta}^t_{i,a},\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}) = R(\vec{\theta}^t_{i,b},\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a})$. This clearly is the case (similar to the proof of the last stage).

2. $\forall_{\mathbf{o}^{t+1}} \quad \Pr(\mathbf{o}^{t+1}|\vec{\theta}^t_{i,a},\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a}) = \Pr(\mathbf{o}^{t+1}|\vec{\theta}^t_{i,b},\vec{\boldsymbol{\theta}}^t_{\neq i},\mathbf{a})$. Equal observation probabilities. This is also evident given that the criterion holds: if the underlying state distribution is the same, the next joint observation probabilities are also identical.

3. The relevant next-stage Q-values are identical. I.e.:

$$\forall_{\mathbf{o}^{t+1}}\forall_{\mathbf{a}^{t+1}} \quad Q_{\boldsymbol{\pi}}(\vec{\boldsymbol{\theta}}^{t+1}_a,\mathbf{a}^{t+1}) = Q_{\boldsymbol{\pi}}(\vec{\boldsymbol{\theta}}^{t+1}_b,\mathbf{a}^{t+1}). \tag{6.3.10}$$

   To prove this, we show that the induction hypothesis applies: We can rewrite the demonstrandum (6.3.10) to

$$\forall_{o_i^{t+1}}\forall_{\mathbf{o}^{t+1}_{\neq i}}\forall_{\mathbf{a}^{t+1}} \quad Q_{\boldsymbol{\pi}}(\vec{\theta}^{t+1}_{i,a} = (\vec{\theta}^t_{i,a},a_i,o_i^{t+1}),\vec{\boldsymbol{\theta}}^{t+1}_{\neq i},\mathbf{a}^{t+1}) = Q_{\boldsymbol{\pi}}(\vec{\theta}^{t+1}_{i,b} = (\vec{\theta}^t_{i,b},a_i,o_i^{t+1}),\vec{\boldsymbol{\theta}}^{t+1}_{\neq i},\mathbf{a}^{t+1}).$$

   This is proven (by application of the induction hypothesis) if we can show that the criterion holds for $\vec{\theta}^{t+1}_{i,a},\vec{\theta}^{t+1}_{i,b}$. Since $\vec{\theta}^{t+1}_{i,a},\vec{\theta}^{t+1}_{i,b}$ are identical extensions of PE histories $\vec{\theta}^t_{i,a},\vec{\theta}^t_{i,b}$, they themselves are PE by application of Lemma 6.2. Therefore the induction hypothesis applies which means that (6.3.10) holds. □

Theorem 6.2 (Lossless clustering)**.** *When two histories $\vec{\theta}_{i,a},\vec{\theta}_{i,b}$ are PE, then they are best-response equivalent and can be clustered as one history without loss in value.*

Proof. We first prove that PE implies BRE. The criterion itself entails (6.2.4). Lemma 6.3 asserts that, for a BG constructed using an optimal Q-value function $Q_{\boldsymbol{\pi}^*}$, (6.2.5) holds. Now, given that PE implies BRE, we can apply Theorem 6.1 to prove that $\vec{\theta}_{i,a},\vec{\theta}_{i,b}$ can be clustered without loss in value. □

Remark. Again, the criterion gives a sufficient, but not necessary condition. In particular given a policy of the other agents, many types are BRE and can be clustered. However, as far as we know, only if the criterion holds we can *guarantee* that two histories have the same best-response against *any* policy of the other agents.

6.4 GMAA*-Cluster

Knowledge of which individual histories can be clustered together without loss of value may potentially be employed by many algorithms. In this paper, we focus on its application within the GMAA* framework.

Emery-Montemerlo et al. (2005) showed how clustering can be incorporated at every stage in their algorithm: when the BG for a stage t is constructed, first a

Algorithm 6.1 $\Phi_{\texttt{Expand}}$ = ConstructAndSolveBG-Cluster($\boldsymbol{\varphi}^t$,$\boldsymbol{b}^0$)

1: **if** BootstrappedClustering **then**
2: $BG^t \leftarrow$ `ConstructExtendedBG`(BG^{t-1},$\boldsymbol{\varphi}^t$)
3: **else**
4: $BG^t \leftarrow$ `ConstructBG`($\boldsymbol{\varphi}^t$,$\boldsymbol{b}^0$)
5: **end if**
6: $BG^t \leftarrow$ ClusterBG(BG^t)
7: **for all** joint BG-policies $\boldsymbol{\beta}^t$ **do**
8: $\boldsymbol{\varphi}^{t+1} \leftarrow$($\boldsymbol{\varphi}^t$,$\boldsymbol{\beta}^t$)
9: $\widehat{V}(\boldsymbol{\varphi}^{t+1}) \leftarrow V^{0\ldots t-1}(\boldsymbol{\varphi}^t) + \widehat{V}(\boldsymbol{\beta}^t)$
10: $\Phi_{\texttt{Expand}} \leftarrow \Phi_{\texttt{Expand}} \cup \boldsymbol{\varphi}^{t+1}$
11: **end for**

clustering of the individual histories (types) is performed and only afterward the (reduced) BG is solved. The same thing can be done within GMAA*, leading to an algorithm we dub GMAA*-Cluster. In particular, GMAA*-Cluster replaces the function `ConstructAndSolveBG` from Algorithm 4.1 with Algorithm 6.1.

The actual clustering is performed by Algorithm 6.2, which performs a pairwise comparison of all types of each agent to see if they satisfy the criterion. This means that

$$O(|\Theta_i|^2)$$

are performed for each agent i. If there is a large number of states some, efficiency may be gained by first checking (6.3.2) and then checking (6.3.3), rather than looping over all $\langle s,\boldsymbol{\theta}_{\neq i}\rangle$ as is done in line 5.

Also note that the algorithm shown assumes that the heuristic used as the payoff function u is admissible (i.e., is an upper bound to the optimal value). Therefore, rather than using (6.2.3), we can take the lowest upper bound in line 15.[1] In general this might increase the tightness of the heuristic, which can have a great effect on the performance as demonstrated in Chapter 4.

6.4.1 Bootstrapped Clustering

Because PE of AOHs propagates forwards (i.e., identical extensions of PE histories are also PE), we do not have to construct all $|\mathcal{O}_i|^t$ possible AOHs at every stage t (given the past policy φ_i^t of agent i). Instead of clustering this exponentially growing set of types, we can simply extend the already clustered types of the previous stage's BG, as shown in Algorithm 6.3.

That is, given Θ_i, the set of types of agent i at the previous stage $t-1$, and β_i^{t-1} the policy agent i took at that stage, the set of types at stage t, Θ_i', can be constructed as

$$\Theta_i' = \left\{\theta_i' = (\theta_i,\beta_i^{t-1}(\theta_i),o_i^t) \mid \theta_i \in \Theta_i, o_i^t \in \mathcal{O}_i\right\}. \tag{6.4.1}$$

[1] For the heuristics we employed there is no difference, because their heuristic value is also guaranteed to be the same if the criterion holds.

Algorithm 6.2 $BG = \text{ClusterBG}(BG)$

for each agent i **do**
  for each individual type $\theta_i \in BG.\Theta_i$ **do**
    for each individual type $\theta'_i \in BG.\Theta_i$ **do**
      isEquivalent $\leftarrow true$
      for all $\langle s,\boldsymbol{\theta}_{\neq i}\rangle$ **do**
        if $\Pr(s,\boldsymbol{\theta}_{\neq i}|\theta_i) \neq \Pr(s,\boldsymbol{\theta}_{\neq i}|\theta'_i)$ **then**
          isEquivalent $\leftarrow false$
          break
        end if
      end for
      if isEquivalent **then**
        $BG.\Theta_i \leftarrow BG.\Theta_i \backslash \theta'_i$ {Remove θ'_i from BG:}
        for each $\mathbf{a} \in \mathcal{A}$ **do**
          for all $\boldsymbol{\theta}_{\neq i}$ **do**
            { take the lowest upper bound }
            $u(\theta_i,\boldsymbol{\theta}_{\neq i},\mathbf{a}) \leftarrow \min(u(\theta_i,\boldsymbol{\theta}_{\neq i},\mathbf{a}),u(\theta'_i,\boldsymbol{\theta}_{\neq i},\mathbf{a}))$
            $\Pr(\theta_i,\boldsymbol{\theta}_{\neq i}) \leftarrow \Pr(\theta_i,\boldsymbol{\theta}_{\neq i}) + \Pr(\theta'_i,\boldsymbol{\theta}_{\neq i})$
            $\Pr(\theta'_i,\boldsymbol{\theta}_{\neq i}) \leftarrow 0$
          end for
        end for
      end if
    end for
  end for
end for

This means that the size of this newly constructed set is

$$|\Theta'_i| = |\Theta_i| \cdot |\mathcal{O}_i| \tag{6.4.2}$$

If the typeset Θ_i at the previous stage $t-1$ was much smaller than the set of all histories $|\Theta_i| \ll |\mathcal{O}_i|^{t-1}$, then the new typeset Θ'_i is also much smaller: $|\Theta'_i| \ll |\mathcal{O}_i|^t$. This way, we bootstrap the clustering at each stage and spend significantly less time clustering.

The above is possible only because we perform an exact, value preserving, clustering for which Lemma 6.2 tells us that identical extensions will also be clustered without loss in value. When performing the same procedure in a lossy clustering scheme (e.g., as in Emery-Montemerlo et al. 2005) errors might accumulate and thus it might be better to re-cluster from scratch at every stage. Still, this will mean that a resulting algorithm only has limited scalability. Since lossy clustering is beyond the scope of this chapter, only bootstrapped clustering is considered.

6.4.2 Complexity

Optimally solving a BG takes exponential time w.r.t. the number of types, as there are $O(|\mathcal{A}_*|^{n|\Theta_*|})$ joint BG-policies. Clustering involves a pairwise comparison of all

Algorithm 6.3 BG^t = ConstructExtendedBG(BG^{t-1},$\boldsymbol{\beta}^{t-1}$)

```
t ← BG^{t-1}.t + 1
pBG ← BG^{t-1}
pPol ← β^{t-1}
for each agent i do
    BG^t.Θ_i = ConstructExtendedTypeSet(i)
end for
for each joint type θ = (θ^{t-1},a^{t-1},o^t) ∈ BG^t.Θ do
    for each state s^t ∈ S do
        Compute Pr(s^t|θ)
    end for
    Pr(θ) ← Pr(o^t|θ^{t-1},a^{t-1}) Pr(θ^{t-1})
    for each a ∈ A do
        q ← ∞
        for each history θ⃗^t represented by θ do
            q ← min(q,Q̂(θ⃗^t,a)) { if Q* ≤ Q̂ we can take the lowest upper bound }
        end for
        u(θ,a) ← q
    end for
end for
```

types of each agent and each of these comparisons needs to check $O(|\Theta_*|^{n-1}\,|\mathcal{S}|)$ numbers for equality to verify (6.3.1). The total cost of clustering can therefore be written as

$$O(n\,|\Theta_*|^2\,|\Theta_*|^{n-1}\,|\mathcal{S}|),$$

which is only polynomial in the number of types. When clustering decreases the number of types $|\Theta_*|$, it can therefore significantly reduce the overall time needed. However, when no clustering is possible, some overhead will be incurred.

From an implementation perspective, it is important to avoid reconstructing flat Dec-POMDP policies, as they can cause an exponential blow up in space requirements. Instead, we maintain a pointer to the previous joint policy $\boldsymbol{\varphi}^t = (\boldsymbol{\varphi}^{t-1},\boldsymbol{\beta}^{t-1})$. A downside is that either extensive pointer administration is necessary to determine whether some previous policy $\boldsymbol{\varphi}^{t-1}$ is still needed (i.e., still being pointed to by some $\boldsymbol{\varphi}^t$ in the policy pool), or all constructed $\boldsymbol{\varphi}^{t-1}$ must be kept in memory. For convenience, our current implementation uses the latter solution.

6.5 Experiments

This section first presents a comparison of the optimal solution of several problems with and without clustering, followed by an analysis of the generality of lossless clustering, also for larger horizons for which optimal solutions are infeasible to compute.

GMAA*-Cluster is evaluated on a range of benchmark problems. All timing results mentioned in this chapter are CPU times with a resolution of 0.01s, and

Dec-Tiger (Q_{BG})					
h	V^*	T_{GMAA^*}(s)	$T_{cluster}$(s)	$\lvert BG^t\rvert$	$\lvert cBG^t\rvert$
2	−4.0000	≤ 0.01	≤ 0.01	4	4.00
3	5.1908	0.02	≤ 0.01	16	9.00
4	4.8028	3,069.4	1.50	64	23.14
5	7.0265	–	130.82	256	40.43
BroadcastChannel (Q_{MDP})					
h	V^*	T_{GMAA^*}(s)	$T_{cluster}$(s)	$\lvert BG^t\rvert$	$\lvert cBG^t\rvert$
2	2.0000	≤ 0.01	≤ 0.01	4	1.00
3	2.9900	≤ 0.01	≤ 0.01	16	1.00
4	3.8900	3.22	≤ 0.01	64	1.00
5	4.7900	–	≤ 0.01	256	1.00
6	5.6900	–	≤ 0.01	1024	1.00
7	6.5900	–	≤ 0.01	4096	1.00
8	7.4900	–	≤ 0.01	16384	1.00
9	8.3900	–	≤ 0.01	65536	1.00
10	9.2900	–	≤ 0.01	$2.62e5$	1.00
15	13.7900	–	≤ 0.01	$2.68e8$	1.00
20	18.3132	–	0.08	$2.75e11$	1.00
25	22.8815	–	1.67	$2.81e14$	1.00
GridSmall (Q_{BG})					
h	V^*	T_{GMAA^*}(s)	$T_{cluster}$(s)	$\lvert BG^t\rvert$	$\lvert cBG^t\rvert$
2	0.9100	≤ 0.01	≤ 0.01	4	4.00
3	1.5504	4.21	0.71	16	12.00
4	2.2416	–	30.17	64	25.00
Cooperative Box Pushing (Q_{MDP})					
h	V^*	T_{GMAA^*}(s)	$T_{cluster}$(s)	$\lvert BG^t\rvert$	$\lvert cBG^t\rvert$
2	17.6000	0.05	≤ 0.01	25	4.00
3	66.0810	–	4.55	625	25.00

Table 6.1: Results of GMAA* on several problems. Listed are the run times of regular GMAA* and GMAA*-Cluster, and the size of the BGs solved at each time step, with and without clustering.

were obtained on 3.4GHz Intel Xeon processors with 2GB memory. The timings exclude time needed to parse the problem and compute the heuristic (which can be amortized).

6.5.1 Optimal Solutions using Clustering

For all considered problems we compared GMAA* against GMAA*-Cluster using the Q_{BG} or Q_{MDP} heuristic Oliehoek et al. (2008b), depending on problem size and planning horizon. Regardless of the particular heuristic, both methods compute an optimal policy, but we expect GMAA*-Cluster to be more efficient when lossless clustering is possible in the domain. The obtained results are shown in Table 6.1 and Table 6.2, which detail the optimal value V^* and the running time T_{GMAA^*} for

Recycling Robots (Q_{MDP})					
h	V^*	T_{GMAA^*}(s)	$T_{cluster}$(s)	$\|BG^t\|$	$\|cBG^t\|$
2	6.8000	≤ 0.01	≤ 0.01	4	4.00
3	9.7647	0.02	≤ 0.01	16	9.00
4	11.7264	23052.5	0.02	64	8.67
5	13.7643	–	0.10	256	9.00
6	15.5760	–	0.19	1024	9.00
7	17.2126	–	0.67	4096	9.00
8	18.6839	–	1.28	16384	9.00
9	20.0085	–	2.72	65536	9.00
10	21.2006	–	4.92	2.62e5	9.00
11	22.2734	–	9.83	1.05e6	9.00
12	23.2390	–	17.11	4.19e6	9.00
13	24.1080	–	30.61	1.68e7	9.00
14	24.8901	–	50.12	6.71e7	9.00
15	25.5940	–	81.46	2.68e8	9.00
Hotel 1 (Q_{BG})					
h	V^*	T_{GMAA^*}(s)	$T_{cluster}$(s)	$\|BG^t\|$	$\|cBG^t\|$
2	9.5000	≤ 0.01	0.02	16	4.00
3	15.7047	–	0.07	256	16.00
4	20.1125	–	1.37	4096	32.00
FireFighting $\langle n_h = 3, n_f = 3\rangle$ (Q_{BG})					
h	V^*	T_{GMAA^*}(s)	$T_{cluster}$(s)	$\|BG^t\|$	$\|cBG^t\|$
2	−4.3825	0.03	0.03	4	4.00
3	−5.7370	0.91	0.70	16	16.00
4	−6.5789	5605.3	5823.5	64	64.00

Table 6.2: Results of GMAA* on several problems. Listed are the run times of regular GMAA* and GMAA*-Cluster, and the size of the BGs solved at each time step, with and without clustering.

GMAA* and $T_{cluster}$ for GMAA*-Cluster. Entries marked '–' indicate that no solution was found within 8 hours. Furthermore, the tables list the number of *joint* types in the BGs constructed for the last stage without clustering, $|BG^t|$, and with, $|cBG^t|$. The former is constant while the latter is an average, as the algorithm can form BGs for different past policies, leading to clusterings of different sizes. For the Dec-Tiger problem, the solution time needed by GMAA*-Cluster is more than 3 orders of magnitude less for horizon $h = 4$. For $h = 5$ this test problem has 3.82e29 joint policies. To our knowledge, no other method has been able to optimally solve $h = 5$ Dec-Tiger. GMAA*-Cluster, however, is able to solve Dec-Tiger for $h = 5$ in reasonable time.

For the FireFighting problem, no lossless clustering is possible at any stage, and as such, we incur some overhead for the clustering. This is clearly shown for $h = 4$. For horizon 3, GMAA*-Cluster is actually a bit faster. Analysis revealed that for this horizon the cost of attempting to cluster is negligible. GMAA*-Cluster is faster because constructing the BGs using bootstrapping from the previous BG

takes less time than constructing a BG from scratch.

For GridSmall, Cooperative Box Pushing, and Hotel 1 the results are comparable to those for Dec-Tiger: substantial clustering is possible, resulting in significant speedups. Because the solution of BGs takes time exponential in their size, even small reductions in size would yield a big increase in efficiency. The substantial amounts of clustering found in these problems, therefore allow optimal solutions for longer horizons than have have been presented before.

For BroadcastChannel, GMAA*-Cluster achieves an even more dramatic increase in performance, allowing the solution of up to horizon $h = 25$. Analysis reveals that the BGs constructed for all stages are fully clustered: they contain only one type for each agent. Consequently, the time needed to solve each BG does not grow with the horizon. The solution time, however, still increases super-linear because of the increased amount of backtracking and memory management. The Recycling Robots problem can also be clustered to a relatively constant number of approximately 9 joint types per stage, allowing for optimal solving to high horizons. Both the BroadcastChannel and Recycling Robots problem run out of (2GB of) memory for higher horizons.

The fact that the BroadcastChannel problem exhibits full clustering can be explained as follows. When constructing a BG for $t = 1$, there is only one joint type for the previous BG, so given $\boldsymbol{\beta}^0$, the solution for the previous BG, there is no uncertainty with respect to the previous joint action $\mathbf{a}^0$. The crucial property of BroadcastChannel is that the (joint) observation tells us nothing about the new state, but only about what joint action was taken (e.g., 'collision' if both agent chose to 'send'). As a result, the observation does not convey any information and the different individual histories can be clustered. In a BG constructed for stage $t = 2$, there will again be only one joint type in the previous game. Therefore, given the past policy, the actions of the other agents can be perfectly predicted. Again the observation will convey no information so this process repeats. Consequently, the problem can be considered a special form of a non-observable Dec-POMDP; lossless clustering automatically exploits this property.

Another special class of problems that exhibits full clustering are those with a known start state and deterministic actions. Again in this case, the observations convey no information (because we can perfectly predict everything), and all histories can be clustered. This can be described as a special case of a fully observable problem which clustering automatically exploits.

The FireFighting problem does not allow any clustering. This can be understood as follows: given a (heuristically) good past joint policy, each agent typically visits different houses (cf. Figure 2.4 on page 28). As such, each different observation history will typically induce a different belief over the global state.

The other problems are harder to analyze. In Dec-Tiger a key property is that opening the door resets the problem. Such resets invalidate the history, allowing for clustering. Another factor is that the observations are taken independently given the new state only. I.e., $\Pr(\mathbf{o}|\mathbf{a},s') = \Pr(o_1|s')\Pr(o_2|s')$, which means that all information regarding the history of the other agent is obtained through estimation of the state.

6.5.2 General Clustering Performance

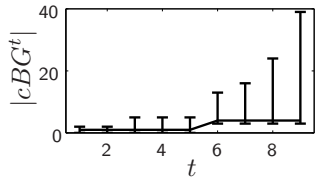
The reduction in BG-size in GMAA*-Cluster leads to significant gains in efficiency, showing that heuristically high-ranked partial policies lead to BGs that allow for much clustering. To test the general applicability of the clustering method, we investigated how much clustering can be done in BGs constructed for random past policies. If substantial clustering is possible on random policies, not just those encountered by GMAA*-Cluster, then the approach may be useful for a much broader set of methods. The results are shown in Figure 6.1, which shows the median number of joint types $|cBG^t|$ in the Bayesian games (constructed for 1,000 random past policies) for different stages after clustering.

The FireFighting problem, which could not be clustered when searching for an optimal policy, does allow for some clustering given randomly selected policies (Figure 6.1g). In both the Recycling Robots and the Hotel 1 problem the growth in BG size appears to stabilize, while in Dec-Tiger, GridSmall, and Cooperative Box Pushing $|cBG^t|$ keeps growing in the planning horizon. Even so, $|BG^t|$ grows faster resulting in high clustering ratios also for these problems.

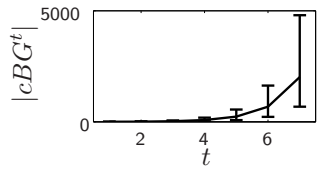
These experiments imply that the proposed clustering technique can provide significantly smaller policy representations without loss of value at a relatively low computational cost, for the benefit of optimal and approximate algorithms alike. Also this technique gives insight into how many future policies an agent should consider: if at some stage and given a past policy an agent has only k types, this means that it maximally needs to consider k future policies from that situation. The memory bounded dynamic programming (MBDP) algorithm and its variants (e.g., Seuken and Zilberstein, 2007b; Carlin and Zilberstein, 2008)), discussed in Section 2.6.5.1, have a parameter controlling the number of future policies considered, but until now there has been no principled way of estimating good values for this parameter. As such we expect that this clustering technique can have a big impact on new and existing, exact and approximate algorithms.

6.6 Conclusions

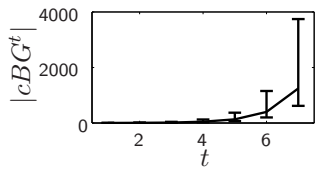
This chapter introduced a method for lossless clustering of action-observation histories in Dec-POMDPs, which can be applied in GMAA* policy search for Dec-POMDPs via Bayesian games. Rather than applying an ad-hoc clustering of these BGs, a probabilistic equivalence criterion was identified that guarantees that, given a particular past joint policy $\boldsymbol{\varphi}^t$, two action-observation histories $\vec{\theta}_i^t$ of agent i at stage t have the same optimal Q-values and therefore can be clustered without loss in solution quality. Empirical evaluation of GMAA* demonstrated that for several domains speedups of multiple orders of magnitude are achieved by clustering. We also investigated the amount of clustering possible for random past policies $\boldsymbol{\varphi}^t$, the result of which suggests that our clustering methods may also be exploited in other algorithms, such as IMBDP (Seuken and Zilberstein, 2007b).



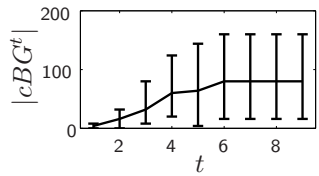
(a) Dec-Tiger



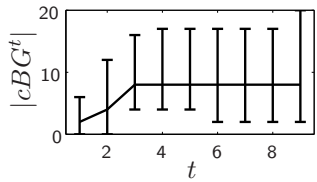
(b) GridSmall



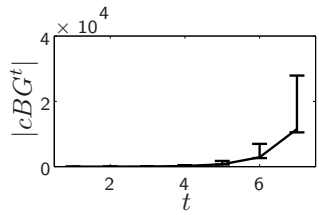
(c) C. Box Pushing



(d) Hotel 1



(e) Recycling Robots



(f) FireFighting

D-T (a)	GS (b)	CBP (c)	H1 (d)	RR (e)	FF (f)
65,536	8.127	4.90e6	8.59e8	32,768	1.42

(g) Median clustering ratios $\frac{|BG^t|}{|cBG^t|}$ for last time step.

Figure 6.1: Empirical clustering performance given random joint policies, for several problems, based on 1,000 independent samples. Plots (a)–(f) show the median size of the Bayesian games at each stage after clustering $|cBG^t|$, and the errorbars show the 0.25 and 0.75-quantile. Table (g) shows their median clustering ratio $\frac{|BG^t|}{|cBG^t|}$ for the last time step tested.

6.7 Discussion and Future Work

The empirical results shown in this chapter demonstrate that lossless clustering offers dramatic performance gains on a diverse set of problems. However, since some domains cannot be clustered in this way, it remains unclear in exactly what types of problems lossless clustering is effective. This is a hard question, as it requires an analysis of the subclasses of Dec-POMDPs, a matter about which relatively few results are known. Most research has focused on analysis of methods, rather than of properties of Dec-POMDP problems, notable exceptions being (Pynadath and Tambe, 2002a; Goldman and Zilberstein, 2004). Although a detailed analysis is beyond the scope of this chapter, some observations based on empirical results have been provided.

Boularias and Chaib-draa (2008) also present an algorithm that improve efficiency of optimal solutions by a form of compression. The performance of their algorithm, however, stays behind when compared to GMAA*-Cluster. Although a more careful analysis is needed, there are two main reasons that can explain this. First, the compression of Boularias and Chaib-draa works on the exponentially larger space of policies, while GMAA*-Cluster works on an exponentially smaller space of histories. Second, GMAA*-Cluster can exploit knowledge of the initial state distribution $\boldsymbol{b}^0$.

The criterion for clustering is quite strict and there will also be many problems in which little or no lossless clustering is possible. In the future, we plan to consider approximations for such cases. In particular, one idea is to cluster approximately PE histories, e.g., if Kullback-Leibler divergence is below some threshold. Another idea is to cluster histories that induce the same *individual belief* over states:

$$\Pr(s|\vec{\theta}_i) = \sum_{\vec{\boldsymbol{\theta}}_{\neq i}} \Pr(s,\vec{\boldsymbol{\theta}}_{\neq i}|\vec{\theta}_i). \tag{6.7.1}$$

Such individual beliefs literally summarize the criterion and may therefore perform quite well in practice. Further investigation is needed to determine for which classes of problems such approximations might work.

Chapter 7

Conclusions and Discussion

This final chapter presents conclusions and summarizes some of the contributions made. Section 7.2 discusses in how far the presented research meets its goals and what the most important issues are that are left for future work.

7.1 Conclusions

This section first describes the big picture presented in this thesis. Subsequently, we list the contributions per chapter in more detail. We also take a brief moment to reflect on what the current state of Dec-POMDP solution methods is after four years of research by both others and ourselves in Subsection 7.1.3.

7.1.1 The Big Picture

The work described in this thesis is motivated by the desire to increase the performance of human decision making with the aid of decision support systems. Especially in complex dynamic environments in which multiple agents interact, as encountered in crisis management for instance, there is substantial room for improvement and the payoff of even slight improvements can be great. One of the most challenging requirements of such decision support systems is the capability to generate plans for teams of agents collaborating in a highly uncertain environment, and this has been the core topic of this thesis.

In particular, we adopted the framework of the *decentralized partially observable Markov decision process (Dec-POMDP)* to formalize the decision making process of a team of agents in a stochastic, partially observable environment. A Dec-POMDP provides a principled way to tackle the uncertainties in an environment, but this expression power comes at a cost: optimally solving a Dec-POMDP is NEXP-complete. For computing a bounded approximation the same worst case complexity result holds. This means that in order to have any chance of scaling to large problems, efficient unbounded approximations have to be developed that

work well in practice. Still, this thesis also considered optimal methods for Dec-POMDPs, as we believe that those form the basis for such efficient approximations.

This thesis introduced a framework of value-based planning for Dec-POMDPs facilitated by interpreting them as series of Bayesian games (BGs). *Forward-sweep policy computation (FSPC)* is a technique that solves these BGs one stage at a time starting with $t = 0$ and was first introduced by Emery-Montemerlo et al. (2004). By showing that there exists an optimal Q-value function Q^* that can be used as the payoff function for the BGs, and that doing so will result in an optimal joint policy, Chapter 3 showed that such modeling using BGs is exact.

Since computation of Q^* is intractable, Chapter 4 introduced approximate Q-value functions. These can be employed in value-based policy search, that generalizes FSPC by allowing backtracking. As such, this framework unifies the previous solution methods by Emery-Montemerlo et al. (2004) and Szer and Charpillet (2005) and presents a generalized 'forward perspective' of Dec-POMDPs. The solution method, dubbed generalized multiagent A* (GMAA*), can be considered a generalized 'top-down approach', as opposed to the 'bottom-up' dynamic programming approaches proposed by Hansen et al. (2004) and derivative works.

Even though GMAA* provides a way to compute both approximate and exact solution methods, scaling is limited: the BGs that represent a Dec-POMDP grow exponentially with respect to both the number of agents and the planning horizon. These sources of intractability were addressed independently of each other in Chapters 5 and 6 respectively.

When all agents in a Dec-POMDP closely interact, there is little hope of efficiently computing solutions for a large number of agents. However, in many problems interaction is sparse, meaning that each agent will only interact with a relatively small number of other agents. Such independence is exploited in Chapter 5 using *factored* Dec-POMDPs with additive rewards. In particular, it was shown that such models can be represented as series of *collaborative graphical BGs (CG-BGs),* that can be much more compact than regular BGs. The solution methods of Chapter 4 can be transferred to this new setting, leading to Factored GMAA*. In order to make this possible, the chapter explained how to efficiently 1) construct CGBGs, 2) compute factored Q-value functions to be used as a heuristic payoff function in the CGBGs, and 3) solve CGBGs. An extensive empirical evaluation demonstrated the potential of this approach, solving problems with up to 1000 agents, where the maximum number of agents previously considered in a general Dec-POMDP (i.e., without imposing assumptions of transition and observation independence) was 5.

The planning horizon is the second source of intractability for the GMAA* algorithm, because the size of the BGs depend on the number of *types*, i.e., (action) observation histories, for each agent and the number of such histories grows exponentially in the planning horizon. In Chapter 6, we reduced the computational costs of GMAA* by clustering histories. This idea was first considered by Emery-Montemerlo et al. (2005), however, their approach uses an ad-hoc heuristic to determine which histories to cluster and consequently finds only approximate solutions. By contrast, we identified a criterion that *guarantees* that two individual histories have the same optimal value, allowing *lossless clustering* and therefore

faster optimal solutions of Dec-POMDPs. Chapter 6 showed how this result can be exploited in optimal policy search and demonstrates empirically that it can provide a speed-up of multiple orders of magnitude, allowing the optimal solution of significantly larger problems.

Throughout the entire thesis we have considered variants of both Dec-POMDPs and Bayesian games and established a close relation between these two. In particular we established that *given a past joint policy* the expected value of continuing optimally afterwards can be represented by a BG. It was also explained how this relates to the notion of *sequential rationality* in extensive games with imperfect information. As such, we established strong links between Dec-POMDPs and game theory, and provided a basis for establishing similar links between game theory and partially observable stochastic games (POSGs).

7.1.2 Specific Contributions

In the following we treat in more detail the most important contributions made in each chapter.

Chapter 2. In this chapter, an in-depth introduction to finite-horizon Dec-POMDPs and their solution methods is provided. As such, it can serve as an introductory text to Dec-POMDPs. An alternative introduction is given by Seuken and Zilberstein (2008), who give a broader overview of different decentralized multiagent models and their solution methods, but provide less examples and details on the Dec-POMDP itself. Also, Chapter 2 includes background on relevant game-theoretic models.

Chapter 3. This chapter provided a framework of Q-value functions for Dec-POMDPs providing a significant contribution to fill this gap in Dec-POMDP theory. The main contributions are for the setting without communication, where it is shown how an optimal joint policy $\boldsymbol{\pi}^*$ induces an optimal Q-value function $Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^t, \mathbf{a})$ and how it is possible to construct an optimal joint policy $\boldsymbol{\pi}^*$ using forward-sweep policy computation. Because $Q_{\boldsymbol{\pi}^*}$ implicitly depends on the optimal past joint policy, there is no clear way to compute it directly. To overcome this problem, we introduced a different description of the optimal Q-value function that makes the dependence on the past joint policy explicit. This new description of Q^* can be computed using dynamic programming and can then be used to construct $\boldsymbol{\pi}^*$.

Another important contribution of this chapter is that it shows that a decrease in communication delay cannot lead to a decrease in expected return. That is, shorter communication delays are not harmful. Although this result is intuitively obvious, to the author's knowledge a formal proof had been lacking thus far. Finally, the chapter has presented a unified overview of the optimal value functions under various delays of communication and discussed how they relate to each other.

Chapter 4. Computing the optimal Dec-POMDP Q-value function Q^* is computationally too expensive, therefore Chapter 4 examined approximate Q-value

functions that can be calculated more efficiently and we discussed how they relate to Q^*. This chapter covered $\mathrm{Q}_{\mathrm{MDP}}$, $\mathrm{Q}_{\mathrm{POMDP}}$, and Q_{BG}, which was recently proposed by Oliehoek and Vlassis (2007a). Also, using the fact that decreasing communication delays in decentralized systems cannot decrease the expected value we established that $Q^* \leq \mathrm{Q}_{\mathrm{BG}} \leq \mathrm{Q}_{\mathrm{POMDP}} \leq \mathrm{Q}_{\mathrm{MDP}}$.

Additionally we showed how the approximate Q-value functions can be used as heuristics in a generalized policy search method GMAA*, thereby presenting a unified perspective of forward-sweep policy computation and the recent Dec-POMDP solution techniques of Emery-Montemerlo et al. (2004) and Szer et al. (2005).

Finally, an empirical evaluation of GMAA* shows significant reductions in computation time when using tighter heuristics to calculate optimal policies. Although there are no guarantees of any sort, using Q_{BG} generally leads to better approximate solutions in forward-sweep policy computation and the 'k-best joint BG policies' variant (k-GMAA*).

Chapter 5 introduced factored Dec-POMDPs with additive rewards. For such models, the scopes of the factored value function grow when going back in time, and will become fully coupled at some point. Nevertheless, Chapter 5 showed that locality of interaction holds at each stage *given* the scopes of the value function for that stage.

Subsequently, we showed how a factored Dec-POMDP can be represented by a series of collaborative graphical Bayesian games (CGBG). However, constructing them exactly requires performing exact inference over the space of states and joint histories, whose size is exponential in the number of state factors and the number of agents respectively. Therefore, we proposed to construct them using approximate inference and in particular the factored frontier algorithm, exploiting intermediate steps of computation to construct the CGBGs.

To use CGBGs in the solution of factored Dec-POMDPs, we needed a factored Q-value function to serve as payoff function for the CGBGs. Using the optimal Q-value function is impractical, because it is computationally too expensive and because it is fully coupled for earlier stages. Therefore we proposed to compute approximate value function *given a predetermined scope structure* that specifies the scopes for each stage for each component of the factored Q-value function. In particular we proposed two methods to compute a factored approximation of the $\mathrm{Q}_{\mathrm{MDP}}$ value function: naive regression (NR) and approximate dynamic programming (ADP). We discussed that projection to factored value functions with predetermined scopes can be performed by defining *induced indicator basis functions,* and that this projection is particularly efficient because the inner product can be computed very efficiently. We also proposed a third way of computing an approximate Q-value function, namely *transfer planning (TP)* that uses an approximate (e.g., $\mathrm{Q}_{\mathrm{MDP}}$) value function for a smaller source problem that involves a smaller number of agents. We refer to the resulting approximate Q-value function as Q_{TP}.

To optimally solve the CGBGs it is possible to employ non-serial dynamic programming that scales exponentially in the induced width of the interaction

graph (w). In the worst case, w is equal to the number of agents (n), but in typical settings $n \gg w$ yielding an exponential speedup. Another option is to convert a CGBG to a *type-action factor graph* and to use MAX-PLUS—a method based on belief propagation that finds a maximum through message passing—to find an approximate solution. These results may also be used in settings other than Dec-POMDPs.

Based on this framework, we proposed an optimal algorithm that can exploit last-stage indepencence, called GMAA*-ELSI. Approximations can be found using FACTORED FSPC and FACTORED k-GMAA* using the approximate Q-value functions mentioned above as heuristic. Empirical evaluation showed that GMAA*-ELSI computed an optimal solution two orders of magnitude faster than regular GMAA*. To our knowledge, these are the first optimal experimental results for general (factored) Dec-POMDPs with more than 2 agents. FACTORED FSPC was compared against the state-of-the-art methods for solving Dec-POMDPs. The results showed it is efficient and scales well with respect to the number of agents, but that the quality of the found solutions depends very much on the used heuristic. Q_{TP} consistently performed very well, finding (near-) optimal policies in little time. For instance, we were able to compute good policies for up to 1000 agents in a matter of seconds.

Chapter 6. We introduced a method for lossless clustering of action-observation histories in Dec-POMDPs, which can be applied in GMAA* policy search for Dec-POMDPs via Bayesian games.

This chapter showed in the context of general BGs that *commitment*, i.e., when an agent is determined to select the same actions for two of its types, allows for the reduction of a BG. It also showed *when* such commitments are sensible. In particular, a rational agent is committed to take the same actions for two types when 1) the probability distributions they induce over the types of the other agents are the same and 2) if they specify the same payoffs. These results may also be used outside the context of Dec-POMDPs.

Subsequently, these results were used within the context of Dec-POMDPs. Rather than applying an ad-hoc clustering of the types in the BGs representing a Dec-POMDP, a probabilistic equivalence criterion was identified that *guarantees* that, given a particular past joint policy, two action-observation histories of an agent at stage t have the same optimal Q-values and therefore can be clustered without loss in solution quality. The optimal algorithm resulting from applying such clustering within GMAA* is referred to as GMAA*-Cluster.

Empirical evaluation of GMAA*-Cluster demonstrated that for several domains a speedup of multiple orders of magnitude is achieved by clustering. It also investigated the amount of clustering possible for random past policies $\boldsymbol{\varphi}^t$, the result of which suggests that our clustering method may also be exploited in other algorithms. Also it may perhaps help to determine the number of policies one should retain in methods as MBDP (Seuken and Zilberstein, 2007a) and its variants.

7.1.3 Current State of Affairs

Both the work described in this thesis and other research performed on Dec-POMDPs in the last four years have significantly increased the size of problems that can be addressed. Especially, clustering within BGs (Emery-Montemerlo et al., 2005) and MBDP (Seuken and Zilberstein, 2007a) and its variants have significantly extended the horizons over which approximate solutions can be found. On the other hand, the research in this thesis contributes to allow scaling in the number of agents (both in exact and approximate settings), and the horizon in optimal settings.

At this point it may be possible to employ Dec-POMDP techniques in simple real-world problems. For instance, simple load balancing tasks may closely resemble the FireFightingGraph benchmark problem. Also, settings such as simple sensor networks, in which assumptions such as transition and observation independence hold, may be candidate for employing Dec-POMDP technology in the near future.

Application in more complex sensor networks and cooperative robotics settings, are still problematic at this point. Especially continuous state, action and observation spaces, which typically enter the picture in these settings, are still not addressed by current Dec-POMDP research. The same holds for applications within crisis management settings, although it may be possible to use Dec-POMDPs or other DTP methods for small parts of the problem as suggested for RoboCup Rescue (see Section 1.6) or recently tested in a field exercise (Maheswaran, Rogers, Sanchez, Szekely, Gati, Smyth, and VanBuskirk, 2009).

7.2 Discussion and Future Work

In this section we self-criticize the research performed and discuss the value of the Dec-POMDP model and decision-theoretic planning (DTP) in general. The statements made here, more than in other parts, express the opinion of the author. Future work is discussed in Subsection 7.2.4.

7.2.1 Scalability of Dec-POMDPs

Scalability issues are always point of criticism for decision-theoretic planning methods, and even more so for Dec-POMDPs, since even finding a bounded approximation is NEXP complete. As such, there really is not much hope of scaling up (ϵ-approximate) solutions for general Dec-POMDPs and one can be inclined to believe that "it will never work in real settings". Although—this is a matter of faith more than science—there is no definite answer to this question, this does not decrease the value of this line of research.

In particular, getting DTP and especially Dec-POMDPs to work on larger real-world problems is a difficult, but important problem and much research is still to be performed. This thesis makes a contribution to this work. Even if in the end Dec-POMDPs turn out to be impractical for some of the tasks we would like them to solve, this knowledge itself will be valuable. Also, single-agent POMDPs

were also considered intractable beyond any hope some years ago. However, in recent years POMDPs have been applied on robots, e.g., see Simmons and Koenig (1995); Roy, Gordon, and Thrun (2005); Spaan and Lima (2009), and in human-computer interaction settings (Roy, Gordon, and Thrun, 2003; Boger, Poupart, Hoey, Boutilier, Fernie, and Mihailidis, 2005; Hoey, von Bertoldi, Poupart, and Mihailidis, 2007; Doshi and Roy, 2008). And even the Dec-POMDP algorithm of Emery-Montemerlo (2005) has been run on real robots.

7.2.2 Robustness and Flexibility

Still, there are valid points of criticism with respect to Dec-POMDPs, as exposed by the following question: which of the motivating properties of multiagent systems in Section 1.3 are actually satisfied? Clearly, at this point there is no speedup due to asynchronous and parallel computation, although such method may still be developed.

However, in the author's opinion a more critical problem is that the Dec-POMDP does not facilitate robustness and flexibility: a plan is constructed for exactly n agents and if one fails the plan becomes useless. This weakness seems to be coupled with the focus on optimal plans, since it provides all the tools to compute an optimal plan for exactly n agents. Providing robustness and flexibility will necessarily come at the cost of optimality.

7.2.3 The No-Communication Assumption

This thesis considered planning for MASs in a stochastic partially observable environment and it assumed that no explicit communication between agents is possible. It is a reasonable question to ask whether these assumption are not too harsh. Especially the assumption with respect to communication may be debated since in most settings communication may be available.

Of course, there are some true non-communicative settings. Problems in which no communication is possible can for instance be found in space exploration, military domains and espionage and in games (e.g., bridge). Another example is given by communication networks, since meta-level communication may be prohibitively expensive, they are also candidate to be modeled as Dec-POMDPs. Still, the number of applications without any communication may be limited. However, as was explained in Section 2.8.3, a Dec-POMDP with communication (Dec-POMDP-Com) can be transformed to a regular Dec-POMDP. Therefore, more accurately stated, the assumption is that there is no mechanism specified to process the semantics of messages. Rather the found policy determines (embeds) the *optimal meaning* of the sent messages. As such, communication networks where some meta-level messages are possible are also a good candidate for being modeled as a Dec-POMDP. Actually in almost all settings communication is something one would want to optimize, as in nearly no setting is communication completely without cost: it consumes resources such as time and battery power. However, it is not clear whether treating communication in the 'embed-the-optimal-meaning' sense is useful for all these domains.

Nevertheless, the Dec-POMDP model may still be of substantial influence also for such other domains. For instance, no-communication periods may be embedded within communicative settings: in a lot of settings, there will be some communication possible, but typically not at every time step and the amount of information transferred may be limited. This limitation may follow from saturation of the communication network or there simply being not enough time (for instance, consider the communication that takes place between soccer players during a match). As such, it is reasonable to think that agents may act independently for several time-steps in between communications (Goldman and Zilberstein, 2008).

Finally, decision-theoretic models such as Dec-POMDPs can contribute in the analysis of other approaches. For instance, they can be used to evaluate and subsequently improve policies resulting from other multiagent frameworks, such as BDI frameworks. An example is given by Nair and Tambe (2005) who analyze the allocation of roles. Other research has analyzed existing communication policies (Pynadath and Tambe, 2002b; Xuan et al., 2001). Closely related, Dec-POMDPs also lend themselves very well to perform an analysis of the value of communication in particular settings. I.e., the no-communication (Dec-POMDP) setting gives a baseline performance for any method that does use communication, because using communication should increase this performance. In fact, special types of Dec-POMDPs have been used to determine *when* to communicate (Beynier and Mouaddib, 2005, 2006; Becker et al., 2005; Carlin and Zilberstein, 2009). Also, it is well conceivable that Dec-POMDP frameworks can help to determine the impact of different information, and thus can help identifying *what* to communicate.

7.2.4 Future Work

This section briefly discusses what the author considers to be the most important directions of future research. More detailed ideas for future work were presented at the end of the chapters.

An obvious, but important point of future work is the integration of factored Dec-POMDPs (i.e. FACTORED GMAA*) and clustering (GMAA*-Cluster). This line of work is not without challenges. When considering approximate clustering based on the heuristic payoff function, we now need to account for the fact that this heuristic is factored. When extending optimal clustering to these settings the criterion can be adapted to

$$\forall_{\vec{\boldsymbol{\theta}}_{\mathcal{N}(i)}} \forall_s \quad \Pr(s,\vec{\boldsymbol{\theta}}_{\mathcal{N}(i)}|\vec{\theta}_{i,a}) = \Pr(s,\vec{\boldsymbol{\theta}}_{\mathcal{N}(i)}|\vec{\theta}_{i,b}), \tag{7.2.1}$$

where $\mathcal{N}(i)$ are the neighbors of agent i as specified by the scopes of the optimal value function, because locality of interaction holds. Still, in such settings it will be hard to scale up the horizon, since that will mean that the intermediate scopes will become fully connected, and thus evaluating (7.2.1) will need to check a number of joint histories $\vec{\boldsymbol{\theta}}_{\mathcal{N}(i)}$ that is exponential in the number of agents. This means that for large number of agents, we most likely need to settle for approximate clustering, which also needs more research. In particular, it is important to try and establish bounds on approximate clustering schemes and it would be interesting to see how well using individual beliefs over states can do in different problems.

Also, it is important to test whether for longer horizons, approximation with smaller scopes will still work. Since dependence spreads over time, the approximation may become worse with the number of stages. On the other hand, in a similar way to how bounded approximation are possible for inference (Boyen and Koller, 1998), this may also be the case for values.

Another issue for future work is the identification of tighter (admissible) heuristics and how to compute them efficiently. Although such improvements are often considered incremental, Chapter 4 showed that the impact of tighter heuristics can be major. Therefore this, in the author's opinion, is a very important direction of future work.

Finally, as discussed above, the current work in Dec-POMDPs aims at optimal behavior *given* n agents, which does not provide any robustness if one agent fails. Future work should consider more flexible extensions of Dec-POMDPs, that allow for adding and removing agents by having a flexible model specification and solution methods. In particular, we hope to build upon the factored Dec-POMDPs framework by decomposing two-stage dynamic Bayesian networks into modules that can be combined on the fly to form the DBN specifying the model.

Summary

Situations in which multiple decision makers influence an environment arise in many important current and future real-world problems such as crisis management, network control, robotic teams and distributed software applications. Making decisions in such multiagent systems is of crucial interest to artificial intelligence and related fields.

This thesis is concerned with the task of computing a plan for a team of cooperative agents. Many real-world planning tasks for such teams of agents are subject to uncertainty: both the outcome of the actions and the perception of the current state of the environment may be uncertain and each of the agents may have a different partial view of this environment. Also, the agents may be uncertain with respect to each other's actions. Such settings can be captured by the decentralized partially observable Markov decision process (Dec-POMDP), a decision-theoretic model that allows a principled treatment of the mentioned uncertainties. Unfortunately, computing an optimal plan, or *joint policy,* that specifies for each agent what to do in each possible situation is proven to be intractable and even finding a bounded approximation to the optimal solution is NEXP-complete. This means that for many interesting problems we have to resort to approximation methods that will not be able to guarantee a bound on the quality of the joint policy.

One option is to apply optimization methods such as genetic algorithms or cross entropy to find a joint policy. However, such methods do not exploit the structure of the problem nor do they provide any insight in how the found approximation relates to an optimal solution. Therefore, this thesis describes a *value-based* approach. For single-agent planning (as formalized by the Markov decision process) many algorithms exist that find an (approximate) solution by constructing an optimal *value function* that represents the expected cumulative reward from each state, and subsequently extracting an optimal policy from the value function. This thesis discusses how a similar procedure can be applied in decentralized settings by identifying optimal value functions for Dec-POMDPs. By using the optimal value function as the payoff function in a series of *Bayesian games (BGs)* the optimal policy can be found, thereby extending the solution method of Emery-Montemerlo et al. (2004), to which we refer as *forward-sweep policy computation (FSPC)*, to include the exact setting.

It may come as no surprise that computing an optimal value function is also intractable, therefore this thesis proposes to use approximate value functions that are easier to compute. In particular, it covers Q_{MDP} and Q_{POMDP} and proposes

a new approximation Q_{BG} and applies them in a heuristic policy search method dubbed *generalized multiagent* A* (GMAA*). GMAA* unifies FSPC and multiagent A* (MAA*) (Szer et al., 2005) and works by solving BGs for different stages. In a BG for a particular stage t, each agent has to select an action for each of its possible histories. By setting a parameter k to 1 GMAA* reduces to FSPC and gives an approximate solution, while for $k = \infty$ the behavior is identical to MAA* and the method is exact. Still, the scalability of GMAA* is limited by the fact that the BGs grow exponentially with respect to the number of agents and time (because the number of histories grows exponentially with time).

To counter the first type of growth, the thesis explores how independence between agents can be exploited: in typical problems not all agents will have to interact at the same time which leads to sparseness in interaction. We propose to exploit this sparseness by using collaborative graphical Bayesian games (CGBGs), which can be represented much more compactly than the regular BGs. For these CGBGs it is possible to efficiently find approximate solutions by converting them to a factor-graph and applying Max-Plus, a message passing algorithm that operates on this graph.

To reduce the growth induced by the number of histories, we consider clustering histories, an idea first introduced by Emery-Montemerlo et al. (2005). However, their approach uses an ad-hoc heuristic to determine which histories to cluster and consequently finds only approximate solutions. By contrast, the work presented in this thesis identifies a criterion that *guarantees* that two individual histories have the same optimal value, allowing *lossless clustering* and therefore faster optimal solutions of Dec-POMDPs and solutions over longer horizons.

The thesis closes with some general conclusions and a discussion of the main directions of future work for practical Dec-POMDP solutions.

Samenvatting

Er zijn veel situaties denkbaar waarin meerdere actoren, of agents, een gezamenlijke omgeving beïnvloeden met hun acties. Voorbeelden zijn te vinden in crisismanagement, netwerkregeling, teams van robots en gedistribueerde software applicaties. Deze zogenaamde multi-agent systemen zijn van cruciaal belang voor de verdere ontwikkeling van kunstmatige intelligentie.

Dit proefschrift heeft het plannen voor teams van samenwerkende agenten als onderwerp. Veel planningtaken voor zulke teams zijn onderhevig aan onzekerheden: het resultaat van een bepaalde actie kan moeilijk te voorspellen zijn, de staat van de omgeving waarin de agenten verkeren kan vaak slechts gedeeltelijk worden waargenomen en iedere agent kan een ander beeld van de omgeving hebben. Ook kunnen de agenten onzeker zijn over elkaars acties. Dergelijke situaties kunnen worden geformaliseerd binnen het raamwerk van *'decentralized partially observable Markov decision processes'* (Dec-POMDPs). De kracht van het Dec-POMDP model is dat het de genoemde vormen van onzekerheid op een gefundeerde manier behandelt. De expressiekracht van het model betekent echter ook dat het zeer moeilijk is er optimale plannen voor te berekenen, omdat de computationele complexiteit erg hoog is (het oplossen van een Dec-POMDP is NEXP-compleet). Het gevolg is dat we voor alle enigszins realistische problemen genoegen zullen moeten nemen met een (onbegrensde) benadering.

Een mogelijkheid is het toepassen van standaard optimalisatiealgoritmen. Het nadeel hiervan is echter dat ze geen inzicht verschaffen in het probleem. Daarom beschrijft dit proefschrift een aanpak die gebaseerd is op zogenaamde *'value functions'*. Bij het plannen voor één agent spelen value functions een centrale rol: het is mogelijk om een optimale value function te berekenen, die de hoogst mogelijke verwachte beloning voor alle toestanden representeert. Vervolgens kan een optimale policy worden afgeleid van de value function. Dit proefschrift laat zien hoe een soortgelijke procedure kan worden gedefinieerd voor Dec-POMDPs aan de hand van *Bayesian games (BGs)* en identificeert de bijbehorende optimale value functions. Hiermee wordt de oplossingsmethode van Emery-Montemerlo et al. (2004), *forward-sweep policy computation (FSPC)*, uitgebreid om ook exacte oplossingen te vinden.

Omdat het berekenen van optimale value functions zelf ook te complex is, worden een aantal benaderingen besproken. In het bijzonder worden Q_{MDP} en Q_{POMDP} beschouwd en wordt een nieuwe benadering, Q_{BG}, geïntroduceerd. Deze benaderingen worden toegepast in *generalized multiagent* A* (GMAA*), een nieuwe heuris-

tische zoekmethode die twee eerdere planningmethoden voor Dec-POMDPs—FSPC en multiagent A* (Szer et al., 2005)—verenigt en generaliseert. De methode is gebaseerd op het herhaaldelijk oplossen van BGs voor verschillende tijdstappen. Het oplossen van een BG voor een tijdstap t houdt in dat iedere agent een actie moet kiezen voor elke mogelijke sequentie van observaties. Een parameter k beïnvloedt het zoekproces, bij $k = 1$ is GMAA* gelijk aan FSPC en vindt het een benadering, voor $k = \infty$ is de methode gelijk aan MAA* en dus exact. GMAA* in zijn oorspronkelijke vorm is echter beperkt doordat de methode slecht schaalt met betrekking tot het aantal agenten en de *horizon*, het aantal tijdstappen waarvoor een plan wordt gezocht. Dit proefschrift biedt een oplossing voor deze beide problemen afzonderlijk.

Betere schaalbaarheid met betrekking tot het aantal agenten wordt gerealiseerd door de aanname dat er onafhankelijkheid is tussen agenten: in een typisch probleem is de interactie van een agent beperkt tot een klein aantal buren. Zulke beperkte interactie wordt benut door middel van collaborative graphical Bayesian games (CGBGs), welke een compactere representatie hebben dan normale BGs en efficiënter kunnen worden opgelost. Voor een CGBG is het mogelijk om zeer vlug benaderingen te vinden door deze te transformeren naar een factor-graaf en Max-Plus, een algoritme dat werkt door het sturen van berichten tussen knopen, op deze graaf te draaien.

Voor het verbeteren van de schaalbaarheid van GMAA* met betrekking tot de horizon, beschrijft dit proefschrift het clusteren van sequenties van observaties, een idee oorspronkelijk afkomstig van Emery-Montemerlo et al. (2005). Dit eerdere werk stelt voor te clusteren aan de hand van een heuristiek en vindt daarom een benadering. Dit proefschrift introduceert een criterium dat garandeert onder welke voorwaarden twee sequenties dezelfde optimale waarde hebben, en maakt het dus mogelijk te clusteren zonder verlies van waarde. Hierdoor is het mogelijk om optimale plannen vlugger te berekenen en te plannen voor meer tijdstappen dan eerst mogelijk was.

Het proefschrift sluit af met enkele conclusies en een bespreking van toekomstige richtingen van onderzoek voor praktische oplossingen voor Dec-POMDPs.

Appendix A

Problem Specifications

This section gives an overview of the benchmark problems used in for the experiments reported in this thesis. Table A.1 gives a numerical overview of the different problems. Also included in the table are the number of joint policies for some horizons, giving an idea of how complex the different problems are.

The Dec-Tiger and FireFighting were extensively treated in Section 2.3. This thesis also considered a slightly modified version version of Dec-Tiger, called Skewed Dec-Tiger, in which the start distribution is not uniform. Instead, initially the tiger is located on the left with probability 0.8. Other than that the problem is the same.

The BroadcastChannel was introduced by Hansen et al. (2004) and models two nodes that have to cooperate to maximize the throughput of a shared communication channel. At each stage the agents can chose to send or not send a message across the channel, and they can noisily observe whether there was a collision (if both agents sent) or not. This problem has 4 states, since every agent can or cannot have a message.

Furthermore, a test problem called "Meeting on a Grid" is provided by Bernstein et al. (2005), in which two robots navigate on a two-by-two grid. The goal of the agents occupy the same square and to this and they have 5 actions: move in any direction (N,W,E or S) or stand still. The outcome of these actions are stochastic. In this thesis we used GridSmall, a version with 2 observations per agent introduced by Amato et al. (2006). In this version, each agent can only observe if there is a wall on the left or right (i.e., whether it is in the left half or right half of the grid).

The Cooperative Box Pushing domain was introduced by Seuken and Zilberstein (2007b) and is a larger two-robot benchmark. Also in this domain the agents are situated in a grid world, but now they have to collaborate to move boxes in this world. In particular, there are small boxes that can be moved by 1 agent, and big boxes that the agents have to push together. Each agent has 4 actions: turn left, turn right, move forward and stay, and 5 observations that describe the grid position in front of the agent: empty, wall, other agent, small

	problem primitives				num. $\boldsymbol{\pi}$ for h		
	n	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}_i\rvert$	$\lvert\mathcal{O}_i\rvert$	2	4	6
Dec-Tiger	2	2	3	2	7.29e02	2.06e14	1.31e60
BroadcastChannel	2	4	2	2	6.40e01	1.07e09	8.51e37
GridSmall	2	16	5	2	1.563e04	9.313e20	1.175e88
Cooperative Box Pushing	2	100	4	5	1.68e7	6.96e187	Inf
Recycling Robots	2	4	3	2	7.29e02	2.06e14	1.31e60
Hotel 1	2	16	3	4	5.90e4	1.29e81	Inf
FireFighting	3*	27*	3*	2	1.97e4	2.94e21	1.50e90
FFG	3*	432 / 81*	2	2	5.12e02	3.52e13	7.85e56
Aloha	3*	27*	2	3	4.10e3	1.33e36	Inf

Table A.1: The number of joint policies for different problems and horizons. 'Inf' denotes a value beyond double machine precision. * denotes that this number can vary, listed are some chosen values: for FireFighting we chose $N_H = 3$ houses, for both FireFighting and FFG we assume $N_f = 3$ fire levels, for Aloha we assume a max. backlog of 2 messages. FireFighting lists two numbers for $|\mathcal{S}|$: it has 432 states when including the positions of the agents in the state description, otherwise it has 81 states for the mentioned parameter settings.

box, large box.

Amato et al. (2007a) introduced Recycling Robots, a problem domain in which 2 agents have to collect garbage. The agents have 3 actions, search for a small can, search for a large can and recharge the battery. Small cans can be collected by a single agent, but again, for large cans the robots have to cooperate.

A last two-agent problem that has been considered in this thesis is Hotel 1 (Spaan and Melo, 2008). In this problem there are two travel agents that at each stage can receive a customer. They can assign the client to a hotel, to a resort or refuse the client. Allocating the client to the hotel successfuly gives the highest reward. However, the hotel has a limited capacity and assigning the client when to hotel is full leads to a severe penalty. The resort is never full, but can also become crowded when both agents refer their client there, which therefore also results in a penalty.

The FireFightingGraph (FFG) and Aloha problem where extensively covered in Chapter 5. Note that Table A.1 only shows figures for the 3-agent variants. Clearly when increasing the number of agents, the complexity of the problems increase exponentially. For instance, for 10 agents FFG has $3^{11} = 177{,}147$ states and 1.07e9 joint policies for $h = 2$.

Appendix B

Immediate Reward Value Function Formulations

In Chapter 3, we discussed that there are two ways of formulation value functions for decentralized settings: the expected reward and immediate reward formulations. The former were treated in Chapter 3. Here we treat the latter, and discuss the relation between the two formulations.

B.1 k-Steps Delay Immediate Reward Formulation

As in the 1-step delay case, it is possible to rephrase the value functions for k-steps delay to an immediate reward (IR) formulation, as described by Oliehoek et al. (2008b). Again, in this immediate reward formulation, the value $V_k^{t,*}$ over stages $t, \ldots, ,h-1$ is expressed in terms of arguments for stage t: $\forall_{0 \leq t \leq h-k-1}$

$$Q_k^{t,*}(\boldsymbol{b}^t, \boldsymbol{q}^t, \boldsymbol{\beta}^{t+k}) = R(\boldsymbol{b}^t, \boldsymbol{q}^t(\vec{\mathbf{o}}_\emptyset)) + \sum_{\mathbf{o}^{t+1} \in \mathcal{O}} \Pr(\mathbf{o}^{t+1} | \boldsymbol{b}^t, \boldsymbol{q}^t(\vec{\mathbf{o}}_\emptyset)) V_k^{t+1,*}(\boldsymbol{b}^{t+1}, \boldsymbol{q}^{t+1}), \tag{B.1.1}$$

with $\boldsymbol{q}^t(\vec{\mathbf{o}}_\emptyset) = \mathbf{a}^t$ the joint action specified for the empty joint history, $\boldsymbol{q}^{t+1} = \langle \boldsymbol{q}^t \circ \boldsymbol{\beta}^{t+k} \rangle \Downarrow_{\mathbf{o}^{t+1}}$ and

$$V_k^{t,*}(\boldsymbol{b}^t, \boldsymbol{q}^t) \equiv \max_{\boldsymbol{\beta}^{t+k}} Q_k(\boldsymbol{b}^t, \boldsymbol{q}^t, \boldsymbol{\beta}^{t+k}). \tag{B.1.2}$$

For the last k stages, $h-k \leq t \leq h-1$, there are $\tau = h-t$ stages to go and we get

$$V_k^{t,*}(\boldsymbol{b}^t, \boldsymbol{q}^t_{|\tau|}) = R(\boldsymbol{b}^t, \boldsymbol{q}^t_{|\tau|}(\vec{\mathbf{o}}_\emptyset)) + \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1} | \boldsymbol{b}^t, \boldsymbol{q}^t_{|\tau|}(\vec{\mathbf{o}}_\emptyset)) V_k^{t+1,*}(\boldsymbol{b}^{t+1}, \boldsymbol{q}^{t+1}_{|\tau-1|}). \tag{B.1.3}$$

Note that (B.1.3) does not include a maximization over 'actions' $\boldsymbol{\beta}^{t+k}$. Therefore, the last k stages should be interpreted as a Markov chain. Standard dynamic

programming can be applied to calculate all $V_k^{t,*}(\boldsymbol{b}^t,\boldsymbol{q}^t)$-values, for all (joint beliefs induced by) all joint action-observation histories.

B.2 Conversion between Formulations

Clearly the immediate and expected reward formulations should have some relation to each other. In particular, the IR-value $V_k^{t,*}(\boldsymbol{b}^t,\boldsymbol{q}^t)$ can be decomposed into as the expected reward over the first k-steps $t,\ldots,t+k-1$, which we will denote using K_k, and the expected reward over the remaining steps, i.e., the expected reward (ER) formulation $V_k^{t+k,*}(\boldsymbol{b}^t,\boldsymbol{q}^t)$:

$$V_k^{t,*}(\boldsymbol{b}^t,\boldsymbol{q}^t) = K_k(\boldsymbol{b}^t,\boldsymbol{q}^t) + V_k^{t+k,*}(\boldsymbol{b}^t,\boldsymbol{q}^t). \tag{B.2.1}$$

In the following, we define, $K_k(\boldsymbol{b}^t,\boldsymbol{q}^t)$ through $K^{\tau=i}(\vec{\boldsymbol{\theta}}^t,\boldsymbol{q}^{\tau=i,t})$, the expected reward for the next i stages, i.e.,

$$K_k(\boldsymbol{b}^t,\boldsymbol{q}^t) \;\equiv\; K^{\tau=k}(\boldsymbol{b}^t,\boldsymbol{q}^t_{|k|}). \tag{B.2.2}$$

We then have $K^{\tau=1}(\boldsymbol{b}^t,\mathbf{a}^t) = R(\boldsymbol{b}^t,\mathbf{a}^t)$ and

$$K^{\tau=i}(\boldsymbol{b}^t,\boldsymbol{q}^t_{|i|}) = R(\boldsymbol{b}^t,\boldsymbol{q}^t_{|i|}(\vec{\mathbf{o}}_\emptyset))+ \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{q}^t_{|i|}(\vec{\mathbf{o}}_\emptyset))K^{\tau=i-1}(\boldsymbol{b}^{t+1},\boldsymbol{q}^{t+1}_{|i-1|}), \tag{B.2.3}$$

where $\boldsymbol{q}^{t+1}_{|i-1|} = \boldsymbol{q}^t_{|i|}\Downarrow_{\mathbf{o}^{t+1}}$ is the depth-$(i-1)$ joint policy that results from $\boldsymbol{q}^t_{|i|}$ after joint observation of $\mathbf{o}^{t+1}$.

B.3 Less Delay Cannot Decrease Value

Also for the IR formulation, the intuitive result that less delay cannot hurt, holds.

Theorem B.1 (Shorter communication delays cannot decrease the value)**.** *The optimal Q-value function Q_k of a finite horizon Dec-POMDP with k-steps delayed communication is an upper bound to Q_{k+1}, that of a $k+1$-steps delayed communication system. That is*

$$\forall_t \forall_{\boldsymbol{b}^t} \forall_{\boldsymbol{q}^t_{|k|},\boldsymbol{\beta}^{t+k}_{|k|}} \quad Q_k^{t,*}(\boldsymbol{b}^t,\boldsymbol{q}^t_{|k|},\boldsymbol{\beta}^{t+k}_{|k|}) \geq \max_{\boldsymbol{\beta}^{t+k+1}_{|k+1|}} Q_{k+1}^{t,*}(\boldsymbol{b}^t,\langle \boldsymbol{q}^t_{|k|} \circ \boldsymbol{\beta}^{t+k}_{|k|}\rangle,\boldsymbol{\beta}^{t+k+1}_{|k+1|}). \tag{B.3.1}$$

Proof. We start be rewriting the left hand side:

$$
\begin{aligned}
&Q_k^{t,*}(\boldsymbol{b}^t,\boldsymbol{q}^t_{|k|},\boldsymbol{\beta}^{t+k}_{|k|}) \\
&\quad = R(\boldsymbol{b}^t,\boldsymbol{q}^t_{|k|}(\vec{\mathbf{o}}_\emptyset)) + \sum_{\mathbf{o}^{t+1}\in\mathcal{O}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{q}^t_{|k|}(\vec{\mathbf{o}}_\emptyset)) V_k^{t+1,*}(\boldsymbol{b}^{t+1},\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle\Downarrow_{\mathbf{o}^{t+1}}) \\
&\quad = R(\boldsymbol{b}^t,\boldsymbol{q}^t_{|k|}(\vec{\mathbf{o}}_\emptyset)) + \sum_{\mathbf{o}^{t+1}\in\mathcal{O}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{q}^t_{|k|}(\vec{\mathbf{o}}_\emptyset)) \\
&\qquad \Big[K_k(\boldsymbol{b}^{t+1},\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle\Downarrow_{\mathbf{o}^{t+1}}) + V_k^{t+k+1,*}(\boldsymbol{b}^{t+1},\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle\Downarrow_{\mathbf{o}^{t+1}})\Big] \qquad \text{(B.3.2)} \\
&\quad = \left[R(\boldsymbol{b}^t,\boldsymbol{q}^t_{|k|}(\vec{\mathbf{o}}_\emptyset)) + \sum_{\mathbf{o}^{t+1}\in\mathcal{O}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{q}^t_{|k|}(\vec{\mathbf{o}}_\emptyset)) K_k(\boldsymbol{b}^{t+1},\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle\Downarrow_{\mathbf{o}^{t+1}})\right] + \\
&\qquad \left[\sum_{\mathbf{o}^{t+1}\in\mathcal{O}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{q}^t_{|k|}(\vec{\mathbf{o}}_\emptyset)) V_k^{t+k+1,*}(\boldsymbol{b}^{t+1},\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle\Downarrow_{\mathbf{o}^{t+1}})\right]. \qquad \text{(B.3.3)}
\end{aligned}
$$

Note that, for the right hand side, we have that

$$
\max_{\boldsymbol{\beta}^{t+k+1}_{|k+1|}} Q_{k+1}^{t,*}(\boldsymbol{b}^t,\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle,\boldsymbol{\beta}^{t+k+1}_{|k+1|}) = V_{k+1}^{t,*}(\boldsymbol{b}^t,\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle) \qquad \text{(B.3.4)}
$$

and therefore we can write

$$
V_{k+1}^{t,*}(\boldsymbol{b}^t,\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle) = K_{k+1}(\boldsymbol{b}^t,\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle) + V_{k+1}^{t+k+1,*}(\boldsymbol{b}^t,\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle) \quad \text{(B.3.5)}
$$

where, per definition (by (B.2.2) and (B.2.3))

$$
\begin{aligned}
&K_{k+1}(\boldsymbol{b}^t,\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle) = K^{\tau=k+1}(\boldsymbol{b}^t,\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle) = R(\boldsymbol{b}^t,\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle(\vec{\mathbf{o}}_\emptyset)) \\
&+ \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle(\vec{\mathbf{o}}_\emptyset)) K^{\tau=k+1-1}(\boldsymbol{b}^{t+1},\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle\Downarrow_{\mathbf{o}^{t+1}}). \qquad \text{(B.3.6)}
\end{aligned}
$$

Note that in this equation $\boldsymbol{\beta}^{t+k}_{|k|}$ does not influence the immediate reward or observation probability. Therefore the last equation is equal to the first part of (B.3.3). This means that we only have to show that

$$
\begin{aligned}
&\sum_{\mathbf{o}^{t+1}\in\mathcal{O}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{q}^t_{|k|}(\vec{\mathbf{o}}_\emptyset)) V_k^{t+k+1,*}(\boldsymbol{b}^{t+1},\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle\Downarrow_{\mathbf{o}^{t+1}}) \geq \\
&\qquad\qquad V_{k+1}^{t+k+1,*}(\boldsymbol{b}^t,\langle \boldsymbol{q}^t_{|k|}\circ\boldsymbol{\beta}^{t+k}_{|k|}\rangle). \qquad \text{(B.3.7)}
\end{aligned}
$$

However, this is exactly what Theorem 3.3 shows. Therefore (B.3.7) holds, concluding the proof. □

B.4 Summary of Q-value Functions for Decentralized Settings

This thesis defined value functions for different settings w.r.t the assumptions on communication. Also, two types of value functions were identified: expected and

	V-form	Q-form A	Q-form B
k-SD	$V_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}_{\|k\|}^{t-k})$	$Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}_{\|k\|}^{t-k},\boldsymbol{\beta}_{\|k\|}^{t})$	$Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}_{\|k\|}^{t-k},\vec{\boldsymbol{\theta}}_{\|k\|}^{t},\boldsymbol{\beta}_{\|k\|}^{t})$
h-SD	$V_h^{t,*}(\boldsymbol{b}^{0},\boldsymbol{\varphi}^{t})$	$Q_h^{t,*}(\boldsymbol{b}^{0},\boldsymbol{\varphi}^{t},\boldsymbol{\delta}^{t})$	$Q_h^{t,*}(\boldsymbol{b}^{0},\boldsymbol{\varphi}^{t},\vec{\boldsymbol{\theta}}^{t},\boldsymbol{\delta}^{t})$
1-SD	$V_1^{t,*}(\boldsymbol{b}^{t-1},\mathbf{a}^{t-1})$	$Q_1^{t,*}(\boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\boldsymbol{\beta}_{\|1\|}^{t})$	$Q_1^{t,*}(\boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\mathbf{o}^{t},\boldsymbol{\beta}_{\|1\|}^{t})$
0-SD	$V_0^{t,*}(\boldsymbol{b}^{t},())$	$Q_1^{t,*}(\boldsymbol{b}^{t},(),\mathbf{a}^{t})$	$Q_1^{t,*}(\boldsymbol{b}^{t},(),(),\mathbf{a}^{t})$

Table B.1: Expected reward formulations

	V-form	Q-form
k-SD	$V_k^{t,*}(\boldsymbol{b}^{t},\boldsymbol{q}_{\|k\|}^{t})$	$Q_k^{t,*}(\boldsymbol{b}^{t},\boldsymbol{q}_{\|k\|}^{t},\boldsymbol{\beta}_{\|k\|}^{t+k})$
h-SD	$V_h^{t}(\boldsymbol{b}^{t},\boldsymbol{q}_{\|h-t\|}^{t})$	$Q_h^{t}(\boldsymbol{b}^{t},\boldsymbol{q}_{\|h-t\|}^{t},())$
1-SD	$V_1^{t,*}(\boldsymbol{b}^{t},\mathbf{a}^{t})$	$Q_1^{t,*}(\boldsymbol{b}^{t},\mathbf{a}^{t},\boldsymbol{\beta}_{\|1\|}^{t+1})$
0-SD	$V_0^{t,*}(\boldsymbol{b}^{t},())$	$Q_0^{t,*}(\boldsymbol{b}^{t},(),\mathbf{a}^{t})$

Table B.2: Immediate reward formulations

immediate reward value functions. Here we we present an overview of all encountered value functions and make some remarks to provide a coherent perspective. For the general discussion we will consider the k-SD setting, as the other settings can be interpreted as special cases.

Table B.1 lists the expected reward (ER) formulations of the value functions for the considered settings. These value functions specify the expected reward over stages $t,\ldots,h-1$, given a distribution over states $\boldsymbol{b}^{t-k}$ that lies in the past. This past joint belief can be computed by the agents during execution and acts as a Markovian signal. Given the joint policy $\boldsymbol{q}_{|k|}^{t-k}$ used during the intermediate stages, at stage t we should select the $\boldsymbol{\beta}_{|k|}^{t}$ that maximizes the future reward, thereby defining V_k^* from Q_k^*. To avoid any confusion, Table B.1 shows both Q-forms used in the thesis.

The immediate reward (IR) formulations are listed in Table B.2. In contrast to ER formulations, these type of value functions express the expected value solely in terms of current joint belief $\boldsymbol{b}^t$ (induced by some joint action observation history) and future policies $\boldsymbol{q}_{|k|}^{t},\boldsymbol{\beta}_{|k|}^{t+k}$.

In general, one can interpret the expected reward formulations to correspond to the forward view of Dec-POMDPs (i.e. forward-sweep policy computation etc.): At stage t, there is common knowledge of $\boldsymbol{b}^{t-k},\boldsymbol{q}_{|k|}^{t-k}$ which defines a BG for that stage, the solution of which is the maximizing $\boldsymbol{\beta}_{|k|}^{t}$. In contrast, the immediate reward formulations correspond to the backwards view (i.e., DP methods): given that $\boldsymbol{q}_{|k|}^{t}$ will be used for the 'last' k stages, $V_k^{t,*}(\boldsymbol{b}^{t},\boldsymbol{q}_{|k|}^{t})$ specifies the value of joint belief $\boldsymbol{b}^t$. For instance, $V_h^{t}(\boldsymbol{b}^{t},\boldsymbol{q}_{|h-t|}^{t})$ gives the expected value of executing $\boldsymbol{q}_{|h-t|}^{t}$ starting from (the state distribution $\boldsymbol{b}^t$ induced by) $\vec{\boldsymbol{\theta}}^t$. That is, $\boldsymbol{q}_{|h-t|}^{t}$ is a joint

sub-tree policy that executes for the remainder of time; stages $t,\ldots,h-1$. This is exactly the relation between the optimal value function and DP for Dec-POMDPs as discussed in Section 3.1.5.1.

It is important to note that Q-form B, $Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}_{|k|}^{t-k},\vec{\boldsymbol{\theta}}_{|k|}^{t},\boldsymbol{\beta}_{|k|}^{t})$, is a function of the joint belief $\boldsymbol{b}^{t-k}$ (induced by $\vec{\boldsymbol{\theta}}^{t-k}$), but *not* of $\boldsymbol{b}^t$ (as would be induced by $\vec{\boldsymbol{\theta}}^t = (\vec{\boldsymbol{\theta}}^{t-k},\vec{\boldsymbol{\theta}}_{|k|}^{t})$: keeping other arguments the same two histories $\vec{\boldsymbol{\theta}}_{|k|}^{t},\vec{\boldsymbol{\theta}}_{|k|}^{t\prime}$ that would induce the same joint belief $\boldsymbol{b}^t$, *can* have different values, because $\vec{\boldsymbol{\theta}}_{|k|}^{t}$ specifies how *individual information* is distributed. For instance, let us the Dec-Tiger problem with 2-steps delayed communication, $\boldsymbol{b}^{t-2}$ is uniform and the intermediate joint policy $\boldsymbol{q}_{|2|}^{t-2}$ specifies only to listen. Now we consider two histories $\vec{\boldsymbol{\theta}}_{|2|}^{t} = \langle(o_{\text{HL}},o_{\text{HL}}),(o_{\text{HR}},o_{\text{HR}})\rangle$ while $\vec{\boldsymbol{\theta}}_{|2|}^{t\prime} = \langle(o_{\text{HL}},o_{\text{HR}}),(o_{\text{HL}},o_{\text{HR}})\rangle$. Even though both 2-step joint histories will lead to the same joint belief (a uniform belief over states), the value of the latter will be higher because the agents have a identical individual beliefs. In case of $\vec{\boldsymbol{\theta}}_{|2|}^{t} = \langle(o_{\text{HL}},o_{\text{HL}}),(o_{\text{HR}},o_{\text{HR}})\rangle$, however, the agents have a completely different perspective of the world (and are fairly sure that their view is right), therefore the optimal joint policy may specify to open the door in these cases, leading to a lower value for this particular joint history. A more formal argument is given by Oliehoek et al. (2007c).

We end this section by making some specific remarks about the value function for the special cases considered.

The 0-SD setting. Table B.1 and B.2 show that many of the arguments of the general formulation become degenerate under instantaneous communication. E.g., for $k=0$ $\boldsymbol{q}_{|k|}^{t-k} = ()$ is an empty joint sub-tree policy. The result is that the IR and ER formulation are identical: $V_0^{t,*}(\boldsymbol{b}^t)$ and $Q_0^{t,*}(\boldsymbol{b}^t,\mathbf{a}^t)$.

The 1-SD setting. Under 1-step delayed communication, the joint policy followed since $\boldsymbol{b}^{t-1}$ reduces to a joint action $\boldsymbol{q}_{|1|}^{t-1} = \mathbf{a}^{t-1}$. Also, as discussed in Section 3.3.1, $Q_1^{t,*}(\boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\mathbf{o}^t,\boldsymbol{\beta}_{|1|}^{t})$ can be immediately reduced to $V_1^{t,*}(\boldsymbol{b}^t,\mathbf{a}^t)$, by noticing that $\boldsymbol{b}^{t-1},\mathbf{a}^{t-1},\mathbf{o}^t$ only influence the value through $\boldsymbol{b}^t$ and that for $\mathbf{o}^t$ $\boldsymbol{\beta}_{|1|}^{t}$ reduces to $\mathbf{a}^t = \boldsymbol{\beta}_{|1|}^{t}(\mathbf{o}^t)$.

A second observation is that $V_1^{t,*}(\boldsymbol{b}^t,\mathbf{a}^t)$ has the same arguments as $Q_0^{t,*}(\boldsymbol{b}^t,\mathbf{a}^t)$. Because of the apparent similarity, $V_1^{t,*}(\boldsymbol{b}^t,\mathbf{a}^t)$ has been denoted as 'Q' in previous work. We still refer to this function as Q_{BG} when used as a heuristic to stay in line with Q_{MDP},Q_{POMDP} naming.

The h-SD setting. Because ER formulations have their common joint belief at stage 0, $\boldsymbol{q}_{|k|}^{t-k}$ reduces to a full past joint policy $\boldsymbol{\varphi}^t$. Also, because the joint BG-policies $\boldsymbol{\beta}_{|k|}^{t}$ are mappings from the entire history, they reduce to joint decision rules $\boldsymbol{\delta}^t$.

In the IR formulation Q_h^t is a degenerate form that reduces to $V_h^t(\boldsymbol{b}^t,\boldsymbol{q}_{|h-t|}^{t})$. Also, it is not starred '$*$' since there is no "continuing optimally afterwards".

Appendix C

Formalization of Regression to Factored Q-Value Functions

C.1 Local State-Action Pairs and Indicator Functions

Formalizing the decomposition of the Q_{MDP} function into a factored Q-value function through regression requires the formalization of several concepts. In particular, linear regression is effectively a projection onto a set of basis functions, so we need to define the basis functions on which $Q^t_M(s,\mathbf{a})$ is projected, such that the resulting approximation is of the desired form: a factored Q-value function with the desired scopes. To this end we use the concept of indicator functions.

Definition C.1 (Indicator function)**.** Let $f : \mathcal{X} \to \{0,1\}$ be a function mapping from some (finite) set of domain elements $\mathcal{X}$ to $\{0,1\}$. f is an *indicator function* if and only if $f(x) = 1$ for exactly one $x \in \mathcal{X}$. This x is called the *indicated* (domain) element.

C.1.1 (State,Action)-Pairs.

In the regression problem expressed by (5.4.2), the domain of the functions are (state, action)-pairs, for which we first introduce the notation. In the following we will assume that we are discussing the regression problem for a particular stage t. We drop the index for this stage to ease the notation.

Definition C.2 ((state, joint action)-pair)**.** Let z denote a *(state, joint action)-pair (SAP)*, $z \equiv (s,\mathbf{a})$. I.e., we have that $z_1,\ldots,z_N$ with $N = |\mathcal{S}|\,|\mathcal{A}|$ is the domain of Q_M.

Similarly, for each local Q-function we have the notion of local SAP.

Definition C.3 (Local (state, joint action)-pair)**.** Let $\hat{z}$ denote a l*ocal SAP (LSAP)* for local Q-function Q^e. That is, $\hat{z}_e \equiv \langle \mathbf{x}_{\mathbb{X}(e,t)},\mathbf{a}_{\mathbb{A}(e,t)}\rangle$.

The number of LSAPs for Q^e (i.e., the size of its domain) is denoted $\hat{N}_e = |\mathcal{X}_{\mathbb{X}(e)} \times \mathcal{A}_{\mathbb{A}(e)}|$. Also, l^e is used to index into the LSAPs of Q^e: $1 \leq l^e \leq \hat{N}_e$. Let $I^e(\cdot)$ be the index function for Q^e: it computes the local index $l^e = I^e(\hat{z}_e)$. The total number of LSAPs (i.e., for all local Q-functions) is $\hat{N} = \sum_{e=1}^{|\rho|} \hat{N}_e$. We use l to index into all LSAPs : $1 \leq l \leq \hat{N}$.

C.1.2 Scope Restriction and Induced Scope.

Before we can properly define the proposed basis functions, we need to introduce the concept of scope restriction and induced scope.

Definition C.4 (Scope restriction function)**.** We use $R_{\mathbb{S}(e)}(\cdot)$ to denote the scope restriction function for Q^e: it restricts s,$\mathbf{a}$ to $\mathbb{S}(Q^e)$, the scope of Q^e. We write $R_{\mathbb{S}(e)}(s,\mathbf{a}) = \langle \mathbf{x}_{\mathbb{X}(e)}, \mathbf{a}_{\mathbb{A}(e)} \rangle = \hat{z}_e$.

Definition C.5 (Induced function and scope)**.** For any function $f(s,\mathbf{a})$, if f can be defined as

$$f(s,\mathbf{a}) = f'(R_{\mathbb{S}}(s,\mathbf{a})), \tag{C.1.1}$$

where $R_{\mathbb{S}}(\cdot)$ is a scope restriction function that restricts to a scope $\mathbb{S}$ and where f' is a function over the smaller scope $\mathbb{S}$, then f is said to have *induced scope* $\mathbb{S}$. Similarly we refer to f as the *induced function*.

By using the scope restriction function, we overload the index function to also work on SAPs:

$$I^e(s,\mathbf{a}) \equiv I^e(R_{\mathbb{S}(e)}(s,\mathbf{a})) = l^e. \tag{C.1.2}$$

This means that $I^e(s,\mathbf{a})$ has induced scope $\mathbb{S}(Q^e)$.

C.1.3 The Basis Functions: Mapping SAPs to LSAPs.

The basis functions we propose to use for regression can be seen as mappings from SAPs to LSAPs. In particular for each LSAP $\hat{z}$ with index l, we introduce a basis function h_l such that $h_l(z)$ indicates whether a SAP z is 'consistent' with $\hat{z}$.

We start by defining 'local indicator functions' $\hat{h}$ for each local Q-function Q^e.

Definition C.6 (Local indicator function)**.** For each local Q-function Q^e, for each LSAP $\hat{z}_e$ and corresponding index l^e, let $\hat{h}_{l^e}(\hat{z}'_e)$ be the *local indicator function* for LSAP $\hat{z}_e$ that act as a filter that indicates whether $\hat{z}'_e = \hat{z}_e$.

$$\hat{h}_{l^e}(\hat{z}'_e) = \begin{cases} 1 & \text{if } I^e(\hat{z}'_e) = l^e, \\ 0 & \text{otherwise.} \end{cases} \tag{C.1.3}$$

I.e., $\hat{h}_{l^e}(\hat{z}'_e)$ returns whether $\hat{z}'_e$ is the indicated element.

Using the definition of local indicator functions, we can finally define the basis functions needed for linear regression. In particular $h_l(z)$ is the basis function that indicates whether global SAP z is consistent with the LSAP with global index l. We call such basis functions *basis functions induced through local indicator functions* or simply *induced indicator basis functions.*

Definition C.7 (Induced indicator basis function)**.** For each LSAP with global index l and local index l^e, there is an associated basis function h_l that has induced scope $\mathbb{S}(Q^e)$ and is defined as

$$h_l(s,\mathbf{a}) \equiv \hat{h}_{l^e}(R_{\mathbb{S}(e)}(s,\mathbf{a})). \tag{C.1.4}$$

An induced indicator basis function h_l can be seen as a column vector:

$$h_l = \begin{pmatrix} h_l(z_1) \\ \vdots \\ h_l(z_N) \end{pmatrix} \tag{C.1.5}$$

in which the SAPs that are consistent with l are 1 and other entries are 0. Note that this means h_l is not an indicator function itself (an indicator function is '1' for exactly one entry).

C.2 Efficient Projections

Section 5.4.3 introduced an algorithm to compute a factored $\mathrm{Q}_{\mathrm{MDP}}$ approximation more efficiently than naive regression, by bootstrapping from the previous approximation. Still, this algorithm needs to perform a projection, i.e. compute (5.4.9), to obtain the least-squares approximations in (5.4.15). This section follows Koller and Parr (1999) and shows that (5.4.9) can be computed more efficiently when the basis functions h have restricted induced scopes. I.e., we show how Step 3 of Algorithm 5.2 can be performed more efficiently. Moreover, we extend the work of Koller and Parr, by showing that this computation is particularly efficient when using induced indicator basis functions.

C.2.1 Rewriting Regression Using Inner Products

We start be showing how (5.4.9) can be rewritten as a number of inner products. In particular let

$$A = (H^T H) = \begin{pmatrix} - & h_1^T & - \\ & \vdots & \\ - & h_{\hat{N}}^T & - \end{pmatrix} \begin{pmatrix} | & & | \\ h_1 & \dots & h_{\hat{N}} \\ | & & | \end{pmatrix} \tag{C.2.1}$$

be an $\hat{N} \times \hat{N}$ matrix, which typically is small enough to represent. This matrix can be constructed efficiently, as each entry can be computed efficiently, because

$$a_{ij} = h_i^T h_j = h_i \cdot h_j \tag{C.2.2}$$

and the inner product '$\cdot$' of two vectors representing functions with a restricted induced scope can be computed efficiently as discussed below.

Also let

$$w' = \begin{pmatrix} | \\ w' \\ | \end{pmatrix} = \begin{pmatrix} - & h_1^T & - \\ & \vdots & \\ - & h_{\hat{N}}^T & - \end{pmatrix} \begin{pmatrix} | \\ Q \\ | \end{pmatrix} = H^T Q. \tag{C.2.3}$$

$$\begin{aligned}
\begin{pmatrix} a \\ b \end{pmatrix} \cdot \begin{pmatrix} c_1 + c_2 \\ d_1 + d_2 \end{pmatrix} &= a(c_1 + c_2) + b(d_1 + d_2) && \text{(C.2.5)} \\
&= ac_1 + ac_2 + bd_1 + bd_2 && \text{(C.2.6)} \\
&= (ac_1 + bd_1) + (ac_2 + bd_2) && \text{(C.2.7)} \\
&= \begin{pmatrix} a \\ b \end{pmatrix} \cdot \begin{pmatrix} c_1 \\ d_1 \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix} \cdot \begin{pmatrix} c_2 \\ d_2 \end{pmatrix} && \text{(C.2.8)}
\end{aligned}$$

Figure C.1: Linearity of inner product.

Even though representing H and Q explicitly requires exponential space, storing w' only requires $\hat{N}$ entries. Moreover, provided that Q is a factored linear value function with restricted scope, each entry w'_i can be computed from a number of inner products:

$$w'_i = h_i^T Q = h_i \cdot Q = h_i \cdot \left[\sum_e Q^e\right] = \sum_e \left[h_i \cdot Q^e\right] \tag{C.2.4}$$

because of the linearity of the inner product (see Figure C.1). Therefore, if all components Q^e have a restricted, small scope, w'_i can be computed efficiently.

Note that when Q is represented flat (as in the naive regression approach outlined in Section 5.4.2), computation of w'_i requires performing the summation over all (state, joint-action pairs):

$$w'_i = \sum_{s,\mathbf{a}} h_i(s,\mathbf{a}) Q(s,\mathbf{a}) \tag{C.2.9}$$

which becomes intractable for large number of states and agents. As such both the computation of the entire $Q(s,\mathbf{a})$-function *and* the regression step form a bottleneck for naive regression as outlined in Algorithm 5.1.

Finally we can compute a new parameter vector w that gives the least squares error approximation as identified in (5.4.9) by putting together the pieces created in this section

$$w = A^{-1} w'. \tag{C.2.10}$$

C.2.2 Efficient Inner Products

In this section we explain how the inner products of restricted induced scope functions can be computed efficiently, and how induced indicator functions gain even more efficiency. To ease the notation, we assume that all functions are functions of states s or subsets of state factors $\mathcal{X}$ only. However, since no special assumptions are made, the analysis immediately extends to functions defined over SAPs z and subsets of state factors and actions $\mathcal{X} \cup \mathcal{A}$.

For arbitrary functions f,g defined over the set of states $\mathcal{S}$ we have that

$$f \cdot g = \sum_{s \in \mathcal{S}} f(s) g(s) = f(s_1) g(s_1) + \cdots + f(s_{|\mathcal{S}|}) g(s_{|\mathcal{S}|}) \tag{C.2.11}$$

	State index							
	1	2	3	4	5	6	7	8
x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	0	1	0	1	0	1

Table C.1: State space of three binary state factors.

which requires summing over all $s \in \mathcal{S}$. If f,g are arbitrary, this cannot be improved upon. However, the inner product of two functions of restricted induced scope can be computed more efficiently.

Example C.1. Let us consider a state space $\mathcal{S}$ spanned by a set of three binary state factors $\mathcal{X} = \{x_1,x_2,x_3\}$ such that we can summarize the state space as shown in Table C.1, i.e., the state index is the number represented by the state factors in binary plus one.
Now consider functions f,g induced by f' defined over x_1 and g' defined over x_2 as follows

$$f'(x_1 = 0) = 4, \quad f'(x_1 = 1) = 2, \tag{C.2.12}$$

$$g'(x_2 = 0) = 3, \quad g'(x_2 = 1) = 1. \tag{C.2.13}$$

This leads to the inner product of $f = (4,4,4,4,2,2,2,2)^T$ and $g = (3,3,1,1,3,3,1,1)^T$

$$\begin{aligned} f \cdot g &= 4 \cdot 3 + 4 \cdot 3 + 4 \cdot 1 + 4 \cdot 1 + 2 \cdot 3 + 2 \cdot 3 + 2 \cdot 1 + 2 \cdot 1 && \text{(C.2.14)} \\ &= 2\,(4 \cdot 3 + 4 \cdot 1 + 2 \cdot 3 + 2 \cdot 1) && \text{(C.2.15)} \\ &= 2\,(12 + 4 + 6 + 2) = 48. && \text{(C.2.16)} \end{aligned}$$

Clearly computation as in (C.2.14) contains redundancy that is exploited to allow for more efficient computation in (C.2.15).

The intuition of this example is formalized in the following proposition.

Proposition C.1 (Inner product of functions with restricted induced scope)**.** *Let us write $\mathbb{S}(f)$,$\mathbb{S}(g)$ for the induced scopes of f,g. Let $\mathcal{U} = \mathbb{S}(f) \cup \mathbb{S}(g)$ be the joint scope of f,g, and $\bar{\mathcal{U}} = \mathcal{X} \backslash \mathcal{U}$ the complement. The inner product of f,g is defined as*

$$f \cdot g = |\mathcal{X}_{\bar{\mathcal{U}}}| \sum_{\mathbf{x}_{\mathcal{U}} \in \mathcal{X}_{\mathcal{U}}} f(\mathbf{x}_{\mathcal{U}}) g(\mathbf{x}_{\mathcal{U}}). \tag{C.2.17}$$

Computation of this inner product takes time linear in $|\mathcal{X}_{\mathcal{U}}|$, the number of local states defined for the union of the induced scopes.

In the example above, we have that $\mathcal{U} = \{x_1,x_2\}$ and $\bar{\mathcal{U}} = \{x_3\}$. It is easy to see that (C.2.15) can be rewritten as (C.2.17). The above analysis naturally extends to functions defined over state-action pairs. For instance consider the previous example, but interpret all indices as (local) (state, action)-pairs. I.e., we now have a (state, joint action)-space $\mathcal{Z}$ (consisting of $N = 8$ SAPs z) that is spanned by three binary LSAP variables $\{\hat{z}_1,\hat{z}_2,\hat{z}_3\}$. (For instance if there is only one agent and with one action, this formulation reduces exactly to the previous example).

C.2.2.1 Inner Products for Induced Indicator Functions

When using basis functions that are induced indicator functions, as in (C.1.3), matters simplify even further, as illustrated by the next example.

Example C.2. Let us consider two induced indicator basis function h_i,h_j with induced scopes $\mathbb{S}(i) = \{x_1\}$,$\mathbb{S}(j) = \{x_2\}$, defined through

$$\hat{h}_i(x_1 = 0) = 1, \quad \hat{h}_i(x_1 = 1) = 0, \tag{C.2.18}$$

$$\hat{h}_j(x_2 = 0) = 0, \quad \hat{h}_j(x_2 = 1) = 1. \tag{C.2.19}$$

Let the joint scope be $\mathcal{U} = \mathbb{S}(i) \cup \mathbb{S}(j) = \{x_1,x_2\}$ and $\bar{\mathcal{U}} = \mathcal{X} \setminus \mathcal{U} = \{x_3\}$. Now because we use indicator functions, we know that

$$f(\mathbf{x}_\mathcal{U})g(\mathbf{x}_\mathcal{U}) = \begin{cases} 1 & \mathbf{x}_\mathcal{U} = \langle x_1 = 0,x_2 = 1\rangle \\ 0 & \text{otherwise.} \end{cases} \tag{C.2.20}$$

Therefore we know that the second part of (C.2.17) sums to 1:

$$\sum_{\mathbf{x}_\mathcal{U} \in \mathcal{X}_\mathcal{U}} f(\mathbf{x}_\mathcal{U})g(\mathbf{x}_\mathcal{U}) = 1, \tag{C.2.21}$$

and therefore that the inner product reduces to

$$f \cdot g = |\mathcal{X}_{\bar{\mathcal{U}}}| \,. \tag{C.2.22}$$

This example demonstrates that computation of the inner product of indicator functions is typically very cheap. However, there are some technical details we need to take care of. In particular, in the above we assume two functions that were specified over disjoint scopes $\mathbb{S}(i) \cap \mathbb{S}(j) = \emptyset$. In the following, we generalize the example.

- Let h_i,h_j be induced indicator basis functions with arbitrary induced scopes $\mathbb{S}(i)$,$\mathbb{S}(j) \subseteq \mathcal{X}$, induced by local indicator functions $\hat{h}_i$ and $\hat{h}_j$.
- We write $\mathbf{x}^i_{\mathbb{S}(i)}$ for the local state indicated by $\hat{h}_i$ and $\mathbf{x}^j_{\mathbb{S}(j)}$ for that indicated by $\hat{h}_j$.
- $\mathcal{U} = \mathbb{S}(i) \cup \mathbb{S}(j)$ is the union of scopes and $\bar{\mathcal{U}} = \mathcal{X} \setminus \mathcal{U}$ is the complement of the union.
- $\mathcal{I} = \mathbb{S}(i) \cap \mathbb{S}(j)$ is the intersection of scopes.
- $\mathbf{x}^i_\mathcal{I}$ is the locate state indicated by $\hat{h}_i$ restricted to contain only factors from $\mathcal{I}$ and similar for $\mathbf{x}^j_\mathcal{I}$.

Proposition C.2 (Inner product of induced indicator functions)**.** *The inner product of two induced indicator functions h_i,h_j is defined as*

$$h_i \cdot h_j = \begin{cases} |\mathcal{X}_{\bar{\mathcal{U}}}| & \textit{if } \mathbf{x}^i_\mathcal{I} = \mathbf{x}^j_\mathcal{I} \\ 0 & \textit{otherwise.} \end{cases} \tag{C.2.23}$$

As such the cost of computing this inner product is constant in $|\mathcal{X}_{\mathcal{U}}|$ and linear in $\bar{\mathcal{U}}$ (i.e., logarithmic in $|\mathcal{X}_{\bar{\mathcal{U}}}|$).

Proof. The second part of (C.2.17) is either 1 or 0, yielding a value of respectively $|\mathcal{X}_{\bar{\mathcal{U}}}|$ or 0. Now, when $\mathbf{x}_{\mathcal{I}}^i = \mathbf{x}_{\mathcal{I}}^j$ we know that the indicated states agree upon their shared variables, and therefore that there is a state $\mathbf{x}_{\mathcal{U}} \in \mathcal{X}_{\mathcal{U}}$ such that $h_i(\mathbf{x}_{\mathcal{U}})h_j(\mathbf{x}_{\mathcal{U}}) = 1$. However, when $\mathbf{x}_{\mathcal{I}}^i \neq \mathbf{x}_{\mathcal{I}}^j$ the indicated local states are inconsistent with each other and we can never find a $\mathbf{x}_{\mathcal{U}}$ that yields $h_i(\mathbf{x}_{\mathcal{U}})h_j(\mathbf{x}_{\mathcal{U}}) = 1$. Therefore, the second part of (C.2.17) is 0 in this case. □

C.2.2.2 Product of Induced Indicator and Regular Functions

In the previous section we discussed how the inner product of two induced basis functions, needed for the computation of (C.2.2), can be computed efficiently. For the computation of (C.2.4), we need the inner product of induced indicator basis functions with the factored Q-value function, which is a regular function of restricted induced scope.

Example C.3. We assume the same 3-factor binary state space as before. Assume we have an induced indicator basis function h, induced by $\hat{h}$ with scope $\{x_1,x_2\}$ that has $\langle 0,1\rangle$ as the indicated element, i.e.

$$\hat{h}(x_1,x_2) = \begin{cases} 1, & x_1 = 0, x_2 = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{C.2.24}$$

The other function g is a function of induced scope $\{x_2,x_3\}$, induced by:

$$g'(0,0) = 5,\ g'(0,1) = 36,\ g'(1,0) = 8,\ g'(1,1) = -6 \tag{C.2.25}$$

This leads to the inner product of $h = (0,0,1,1,0,0,0,0)^T$ and $g = (5,36,8,-6,5,36,8,-6)^T$

$$\begin{aligned} f \cdot g &= 0 \cdot 5 + 0 \cdot 36 + 1 \cdot 8 + 1 \cdot -6 + 0 \cdot 5 + 0 \cdot 36 + 0 \cdot 8 + 0 \cdot -6 \\ &= 8 - 6 = 2 \end{aligned} \tag{C.2.26}$$

Analyzing the above situation, we see that the general formula (C.2.17) still performs redundant computations, as it sums over many entries that are 0. Here, we split the union of state variables $\mathcal{U} = \mathbb{S}(h) \cup \mathbb{S}(g)$ into the the sets of variables in the intersection and those contained only in the induced scope of h or g. I.e., we define $\mathcal{I} = \mathbb{S}(h) \cap \mathbb{S}(g)$

$$\mathcal{U} = \mathcal{I} \cup \mathcal{H} \cup \mathcal{G},$$

where $\mathcal{H} = \mathbb{S}(h) \setminus \mathcal{I}$ and $\mathcal{G} = \mathbb{S}(g) \setminus \mathcal{I}$ are the 'private' factors of h and g. Now we can write

$$\begin{aligned} h \cdot g &= |\mathcal{X}_{\bar{\mathcal{U}}}| \sum_{\mathbf{x}_{\mathcal{I}} \in \mathcal{X}_{\mathcal{I}}} \sum_{\mathbf{x}_{\mathcal{H}} \in \mathcal{X}_{\mathcal{H}}} \sum_{\mathbf{x}_{\mathcal{G}} \in \mathcal{X}_{\mathcal{G}}} h(\mathbf{x}_{\mathcal{U}}) g(\mathbf{x}_{\mathcal{U}}). \\ &= |\mathcal{X}_{\bar{\mathcal{U}}}| \sum_{\mathbf{x}_{\mathcal{I}} \in \mathcal{X}_{\mathcal{I}}} \sum_{\mathbf{x}_{\mathcal{H}} \in \mathcal{X}_{\mathcal{H}}} h(\langle \mathbf{x}_{\mathcal{I}}, \mathbf{x}_{\mathcal{H}} \rangle) \sum_{\mathbf{x}_{\mathcal{G}} \in \mathcal{X}_{\mathcal{G}}} g(\langle \mathbf{x}_{\mathcal{I}}, \mathbf{x}_{\mathcal{G}} \rangle). \end{aligned}$$

Note that $\mathcal{I} \cup \mathcal{H} = \mathbb{S}(h)$, therefore the summation $\sum_{\mathbf{x}_\mathcal{I} \in \mathcal{X}_\mathcal{I}} \sum_{\mathbf{x}_\mathcal{H} \in \mathcal{X}_\mathcal{H}} h(\langle \mathbf{x}_\mathcal{I}, \mathbf{x}_\mathcal{H} \rangle)$ actually is just a summation over the scope of indicator function h; in this summation there is only one non-zero entry (for the indicated element). As such we can further reduce the inner product.

Proposition C.3 (Inner product of induced function and induced indicator function)**.** *The inner product of an induced indicator basis function h and induced function g is given by*

$$h \cdot g = |\mathcal{X}_{\overline{\mathcal{U}}}| \sum_{\mathbf{x}_\mathcal{G} \in \mathcal{X}_\mathcal{G}} g(\langle \mathbf{x}_\mathcal{I}^{ind}, \mathbf{x}_\mathcal{G} \rangle)$$

where $\mathbf{x}_\mathcal{I}^{ind}$ is the restriction (to $\mathcal{I}$) of the element indicated by h. Computation of this inner product is linear in $|\mathcal{X}_\mathcal{G}|$ the number of instantiations of the variables exclusive to the induced scope of g.

C.2.3 Translation to Indicator Functions for SAPs

In Section C.2.2.1 we make no special assumption on the set of variables comprising $\mathcal{X}$. As such all the above transfers to indicator functions specified over subsets of states and action variables. I.e., we can replace $\mathcal{X}$ by any set of variables $\mathcal{Z}$ and therefore also by $\mathcal{Z} = \mathcal{X} \cup \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$.

Although this transformation is trivial, it is interesting to make the following observations in the context of regression to basis functions as described in Section 5.4.2.2: because we have k local Q-functions, we have k groups of basis functions. Each such group j, $1 \leq j \leq k$ represents $Q^{j,t}$. The basis function within such a group all have the same scope, namely the scope $\mathbb{S}(j,t)$. Therefore inner products of basis functions that are part of the same Q-function component $Q^{j,t}$ have inner product 0.

Appendix D

Proofs

This appendix contains several various proofs that would have interrupted the flow of the text too much. For convenience, the propositions, theorems, lemmas, etc. themselves have been repeated.

D.1 Proofs of Chapter 2

Proof of Theorem 2.1. For a BG with identical payoffs, i.e., $\forall_{i,j}\forall_{\boldsymbol{\theta}}\forall_{\mathbf{a}}\ u_i(\boldsymbol{\theta},\mathbf{a}) = u_j(\boldsymbol{\theta},\mathbf{a})$, the solution is given by:

$$\boldsymbol{\beta}^* = \arg\max_{\boldsymbol{\beta}} \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \Pr(\boldsymbol{\theta}) u(\boldsymbol{\theta},\boldsymbol{\beta}(\boldsymbol{\theta})), \tag{D.1.1}$$

where $\boldsymbol{\beta}(\boldsymbol{\theta}) = \langle \beta_1(\theta_1),...,\beta_n(\theta_n)\rangle$ is the joint action specified by $\boldsymbol{\beta}$ for joint type $\boldsymbol{\theta}$. This solution constitutes a Pareto optimal Nash equilibrium.

Proof. The proof consists of two parts: the first shows that $\boldsymbol{\beta}^*$ is a Nash equilibrium, the second shows it is Pareto optimal.

Nash Equilibrium Proof. It is clear that $\boldsymbol{\beta}^*$ satisfying (D.1.1) is a Nash equilibrium by rewriting from the perspective of an arbitrary agent i as follows:

$$\begin{aligned}
\beta_i^* &= \arg\max_{\beta_i}\left[\max_{\beta_{\neq i}} \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \Pr(\boldsymbol{\theta}) u(\boldsymbol{\theta},\boldsymbol{\beta}(\boldsymbol{\theta}))\right], \\
&= \arg\max_{\beta_i}\left[\max_{\beta_{\neq i}} \sum_{\theta_i}\sum_{\theta_{\neq i}} \Pr(\theta_{\neq i}|\theta_i) \underbrace{\left[\sum_{\theta_{\neq i}} \Pr(\langle\theta_i,\theta_{\neq i}\rangle)\right]}_{\Pr(\theta_i)} u(\boldsymbol{\theta},\boldsymbol{\beta}(\boldsymbol{\theta}))\right], \\
&= \arg\max_{\beta_i}\left[\max_{\beta_{\neq i}} \sum_{\theta_i} \Pr(\theta_i) \sum_{\theta_{\neq i}} \Pr(\theta_{\neq i}|\theta_i) u(\boldsymbol{\theta},\boldsymbol{\beta}(\boldsymbol{\theta}))\right], \\
&= \arg\max_{\beta_i} \sum_{\theta_i} \Pr(\theta_i) \sum_{\theta_{\neq i}} \Pr(\theta_{\neq i}|\theta_i) u(\langle\theta_i,\theta_{\neq i}\rangle, \langle\beta_i(\theta_i),\beta^*_{\neq i}(\theta_{\neq i})\rangle),
\end{aligned}$$

which means that β_i^* is a best response for $\beta_{\neq i}^*$. Since no special assumptions were made on i, it follows that $\boldsymbol{\beta}^*$ is a Nash equilibrium.

Pareto Optimality Proof. Let us write $V_{\theta_i}(a_i,\beta_{\neq i})$ for the payoff agent i expects for θ_i when performing a_i when the other agents use policy profile $\beta_{\neq i}$. We have that

$$V_{\theta_i}(a_i,\beta_{\neq i}) = \sum_{\theta_{\neq i}} \Pr(\theta_{\neq i}|\theta_i) u(\langle \theta_i,\theta_{\neq i}\rangle, \langle a_i,\beta_{\neq i}(\theta_{\neq i})\rangle).$$

Now, a joint policy $\boldsymbol{\beta}^*$ satisfying (D.1.1) is not Pareto optimal if and only if there is another Nash equilibrium $\boldsymbol{\beta}'$ that attains at least the same payoff for all agents i and for all types θ_i and strictly more for at least one agent and type. Formally $\boldsymbol{\beta}^*$ is not Pareto optimal when $\exists_{\boldsymbol{\beta}'}$ such that, $\forall_i \forall_{\theta_i}$

$$V_{\theta_i}(\beta_i^*(\theta_i),\beta_{\neq i}^*) \leq V_{\theta_i}(\beta_i{}'(\theta_i),\beta_{\neq i}{}') \ \wedge \ \exists_i \exists_{\theta_i} V_{\theta_i}(\beta_i^*(\theta_i),\beta_{\neq i}^*) < V_{\theta_i}(\beta_i{}'(\theta_i),\beta_{\neq i}{}'). \tag{D.1.2}$$

We prove that no such $\boldsymbol{\beta}'$ can exist by contradiction. Suppose that $\boldsymbol{\beta}' = \langle \beta_i{}', \beta'_{\neq i}\rangle$ is a NE such that (D.1.2) holds (and thus $\boldsymbol{\beta}^*$ is not Pareto optimal). Because $\boldsymbol{\beta}^*$ satisfies (D.1.1) we know that:

$$\sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \Pr(\boldsymbol{\theta}) u(\boldsymbol{\theta},\boldsymbol{\beta}^*(\boldsymbol{\theta})) \geq \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \Pr(\boldsymbol{\theta}) u(\boldsymbol{\theta},\boldsymbol{\beta}'(\boldsymbol{\theta})), \tag{D.1.3}$$

and therefore, for all agents i

$$\begin{aligned}\Pr(\theta_{i,1}) V_{\theta_{i,1}}(\beta_i^*(\theta_{i,1}),\beta_{\neq i}^*) + ... + \Pr(\theta_{i,|\Theta_i|}) V_{\theta_{i,|\Theta_i|}}(\beta_i^*(\theta_{i,|\Theta_i|}),\beta_{\neq i}^*) \geq \\ \Pr(\theta_{i,1}) V_{\theta_{i,1}}(\beta_i'(\theta_{i,1}),\beta'_{\neq i}) + ... + \Pr(\theta_{i,|\Theta_i|}) V_{\theta_{i,|\Theta_i|}}(\beta_i'(\theta_{i,|\Theta_i|}),\beta'_{\neq i})\end{aligned}$$

holds. However, by assumption that $\boldsymbol{\beta}'$ satisfies (D.1.2) we get that

$$\exists_j \quad V_{\theta_{i,j}}(\beta_i^*(\theta_{i,j}),\beta_{\neq i}^*) < V_{\theta_{i,j}}(\beta_i'(\theta_{i,j}),\beta'_{\neq i}).$$

Therefore it must be that

$$\sum_{k\neq j} \Pr(\theta_{i,k}) V_{\theta_{i,k}}(\beta_i^*(\theta_{i,k}),\beta_{\neq i}^*) > \sum_{k\neq j} \Pr(\theta_{i,k}) V_{\theta_{i,k}}(\beta_i'(\theta_{i,k}),\beta'_{\neq i}),$$

and thus that

$$\exists_k \quad V_{\theta_{i,k}}(\beta_i^*(\theta_{i,k}),\beta_{\neq i}^*) > V_{\theta_{i,k}}(\beta_i'(\theta_{i,k}),\beta'_{\neq i}),$$

contradicting the assumption that $\boldsymbol{\beta}'$ satisfies (D.1.2). □

D.2 Proofs of Chapter 3

Proof of Proposition 3.1 (Value of an optimal joint policy)**.** The expected cumulative reward over stages $t,\ldots,h-1$ induced by $\boldsymbol{\pi}^*$, an optimal pure joint policy for a Dec-POMDP, is given by:

$$V^t(\boldsymbol{\pi}^*) = \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\pi}^*}} \Pr(\vec{\boldsymbol{\theta}}^t|\boldsymbol{b}^0) Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^t,\boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^t)), \tag{D.2.1}$$

where $\vec{\boldsymbol{\theta}}^t = \langle \vec{\mathbf{o}}^t, \vec{\mathbf{a}}^t \rangle$, where $\boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^t) = \boldsymbol{\pi}^*(\vec{\mathbf{o}}^t)$ denotes the joint action $\boldsymbol{\pi}^*$ specifies for $\vec{\mathbf{o}}^t$, and where

$$Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^t, \mathbf{a}) = R(\vec{\boldsymbol{\theta}}^t, \mathbf{a}) + \sum_{\mathbf{o}^{t+1} \in \mathcal{O}} \Pr(\mathbf{o}^{t+1} | \vec{\boldsymbol{\theta}}^t, \mathbf{a}) Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^{t+1}, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{t+1})) \tag{D.2.2}$$

is the Q-value function for $\boldsymbol{\pi}^*$, which gives the expected cumulative future reward when taking joint action $\mathbf{a}$ at $\vec{\boldsymbol{\theta}}^t$ given that $\boldsymbol{\pi}^*$ is followed hereafter.

Proof. By filling out (2.5.5) for an optimal pure joint policy $\boldsymbol{\pi}^*$, we obtain its expected cumulative reward as the summation of $E\left[R(s^t, \mathbf{a}^t) \middle| \boldsymbol{\pi}^*\right]$ the expected rewards it yields for each time step:

$$V(\boldsymbol{\pi}^*) = \sum_{t=0}^{h-1} E\left[R(s^t, \mathbf{a}^t) \middle| \boldsymbol{\pi}^*\right] = \sum_{t=0}^{h-1} \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t} \Pr(\vec{\boldsymbol{\theta}}^t | \boldsymbol{\pi}^*, \boldsymbol{b}^0) R(\vec{\boldsymbol{\theta}}^t, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^t)). \tag{D.2.3}$$

In this equation, $\Pr(\vec{\boldsymbol{\theta}}^t | \boldsymbol{\pi}^*, \boldsymbol{b}^0)$ is given by (3.1.3). As a result, the influence of $\boldsymbol{\pi}^*$ on $\Pr(\vec{\boldsymbol{\theta}}^t | \boldsymbol{\pi}^*, \boldsymbol{b}^0)$ is only through C. I.e., $\boldsymbol{\pi}^*$ is only used to 'filter out' inconsistent histories. Therefore we can write:

$$E\left[R(s^t, \mathbf{a}^t) \middle| \boldsymbol{\pi}^*\right] = \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\pi}^*}} \Pr(\vec{\boldsymbol{\theta}}^t | \boldsymbol{b}^0) R(\vec{\boldsymbol{\theta}}^t, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^t)), \tag{D.2.4}$$

where $\Pr(\vec{\boldsymbol{\theta}}^t | \boldsymbol{b}^0)$ is given by directly taking the marginal of (3.1.4). Now, let us define the value starting from time step t:

$$\begin{aligned} V^t(\boldsymbol{\pi}^*) &= E\left[R(s^t, \mathbf{a}^t) \middle| \boldsymbol{\pi}^*\right] + V^{t+1}(\boldsymbol{\pi}^*) \\ &= \sum_{\vec{\boldsymbol{\theta}}^t \in \vec{\boldsymbol{\Theta}}^t_{\boldsymbol{\pi}^*}} \Pr(\vec{\boldsymbol{\theta}}^t | \boldsymbol{b}^0) R(\vec{\boldsymbol{\theta}}^t, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^t)) + V^{t+1}(\boldsymbol{\pi}^*). \end{aligned} \tag{D.2.5}$$

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

$$V^{h-1}(\boldsymbol{\pi}^*) = \sum_{\vec{\boldsymbol{\theta}}^{h-1} \in \vec{\boldsymbol{\Theta}}^{h-1}_{\boldsymbol{\pi}^*}} \Pr(\vec{\boldsymbol{\theta}}^{h-1} | \boldsymbol{b}^0) \underbrace{R(\vec{\boldsymbol{\theta}}^{h-1}, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{h-1}))}_{Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^{h-1}, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{h-1}))}. \tag{D.2.6}$$

For time step $h-2$ this becomes:

$$\begin{aligned} V^{h-2}(\boldsymbol{\pi}^*) \equiv E\left[R(s^{h-2}, \mathbf{a}^{h-2}) \middle| \boldsymbol{\pi}^*\right] + V^{h-1}(\boldsymbol{\pi}^*) = \sum_{\vec{\boldsymbol{\theta}}^{h-2} \in \vec{\boldsymbol{\Theta}}^{h-2}_{\boldsymbol{\pi}^*}} \Pr(\vec{\boldsymbol{\theta}}^{h-2} | \boldsymbol{b}^0) \\ R(\vec{\boldsymbol{\theta}}^{h-2}, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{h-2})) + \sum_{\vec{\boldsymbol{\theta}}^{h-1} \in \vec{\boldsymbol{\Theta}}^{h-1}_{\boldsymbol{\pi}^*}} \Pr(\vec{\boldsymbol{\theta}}^{h-1} | \boldsymbol{b}^0) Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^{h-1}, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{h-1})). \end{aligned} \tag{D.2.7}$$

Because $\Pr(\vec{\boldsymbol{\theta}}^{h-1}) = \Pr(\vec{\boldsymbol{\theta}}^{h-2}) \Pr(\mathbf{o}^{h-1} | \vec{\boldsymbol{\theta}}^{h-2}, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{h-2}))$, (D.2.7) can be rewritten to:

$$V^{h-2}(\boldsymbol{\pi}^*) = \sum_{\vec{\boldsymbol{\theta}}^{h-2} \in \vec{\boldsymbol{\Theta}}^{h-2}_{\boldsymbol{\pi}^*}} \Pr(\vec{\boldsymbol{\theta}}^{h-2} | \boldsymbol{b}^0) Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^{h-2}, \boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{h-2})), \tag{D.2.8}$$

with

$$Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^{h-2},\boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{h-2})) = R(\vec{\boldsymbol{\theta}}^{h-2},\boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{h-2}))+ \sum_{\mathbf{o}^{h-1}} \Pr(\mathbf{o}^{h-1}|\vec{\boldsymbol{\theta}}^{h-2},\boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{h-2}))Q_{\boldsymbol{\pi}^*}(\vec{\boldsymbol{\theta}}^{h-1},\boldsymbol{\pi}^*(\vec{\boldsymbol{\theta}}^{h-1})). \quad \text{(D.2.9)}$$

Reasoning in the same way we see that (D.2.1) and (D.2.2) constitute a generic expression for the expected cumulative future reward starting from time step t. □

Proof of Lemma 3.2 (Value of k-steps delayed communication). The optimal value function for a finite horizon Dec-POMDP with k-steps delayed, cost and noise free, communication is given by:

$$V_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}) = \max_{\boldsymbol{\beta}^t} Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t). \quad \text{(D.2.10)}$$

$$Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t) = \sum_{\vec{\boldsymbol{\theta}}^t_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k})Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\vec{\boldsymbol{\theta}}^t_{|k|},\boldsymbol{\beta}^t) \quad \text{(D.2.11)}$$

$$Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\vec{\boldsymbol{\theta}}^t_{|k|},\boldsymbol{\beta}^t) = R(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|}))+ \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|}))Q_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1},\vec{\boldsymbol{\theta}}^{t+1}_{|k|},\boldsymbol{\beta}^{t+1,*}) \quad \text{(D.2.12)}$$

where $\boldsymbol{b}^t$ results from $\boldsymbol{b}^{t-k},\vec{\boldsymbol{\theta}}^t_{|k|}$, and

$$\boldsymbol{\beta}^{t+1,*} = \arg\max_{\boldsymbol{\beta}^{t+1}} \sum_{\vec{\boldsymbol{\theta}}^{t+1}} \Pr(\vec{\boldsymbol{\theta}}^{t+1}_{|k|}|\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1})Q_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1},\vec{\boldsymbol{\theta}}^{t+1}_{|k|},\boldsymbol{\beta}^{t+1}) \quad \text{(D.2.13)}$$

Proof. We will show that

$$Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t) = E\Big[R(s^t,\mathbf{a}^t)|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t\Big] + E\left[V_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1}) \;\middle|\; \boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t\right] \quad \text{(D.2.14)}$$

and that the other equations are consistent with this definition. This means that, for the last stage (D.2.10) maximizes the expected reward and therefore is optimal, optimality for other stages follows immediately by induction.

$$Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t) = \sum_{\vec{\boldsymbol{\theta}}^t_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k})R(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|})) + \Big[\sum_{\mathbf{o}^{t-k+1}} \Pr(\mathbf{o}^{t-k+1}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k})V_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1})\Big] \quad \text{(D.2.15)}$$

where $\boldsymbol{q}^{t-k+1} = \langle \boldsymbol{q}^{t-k} \circ \boldsymbol{\beta}^t \rangle \Big\downarrow_{\mathbf{o}^{t-k+1}}$, and $\boldsymbol{b}^{t-k+1}$ results from $\boldsymbol{b}^{t-k}$ via the joint action $\mathbf{a}^{t-k}$ (specified by $\boldsymbol{q}^{t-k}$) and $\mathbf{o}^{t-k+1}$.

By introducing $\boldsymbol{\beta}^{t+1,*}$ as in (D.2.13), $V_k^{t+1,*}$ can be replaced with

$$\begin{aligned} &V_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1}) \\ &\quad = \max_{\boldsymbol{\beta}^{t+1}} \sum_{\vec{\boldsymbol{\theta}}^t_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1}) Q_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1},\vec{\boldsymbol{\theta}}^t_{|k|},\boldsymbol{\beta}^{t+1}) \qquad \text{(D.2.16)} \\ &\quad = \sum_{\vec{\boldsymbol{\theta}}^t_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1}) Q_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1},\vec{\boldsymbol{\theta}}^t_{|k|},\boldsymbol{\beta}^{t+1,*}) \qquad \text{(D.2.17)} \end{aligned}$$

The result of this replacement is

$$\begin{aligned} Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t) &= \sum_{\vec{\boldsymbol{\theta}}^t_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}) R(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|})) + \Big[\sum_{\mathbf{o}^{t-k+1}} \Pr(\mathbf{o}^{t-k+1}| \\ &\quad \boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}) \sum_{\vec{\boldsymbol{\theta}}^{t+1}_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^{t+1}_{|k|}|\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1}) Q_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1},\vec{\boldsymbol{\theta}}^{t+1}_{|k|},\boldsymbol{\beta}^{t+1,*}) \Big] \end{aligned}$$

$$\begin{aligned} Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t) &= \sum_{\vec{\boldsymbol{\theta}}^t_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}) R(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t)) + \\ &\quad \Big[\sum_{\vec{\boldsymbol{\theta}}^{t+1}_{|k+1|}} \Pr(\vec{\boldsymbol{\theta}}^{t+1}_{|k+1|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t) Q_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1},\vec{\boldsymbol{\theta}}^{t+1}_{|k|},\boldsymbol{\beta}^{t+1,*}) \Big] \end{aligned}$$

$$\begin{aligned} Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t) &= \sum_{\vec{\boldsymbol{\theta}}^t_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}) R(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|})) + \\ &\Big[\sum_{\vec{\boldsymbol{\theta}}^t_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}) \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|})) Q_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1},\vec{\boldsymbol{\theta}}^{t+1}_{|k|},\boldsymbol{\beta}^{t+1,*}) \Big] \end{aligned}$$

$$\begin{aligned} Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\boldsymbol{\beta}^t) &= \sum_{\vec{\boldsymbol{\theta}}^t_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^t_{|k|}|\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}) \Big[R(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|})) + \\ &\quad \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|})) Q_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1},\vec{\boldsymbol{\theta}}^{t+1}_{|k|},\boldsymbol{\beta}^{t+1,*}) \Big] \end{aligned}$$

and thus, finally,

$$\begin{aligned} Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k},\vec{\boldsymbol{\theta}}^t_{|k|},\boldsymbol{\beta}^t) &= R(\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|})) + \\ &\quad \sum_{\mathbf{o}^{t+1}} \Pr(\mathbf{o}^{t+1}|\boldsymbol{b}^t,\boldsymbol{\beta}^t(\vec{\boldsymbol{\theta}}^t_{|k|})) Q_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1},\vec{\boldsymbol{\theta}}^{t+1}_{|k|},\boldsymbol{\beta}^{t+1,*}) \qquad \text{(D.2.18)} \end{aligned}$$

concluding the proof. □

Proof of Theorem 3.3 (Shorter communication delays cannot decrease the expected value). The expected value over stages $t,\ldots,h-1$ given a joint belief $\boldsymbol{b}^{t-k-1}$ and joint policy $\boldsymbol{q}^{t-k-1}_{|k+1|}$ followed during stages $t-k-1,\ldots,t-1$ is no less under k-steps communication delay, than under $(k+1)$-steps delay. That is

$$\forall_t \forall_{\boldsymbol{b}^{t-k-1}} \forall_{\boldsymbol{q}^{t-k-1}_{|k+1|}} \quad E\big[V_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}_{|k|}) \mid \boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|}\big] \geq V_{k+1}^{t,*}(\boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|}). \tag{D.2.19}$$

Proof. The proof is by induction. The base case is that (D.2.19) holds for stage $t=h-1$. The induction hypothesis is that (D.2.19) holds for some stage $t+1$

$$E\big[V_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1}_{|k|}) \mid \boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}_{|k+1|}\big] \geq V_{k+1}^{t+1,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}_{|k+1|}). \tag{D.2.20}$$

We now need to show that (D.2.19) holds given (D.2.20). Assuming an arbitrary $t<h-1$, $\vec{\boldsymbol{\theta}}^{t-k-1}$ and $\boldsymbol{q}^{t-k-1}_{|k+1|}$, the left side of (D.2.19) can be rewritten as follows

$$E\big[V_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}_{|k|}) \mid \boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|}\big] \tag{D.2.21}$$

$$= E\Big[\max_{\boldsymbol{\beta}^t_{|k|}} Q_k^{t,*}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}_{|k|},\boldsymbol{\beta}^t_{|k|}) \;\Big|\; \boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|}\Big] \tag{D.2.22}$$

$$= E\Big[\max_{\boldsymbol{\beta}^t_{|k|}}\Big[E\big[R(s^t,\mathbf{a}^t) \mid \boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}_{|k|},\boldsymbol{\beta}^t_{|k|}\big] + E\big[V_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1}_{|k|}) \mid \boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}_{|k|},\boldsymbol{\beta}^t_{|k|}\big]\Big] \;\Big|\; \boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|}\Big] \tag{D.2.23}$$

Note that $\boldsymbol{q}^{t-k}_{|k|},\boldsymbol{\beta}^t_{|k|}$ together form $\boldsymbol{q}^{t-k}_{|k+1|}$. I.e., $\boldsymbol{q}^{t-k}_{|k+1|} = \langle \boldsymbol{q}^{t-k}_{|k|} \circ \boldsymbol{\beta}^t_{|k|}\rangle$. Now the induction hypothesis can be applied:

$$= E\Big[\max_{\boldsymbol{\beta}^t_{|k|}}\Big[E\big[R(s^t,\mathbf{a}^t) \mid \boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}_{|k|},\boldsymbol{\beta}^t_{|k|}\big] + E\big[V_k^{t+1,*}(\boldsymbol{b}^{t-k+1},\boldsymbol{q}^{t-k+1}_{|k|}) \mid \boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}_{|k+1|} = \langle \boldsymbol{q}^{t-k}_{|k|} \circ \boldsymbol{\beta}^t_{|k|}\rangle\big]\Big] \;\Big|\; \boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|}\Big]$$

$$\geq E\Big[\max_{\boldsymbol{\beta}^t_{|k|}}\Big[E\big[R(s^t,\mathbf{a}^t) \mid \boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}_{|k|},\boldsymbol{\beta}^t_{|k|}\big] + V_{k+1}^{t+1,*}(\boldsymbol{b}^{t-k},\langle \boldsymbol{q}^{t-k}_{|k|} \circ \boldsymbol{\beta}^t_{|k|}\rangle)\Big] \;\Big|\; \boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|}\Big]$$

Now we make the outer expectation over $\mathbf{o}^{t-k}$ explicit. In particular $\boldsymbol{b}^{t-k}$ depends on $\mathbf{o}^{t-k}$ and $\boldsymbol{q}^{t-k}_{|k|} = \boldsymbol{q}^{t-k-1}_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}}$, leading to

$$= \sum_{\mathbf{o}^{t-k}} \Pr(\mathbf{o}^{t-k}|\boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|}) \max_{\boldsymbol{\beta}^t_{|k|}}\Big[E\big[R(s^t,\mathbf{a}^t) \mid \boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k-1}_{|k+1|}\Downarrow_{\mathbf{o}^{t-k}},\boldsymbol{\beta}^t_{|k|}\big] + V_{k+1}^{t+1,*}(\boldsymbol{b}^{t-k},\langle \boldsymbol{q}^{t-k-1}_{|k+1|}\Downarrow_{\mathbf{o}^{t-k}} \circ \boldsymbol{\beta}^t_{|k|}\rangle)\Big] \tag{D.2.24}$$

where $\boldsymbol{\beta}^t_{|k|}$ is that length-k joint BG policies that is selected (is maximizing) for $\mathbf{o}^{t-k}$. Let us combine these selected policies in one 'First-Joint-Observation policy' term: $\boldsymbol{\beta}^{FJO,t}_{|k+1|} = \langle \beta^{FJO,t}_{i,|k+1|} \rangle_{i \in \mathcal{D}}$ where an individual policy maps $\mathbf{o}^{t-k}$ to $\beta^t_{i,|k|}$ the individual length-k BG policy $\beta^{FJO,t}_{i,|k+1|}(\mathbf{o}^{t-k}) = \beta^t_{i,|k|}$ and thus

$$\beta^{FJO,t}_{i,|k+1|} : \mathcal{O}^{t-k} \times \vec{\mathcal{O}}^k_i \to \mathcal{A}_i. \tag{D.2.25}$$

We also write $\boldsymbol{\beta}^{FJO,t}_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}} = \boldsymbol{\beta}^t_{|k|}$. Using this notation, we can rewrite to

$$\begin{aligned}
&= \max_{\boldsymbol{\beta}^{FJO,t}_{|k+1|}} \sum_{\mathbf{o}^{t-k}} \Pr(\mathbf{o}^{t-k} | \boldsymbol{b}^{t-k-1}, \boldsymbol{q}^{t-k-1}_{|k+1|}) \Big[E\big[R(s^t, \mathbf{a}^t) \mid \\
&\boldsymbol{b}^{t-k}, \boldsymbol{q}^{t-k-1}_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}}, \boldsymbol{\beta}^{FJO,t}_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}} \big] + V^{t+1,*}_{k+1}(\boldsymbol{b}^{t-k}, \langle \boldsymbol{q}^{t-k-1}_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}} \circ \boldsymbol{\beta}^{FJO,t}_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}} \rangle) \Big]
\end{aligned}$$

Because the set of joint policies $\boldsymbol{\beta}^{FJO,t}_{|k+1|}$ is a strict superset of the set of all possible $\boldsymbol{\beta}^t_{|k+1|}$, we get

$$\begin{aligned}
&\geq \max_{\boldsymbol{\beta}^t_{|k+1|}} \sum_{\mathbf{o}^{t-k}} \Pr(\mathbf{o}^{t-k} | \boldsymbol{b}^{t-k-1}, \boldsymbol{q}^{t-k-1}_{|k+1|}) \Big[E\big[R(s^t, \mathbf{a}^t) \mid \\
&\boldsymbol{b}^{t-k}, \boldsymbol{q}^{t-k-1}_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}}, \boldsymbol{\beta}^t_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}} \big] + V^{t+1,*}_{k+1}(\boldsymbol{b}^{t-k}, \langle \boldsymbol{q}^{t-k-1}_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}} \circ \boldsymbol{\beta}^t_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}} \rangle) \Big].
\end{aligned}$$

Now, using $\langle \boldsymbol{q}^{t-k-1}_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}} \circ \boldsymbol{\beta}^t_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}} \rangle = \langle \boldsymbol{q}^{t-k-1}_{|k+1|} \circ \boldsymbol{\beta}^t_{|k+1|} \rangle \Downarrow_{\mathbf{o}^{t-k}}$ and

$$E\big[R(s^t, \mathbf{a}^t) \mid \boldsymbol{b}^{t-k}, \boldsymbol{q}^{t-k}_{|k|}, \boldsymbol{\beta}^t_{|k|} \big] = \sum_{\vec{\boldsymbol{\theta}}^t_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^t_{|k|} | \boldsymbol{b}^{t-k}, \boldsymbol{q}^{t-k}_{|k|}) R(\boldsymbol{b}^t, \boldsymbol{\beta}^t_{|k|}(\vec{\boldsymbol{\theta}}^t_{|k|})) \tag{D.2.26}$$

we write

$$\begin{aligned}
&= \max_{\boldsymbol{\beta}^t_{|k+1|}} \sum_{\mathbf{o}^{t-k}} \Pr(\mathbf{o}^{t-k} | \boldsymbol{b}^{t-k-1}, \boldsymbol{q}^{t-k-1}_{|k+1|}) \Big[\sum_{\vec{\boldsymbol{\theta}}^t_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^t_{|k|} | \boldsymbol{b}^{t-k}, \boldsymbol{q}^{t-k-1}_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}}) \\
&\qquad R(\boldsymbol{b}^t, \boldsymbol{\beta}^t_{|k+1|} \Downarrow_{\mathbf{o}^{t-k}}(\vec{\boldsymbol{\theta}}^t_{|k|})) + V^{t+1,*}_k(\boldsymbol{b}^{t-k}, \langle \boldsymbol{q}^{t-k-1}_{|k+1|} \circ \boldsymbol{\beta}^t_{|k+1|} \rangle \Downarrow_{\mathbf{o}^{t-k}}) \Big]
\end{aligned} \tag{D.2.27}$$

$$\begin{aligned}
&= \max_{\boldsymbol{\beta}^t_{|k+1|}} \Big[\overbrace{\sum_{\vec{\boldsymbol{\theta}}^t_{|k+1|}} \Pr(\vec{\boldsymbol{\theta}}^t_{|k+1|} | \boldsymbol{b}^{t-k-1}, \boldsymbol{q}^{t-k-1}_{|k+1|}) R(\boldsymbol{b}^t, \boldsymbol{\beta}^t_{|k+1|}(\vec{\boldsymbol{\theta}}^t_{|k+1|}))}^{E\left[R(s^t, \mathbf{a}^t) \middle| \boldsymbol{b}^{t-k-1}, \boldsymbol{q}^{t-k-1}_{|k+1|}, \boldsymbol{\beta}^t_{|k+1|} \right]} \\
&\quad + \underbrace{\sum_{\mathbf{o}^{t-k}} \Pr(\mathbf{o}^{t-k} | \boldsymbol{b}^{t-k-1}, \boldsymbol{q}^{t-k-1}_{|k+1|}) V^{t+1,*}_k(\boldsymbol{b}^{t-k}, \langle \boldsymbol{q}^{t-k-1}_{|k+1|} \circ \boldsymbol{\beta}^t_{|k+1|} \rangle \Downarrow_{\mathbf{o}^{t-k}}) \Big]}_{E\left[V^{t+1,*}_{k+1}(\boldsymbol{b}^{t-k}, \boldsymbol{q}^{t-k}_{|k+1|}) \middle| \boldsymbol{b}^{t-k-1}, \boldsymbol{q}^{t-k-1}_{|k+1|}, \boldsymbol{\beta}^t_{|k+1|} \right]}
\end{aligned} \tag{D.2.28}$$

writing $\boldsymbol{q}^{t-k}_{|k+1|} = \langle \boldsymbol{q}^{t-k-1}_{|k+1|} \circ \boldsymbol{\beta}^{t}_{|k+1|} \rangle \Downarrow_{\mathbf{o}^{t-k}}$ we further reduce

$$
\begin{aligned}
&= \max_{\boldsymbol{\beta}^{t}_{|k+1|}} \Big[E\big[R(s^t,\mathbf{a}^t) \;\big|\; \boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|},\boldsymbol{\beta}^{t}_{|k+1|}\big] \\
&\quad + E\big[V^{t+1,*}_{k+1}(\boldsymbol{b}^{t-k},\boldsymbol{q}^{t-k}_{|k+1|}) \;\big|\; \boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|},\boldsymbol{\beta}^{t}_{|k+1|}\big]\Big] && \text{(D.2.29)} \\
&= \max_{\boldsymbol{\beta}^{t}_{|k+1|}} Q^{t,*}_{k+1}(\boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|},\boldsymbol{\beta}^{t}_{|k+1|}) && \text{(D.2.30)} \\
&= V^{t,*}_{k+1}(\boldsymbol{b}^{t-k-1},\boldsymbol{q}^{t-k-1}_{|k+1|}) && \text{(D.2.31)}
\end{aligned}
$$

which proves the induction step. The prove is completed by the base case, which is given by the last stage. I.e., we need to show that for an arbitrarily chosen $\boldsymbol{b}^{h-k-2}$ (corresponding to some $\vec{\boldsymbol{\theta}}^{h-k-2}$) and $\boldsymbol{q}^{h-k-2}_{|k+1|}$

$$
E\big[V^{h-1,*}_{k}(\boldsymbol{b}^{h-1-k},\boldsymbol{q}^{h-1-k}_{|k|}) \mid \boldsymbol{b}^{h-k-2},\boldsymbol{q}^{h-k-2}_{|k+1|}\big] \geq V^{h-1,*}_{k+1}(\boldsymbol{b}^{h-k-2},\boldsymbol{q}^{h-k-2}_{|k+1|}). \tag{D.2.32}
$$

Starting from the left side, we make the expectation explicit

$$
\begin{aligned}
&E\big[V^{h-1,*}_{k}(\boldsymbol{b}^{h-1-k},\boldsymbol{q}^{h-1-k}_{|k|}) \mid \boldsymbol{b}^{h-k-2},\boldsymbol{q}^{h-k-2}_{|k+1|}\big] = \\
&\quad \sum_{\mathbf{o}^{h-1-k}} \Pr(\mathbf{o}^{h-1-k}|\boldsymbol{b}^{h-k-2},\boldsymbol{q}^{h-k-2}_{|k+1|}) V^{h-1,*}_{k}(\boldsymbol{b}^{h-1-k},\boldsymbol{q}^{h-k-2}_{|k+1|}\Downarrow_{\mathbf{o}^{h-1-k}}) && \text{(D.2.33)}
\end{aligned}
$$

because we consider the last stage $V^{h-1,*}_{k}$ only consists of the expected immediate reward

$$
\begin{aligned}
&V^{h-1,*}_{k}(\boldsymbol{b}^{h-1-k},\boldsymbol{q}^{h-k-2}_{|k+1|}\Downarrow_{\mathbf{o}^{h-1-k}}) && \text{(D.2.34)} \\
&= \max_{\boldsymbol{\beta}^{h-1}_{|k|}} E\big[R(s^{h-1},\mathbf{a}^{h-1}) \;\big|\; \boldsymbol{b}^{h-k-1},\boldsymbol{q}^{h-k-2}_{|k+1|}\Downarrow_{\mathbf{o}^{h-1-k}},\boldsymbol{\beta}^{h-1}_{|k|}\big] && \text{(D.2.35)} \\
&= \max_{\boldsymbol{\beta}^{h-1}_{|k|}} \sum_{\vec{\boldsymbol{\theta}}^{h-1}_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^{h-1}_{|k|}|\boldsymbol{b}^{h-k-1},\boldsymbol{q}^{h-k-2}_{|k+1|}\Downarrow_{\mathbf{o}^{h-1-k}}) R(\boldsymbol{b}^{h-1},\boldsymbol{\beta}^{h-1}_{|k|}(\vec{\boldsymbol{\theta}}^{h-1}_{|k|})) && \text{(D.2.36)}
\end{aligned}
$$

Thus (D.2.33) equals

$$
\begin{aligned}
&\sum_{\mathbf{o}^{h-1-k}} \Pr(\mathbf{o}^{h-1-k}|\boldsymbol{b}^{h-k-2},\boldsymbol{q}^{h-k-2}_{|k+1|}) \max_{\boldsymbol{\beta}^{h-1}_{|k|}} \\
&\qquad \sum_{\vec{\boldsymbol{\theta}}^{h-1}_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^{h-1}_{|k|}|\boldsymbol{b}^{h-k-1},\boldsymbol{q}^{h-k-2}_{|k+1|}\Downarrow_{\mathbf{o}^{h-1-k}}) R(\boldsymbol{b}^{h-1},\boldsymbol{\beta}^{h-1}_{|k|}(\vec{\boldsymbol{\theta}}^{h-1}_{|k|})) && \text{(D.2.37)}
\end{aligned}
$$

Again using FJO policies, we can write

$$= \max_{\boldsymbol{\beta}^{FJO,h-1}_{|k+1|}} \sum_{\mathbf{o}^{h-1-k}} \Pr(\mathbf{o}^{h-1-k}|\boldsymbol{b}^{h-k-2},\boldsymbol{q}^{h-k-2}_{|k+1|}) \sum_{\vec{\boldsymbol{\theta}}^{h-1}_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^{h-1}_{|k|}|\boldsymbol{b}^{h-k-1},\boldsymbol{q}^{h-k-2}_{|k+1|}\Downarrow_{\mathbf{o}^{h-1-k}}) R(\boldsymbol{b}^{h-1},\boldsymbol{\beta}^{FJO,h-1}_{|k+1|}\Downarrow_{\mathbf{o}^{h-1-k}}(\vec{\boldsymbol{\theta}}^{h-1}_{|k|})) \tag{D.2.38}$$

$$\geq \max_{\boldsymbol{\beta}^{h-1}_{|k+1|}} \sum_{\mathbf{o}^{h-1-k}} \Pr(\mathbf{o}^{h-1-k}|\boldsymbol{b}^{h-k-2},\boldsymbol{q}^{h-k-2}_{|k+1|}) \sum_{\vec{\boldsymbol{\theta}}^{h-1}_{|k|}} \Pr(\vec{\boldsymbol{\theta}}^{h-1}_{|k|}|\boldsymbol{b}^{h-k-1},\boldsymbol{q}^{h-k-2}_{|k+1|}\Downarrow_{\mathbf{o}^{h-1-k}}) R(\boldsymbol{b}^{h-1},\boldsymbol{\beta}^{h-1}_{|k+1|}\Downarrow_{\mathbf{o}^{h-1-k}}(\vec{\boldsymbol{\theta}}^{h-1}_{|k|})) \tag{D.2.39}$$

because the set of FJO policies $\boldsymbol{\beta}^{FJO,h-1}_{|k+1|}$ is a strict superset of the set of $\boldsymbol{\beta}^{h-1}_{|k+1|}$. Taking together the summations in (D.2.39) yields

$$\max_{\boldsymbol{\beta}^{h-1}_{|k+1|}} \sum_{\vec{\boldsymbol{\theta}}^{h-1}_{|k+1|}} \Pr(\vec{\boldsymbol{\theta}}^{h-1}_{|k|}|\boldsymbol{b}^{h-k-2},\boldsymbol{q}^{h-k-2}_{|k+1|}) R(\boldsymbol{b}^{h-1},\boldsymbol{\beta}^{h-1}_{|k+1|}(\vec{\boldsymbol{\theta}}^{h-1}_{|k+1|})) = V^{h-1,*}_{k+1}(\boldsymbol{b}^{h-k-2},\boldsymbol{q}^{h-k-2}_{|k+1|}), \tag{D.2.40}$$

which proves the base case. □

D.3 Proofs of Chapter 5

Proof of Theorem 5.1 (Decomposition of $V^t(\boldsymbol{\pi})$). Given an additively factored immediate reward function, the value $V^t(\boldsymbol{\pi})$ of a finite-horizon factored Dec-POMDP is decomposable for any t. That is, for any joint policy $\boldsymbol{\pi}$ the value function is factored. $V^t(\boldsymbol{\pi})$ is defined as

$$V^t(\boldsymbol{\pi}) = \sum_{e\in\mathcal{E}} V^{e,t}(\boldsymbol{\pi}) = \sum_{e\in\mathcal{E}} \sum_{\mathbf{x}^t_{\mathbb{X}_e}} \sum_{\vec{\boldsymbol{\theta}}^t_{\mathbb{A}_e}} \Pr(\mathbf{x}^t_{\mathbb{X}_e},\vec{\boldsymbol{\theta}}^t_{\mathbb{A}_e}|\boldsymbol{b}^0,\boldsymbol{\pi}) Q^e_{\boldsymbol{\pi}}(\mathbf{x}^t_{\mathbb{X}_e},\vec{\boldsymbol{\theta}}^t_{\mathbb{A}_e},\boldsymbol{\pi}_{\mathbb{A}_e}(\vec{\boldsymbol{\theta}}^t_{\mathbb{A}_e})) \tag{D.3.1}$$

where, using shorthand notation $\Gamma^{\mathbb{X}} = \Gamma^{\mathbb{X}}(\mathbf{x}^{t+1}_{\mathbb{X}'_e} \cup \mathbf{o}^{t+1}_{\mathbb{A}'_e})$ and $\Gamma^{\mathbb{A}} = \Gamma^{\mathbb{A}}(\mathbf{x}^{t+1}_{\mathbb{X}_e} \cup \mathbf{o}^{t+1}_{\mathbb{A}_e})$ to denote the backup scopes and $\mathbb{X}_e \equiv \mathbb{X}(R^e) \cup \Gamma^{\mathbb{X}}$ and $\mathbb{A}_e \equiv \mathbb{A}(R^e) \cup \Gamma^{\mathbb{A}} \cup \mathbb{A}'_e$ to denote the scopes of $Q^{e,t}_{\boldsymbol{\pi}}$,

$$Q^e_{\boldsymbol{\pi}}(\mathbf{x}^t_{\mathbb{X}_e},\vec{\boldsymbol{\theta}}^t_{\mathbb{A}_e},\mathbf{a}_{\mathbb{A}_e}) = R^e(\mathbf{x}^t_e,\mathbf{a}_e) + \sum_{\mathbf{x}^{t+1}_{\mathbb{X}'_e}} \sum_{\mathbf{o}^{t+1}_{\mathbb{A}'_e}} \Pr(\mathbf{x}^{t+1}_{\mathbb{X}'_e},\mathbf{o}^{t+1}_{\mathbb{A}'_e}|\mathbf{x}^t_{\Gamma^{\mathbb{X}}},\mathbf{a}_{\Gamma^{\mathbb{A}}}) Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{t+1}_{\mathbb{X}'_e},\vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e},\boldsymbol{\pi}_{\mathbb{A}'_e}(\vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e})). \tag{D.3.2}$$

Proof. This proof assumes $\boldsymbol{\pi}$ is a pure joint policy, but can be generalized to stochastic policies. Per induction hypothesis, we assume the next-stage value function is decomposable:

$$V^{t+1}(\boldsymbol{\pi}) = \sum_{e\in\mathcal{E}} V^{e,t+1}(\boldsymbol{\pi}) \tag{D.3.3}$$

and that each $V^{e,t+1}$ can be written using Q-value functions:

$$V^{e,t+1}(\boldsymbol{\pi}) = \sum_{\mathbf{x}^{t+1}_{\mathbb{X}'_e}} \sum_{\vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e}} \Pr(\mathbf{x}^{t+1}_{\mathbb{X}'_e}, \vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e} | \boldsymbol{b}^0, \boldsymbol{\pi}) Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{t+1}_{\mathbb{X}'_e}, \vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e}, \boldsymbol{\pi}_{\mathbb{A}'_e}(\vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e})) \tag{D.3.4}$$

with $\mathbb{X}'_e$,$\mathbb{A}'_e$ the scopes of $Q^e_{\boldsymbol{\pi}}$ for stage $t+1$. Now, we can show that V^t is decomposable, although the scope of V^t might grow. Per definition we have that

$$V^t(\boldsymbol{\pi}) = E[R(s^t, \mathbf{a}) | \boldsymbol{b}^0, \boldsymbol{\pi}] + V^{t+1}(\boldsymbol{\pi}) = \sum_{s^t, \vec{\boldsymbol{\theta}}^t} \Pr(s^t, \vec{\boldsymbol{\theta}}^t | \boldsymbol{b}^0, \boldsymbol{\pi}) R(s^t, \boldsymbol{\pi}(\vec{\boldsymbol{\theta}}^t)) + \sum_{e \in \mathcal{E}} V^{e,t+1}(\boldsymbol{\pi}) \tag{D.3.5}$$

By applying the definition of the additive immediate reward function and using the induction hypothesis we get

$$V^t(\boldsymbol{\pi}) = \sum_{s^t, \vec{\boldsymbol{\theta}}^t} \Pr(s^t, \vec{\boldsymbol{\theta}}^t | \boldsymbol{b}^0, \boldsymbol{\pi}) \sum_{e \in \mathcal{E}} R^e(\mathbf{x}^t_e, \boldsymbol{\pi}_e(\vec{\boldsymbol{\theta}}^t_e)) + \sum_{e \in \mathcal{E}} \sum_{\mathbf{x}^{t+1}_{\mathbb{X}'_e}} \sum_{\vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e}} \Pr(\mathbf{x}^{t+1}_{\mathbb{X}'_e}, \vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e} | \boldsymbol{b}^0, \boldsymbol{\pi}) Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{t+1}_{\mathbb{X}'_e}, \vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e}, \boldsymbol{\pi}_{\mathbb{A}'_e}(\vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e})). \tag{D.3.6}$$

We can decompose $\Pr(\mathbf{x}^{t+1}_{\mathbb{X}'_e}, \vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e} | \boldsymbol{b}^0, \boldsymbol{\pi})$ in a fashion similar to (2.5.6) and obtain

$$V^t(\boldsymbol{\pi}) = \sum_{s^t, \vec{\boldsymbol{\theta}}^t} \Pr(s^t, \vec{\boldsymbol{\theta}}^t | \boldsymbol{b}^0, \boldsymbol{\pi}) \sum_{e \in \mathcal{E}} R^e(\mathbf{x}^t_e, \boldsymbol{\pi}_e(\vec{\boldsymbol{\theta}}^t_e)) + \sum_{s^t, \vec{\boldsymbol{\theta}}^t} \Pr(s^t, \vec{\boldsymbol{\theta}}^t | \boldsymbol{b}^0, \boldsymbol{\pi}) \sum_{e \in \mathcal{E}} \sum_{\mathbf{x}^{t+1}_{\mathbb{X}'_e}} \sum_{\mathbf{o}^{t+1}_{\mathbb{A}'_e}} \Pr(\mathbf{x}^{t+1}_{\mathbb{X}'_e}, \mathbf{o}^{t+1}_{\mathbb{A}'_e} | s^t, \boldsymbol{\pi}(\vec{\boldsymbol{\theta}}^t)) Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{t+1}_{\mathbb{X}'_e}, \vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e}, \boldsymbol{\pi}_{\mathbb{A}'_e}(\vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e})) \tag{D.3.7}$$

and thus yield

$$V^t(\boldsymbol{\pi}) = \sum_{s^t, \vec{\boldsymbol{\theta}}^t} \Pr(s^t, \vec{\boldsymbol{\theta}}^t | \boldsymbol{b}^0, \boldsymbol{\pi}) \sum_{e \in \mathcal{E}} \Big[R^e(\mathbf{x}^t_e, \boldsymbol{\pi}_e(\vec{\boldsymbol{\theta}}^t_e)) + \sum_{\mathbf{x}^{t+1}_{\mathbb{X}'_e}} \sum_{\mathbf{o}^{t+1}_{\mathbb{A}'_e}} \Pr(\mathbf{x}^{t+1}_{\mathbb{X}'_e}, \mathbf{o}^{t+1}_{\mathbb{A}'_e} | s^t, \boldsymbol{\pi}(\vec{\boldsymbol{\theta}}^t)) Q^e_{\boldsymbol{\pi}}(\mathbf{x}^{t+1}_{\mathbb{X}'_e}, \vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e}, \boldsymbol{\pi}_{\mathbb{A}'_e}(\vec{\boldsymbol{\theta}}^{t+1}_{\mathbb{A}'_e})) \Big] \tag{D.3.8}$$

where the bracketed part acts as a Q-value function for stage t. When there is no independence this value depends on the full s^t,$\boldsymbol{\pi}(\vec{\boldsymbol{\theta}}^t)$ and thus the scope of this value function includes all states factors and agents. However, we can exploit any independence that does hold. In particular, we know that the probability of a pair $\mathbf{x}^{t+1}_{\mathbb{X}'_e}$,$\mathbf{o}^{t+1}_{\mathbb{A}'_e}$ is influenced only by the parents in the DBN: the state factors with indices $\Gamma^{\mathbb{X}}(\mathbf{x}^{t+1}_{\mathbb{X}'_e} \cup \mathbf{o}^{t+1}_{\mathbb{A}'_e})$ and the action nodes of agents $\Gamma^{\mathbb{A}}(\mathbf{x}^{t+1}_{\mathbb{X}_e} \cup \mathbf{o}^{t+1}_{\mathbb{A}_e})$. We use

shorthand notation $\Gamma^{\mathbb{X}}$ and $\Gamma^{\mathbb{A}}$ to derive

$$V^t(\boldsymbol{\pi}) = \sum_{s^t,\vec{\boldsymbol{\theta}}^t} \Pr(s^t,\vec{\boldsymbol{\theta}}^t|\boldsymbol{b}^0,\boldsymbol{\pi}) \sum_{e\in\mathcal{E}} \Big[R^e(\mathbf{x}_e^t,\boldsymbol{\pi}_e(\vec{\boldsymbol{\theta}}_e^t)) + \sum_{\mathbf{x}_{\mathbb{X}'_e}^{t+1}} \sum_{\mathbf{o}_{\mathbb{A}'_e}^{t+1}} \Pr(\mathbf{x}_{\mathbb{X}'_e}^{t+1},\mathbf{o}_{\mathbb{A}'_e}^{t+1}|\mathbf{x}_{\Gamma^{\mathbb{X}}}^t,\boldsymbol{\pi}_{\Gamma^{\mathbb{A}}}(\vec{\boldsymbol{\theta}}_{\Gamma^{\mathbb{A}}}^t)) Q_{\boldsymbol{\pi}}^e(\mathbf{x}_{\mathbb{X}'_e}^{t+1},\vec{\boldsymbol{\theta}}_{\mathbb{A}'_e}^{t+1},\boldsymbol{\pi}_{\mathbb{A}'_e}(\vec{\boldsymbol{\theta}}_{\mathbb{A}'_e}^{t+1})) \Big]. \quad \text{(D.3.9)}$$

We write $\mathbb{X}_e = \mathbb{X}(R^e) \cup \Gamma^{\mathbb{X}}$, $\mathbb{A}_e = \mathbb{A}(R^e) \cup \Gamma^{\mathbb{A}} \cup \mathbb{A}'_e$ for the union of scopes of the entire bracketed part and swap the summations

$$V^t(\boldsymbol{\pi}) = \sum_{e\in\mathcal{E}} \sum_{\mathbf{x}_{\mathbb{X}_e}^t} \sum_{\vec{\boldsymbol{\theta}}_{\mathbb{A}_e}^t} \Pr(\mathbf{x}_{\mathbb{X}_e}^t,\vec{\boldsymbol{\theta}}_{\mathbb{A}_e}^t|\boldsymbol{b}^0,\boldsymbol{\pi}) \Big[R^e(\mathbf{x}_e^t,\boldsymbol{\pi}_e(\vec{\boldsymbol{\theta}}_e^t)) + \sum_{\mathbf{x}_{\mathbb{X}'_e}^{t+1}} \sum_{\mathbf{o}_{\mathbb{A}'_e}^{t+1}} \Pr(\mathbf{x}_{\mathbb{X}'_e}^{t+1},\mathbf{o}_{\mathbb{A}'_e}^{t+1}|\mathbf{x}_{\Gamma^{\mathbb{X}}}^t,\boldsymbol{\pi}_{\Gamma^{\mathbb{A}}}(\vec{\boldsymbol{\theta}}_{\Gamma^{\mathbb{A}}}^t)) Q_{\boldsymbol{\pi}}^e(\mathbf{x}_{\mathbb{X}'_e}^{t+1},\vec{\boldsymbol{\theta}}_{\mathbb{A}'_e}^{t+1},\boldsymbol{\pi}_{\mathbb{A}'_e}(\vec{\boldsymbol{\theta}}_{\mathbb{A}'_e}^{t+1})) \Big]. \quad \text{(D.3.10)}$$

At this point we have derived (D.3.1) and (D.3.2). The last stage as treated in Lemma 5.1 forms the base case and thus completes the proof. □

Bibliography

M. Aicardi, F. Davoli, and R. Minciardi. Decentralized optimal control of Markov chains with a common past information set. *IEEE Transactions on Automatic Control*, 32 (11):1028–1031, Nov. 1987.

E. Altman. Applications of Markov decision processes in communication networks. In E. A. Feinberg and A. Shwartz, editors, *Handbook of Markov Decision Processes: Methods and Applications*. Kluwer Academic Publishers, 2002.

C. Amato and S. Zilberstein. Achieving goals in decentralized POMDPs. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 593–600, 2009.

C. Amato, D. S. Bernstein, and S. Zilberstein. Optimal fixed-size controllers for decentralized POMDPs. In *Proc. of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*, May 2006.

C. Amato, D. S. Bernstein, and S. Zilberstein. Optimizing memory-bounded controllers for decentralized POMDPs. In *Proc. of Uncertainty in Artificial Intelligence*, July 2007a.

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

C. Amato, J. S. Dibangoye, and S. Zilberstein. Incremental policy generation for finite-horizon DEC-POMDPs. In *Proc. of the International Conference on Automated Planning and Scheduling*, 2009.

T. Arai, E. Pagello, and L. Parker. Editorial: Advances in multirobot systems. *IEEE Transactions on Robotics and Automation*, 18(5):655–661, Oct. 2002.

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

J. Bander and C. White, III. Markov decision processes with noise-corrupted and delayed state observations. *Journal of the Operational Research Society*, 50:660–668, 1999.

T. Basar and G. J. Olsder. *Dynamic Noncooperative Game Theory*. Academic Press, December 1995.

J. Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149–198, 2000.

R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Transition-independent decentralized Markov decision processes. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 41–48, 2003.

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

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

R. Becker, V. Lesser, and S. Zilberstein. Analyzing myopic approaches for multi-agent communication. In *Proc. of the International Conference on Intelligent Agent Technology*, pages 550–557, September 2005.

R. Bellman. *Dynamic Programming*. Princeton University Press, 1957a.

R. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5): 679–684, 1957b.

D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of Markov decision processes. In *Proc. of Uncertainty in Artificial Intelligence*, pages 32–37, 2000.

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

D. S. Bernstein, E. A. Hansen, and S. Zilberstein. Bounded policy iteration for decentralized POMDPs. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1287–1292, 2005.

U. Bertele and F. Brioschi. *Nonserial Dynamic Programming*. Academic Press, Inc., 1972.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume I. Athena Scientific, 3rd edition, 2005.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume II. Athena Scientific, 3rd edition, 2007.

A. Beynier and A.-I. Mouaddib. A polynomial algorithm for decentralized Markov decision processes with temporal constraints. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 963–969, 2005.

A. Beynier and A.-I. Mouaddib. An iterative algorithm for solving constrained decentralized Markov decision processes. In *Proc. of the National Conference on Artificial Intelligence*. AAAI Press, 2006.

K. Binmore. *Fun and Games*. D.C. Heath and Company, 1992.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

P.-T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis. A decision-theoretic approach to task assistance for persons with dementia. In *Proc. of the International Joint Conference on Artificial Intelligence*, 2005.

R. Bordini, M. Dastani, J. Dix, and A. El Fallah Segrouchni, editors. *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, 2005.

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

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

C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1-2):49–107, 2000.

X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proc. of Uncertainty in Artificial Intelligence*, pages 33–42, 1998.

M. E. Bratman. *Intention, Plans and Practical Reason*. Harvard University Press, 1987.

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

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

A. Carlin and S. Zilberstein. Value of communication in decentralized POMDPs. In *Proc. of the AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*, pages 16–21, May 2009.

A. R. Cassandra. *Exact and approximate algorithms for partially observable Markov decision processes*. PhD thesis, Brown University, 1998.

I. Chades, B. Scherrer, and F. Charpillet. A heuristic approach for solving decentralized-POMDP: assessment on the pursuit problem. In *Proc. of the 2002 ACM Symposium on Applied Computing*, pages 57–62, 2002.

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

P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3):213–261, 1990.

P. R. Cohen and H. J. Levesque. Confirmations and joint action. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 951–957. Morgan Kaufmann, 1991a.

P. R. Cohen and H. J. Levesque. Teamwork. *Nous*, 25(4), 1991b.

T. A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):165–195, 2004.

R. M. Dawes. *Rational Choice in an Uncertain World*. Hartcourt Brace Jovanovich, 1988.

D. P. de Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.

M. H. DeGroot. *Optimal Statistical Decisions*. Wiley-Interscience, Apr. 2004.

J. S. Dibangoye, A.-I. Mouaddib, and B. Chai-draa. Point-based incremental pruning heuristic for solving finite-horizon DEC-POMDPs. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 569–576, 2009.

F. Doshi and N. Roy. The permutable POMDP: fast solutions to POMDPs for preference elicitation. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 493–500, 2008.

P. Doshi, Y. Zeng, and Q. Chen. Graphical models for interactive POMDPs: representations and solutions. *Autonomous Agents and Multi-Agent Systems*, 18(3):376–416, 2008.

M. J. Druzdzel and R. R. Flynn. *Encyclopedia of Library and Information Science*, chapter Decision Support Systems. Marcel Dekker, 2003.

E. H. Durfee. Distributed problem solving and planning. In *Mutli-agents systems and applications*, pages 118–149. Springer-Verlag New York, Inc., 2001.

R. Emery-Montemerlo. *Game-Theoretic Control for Robot Teams*. PhD thesis, Carnegie Mellon University, 2005.

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

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

J. L. Fernández, R. Sanz, R. G. Simmons, and A. R. Diéguez. Heuristic anytime approaches to stochastic decision processes. *Journal of Heuristics*, 12(3):181–209, 2006. ISSN 1381-1231.

Y. Gal and A. Pfeffer. Networks of influence diagrams: A formalism for representing agents' beliefs and decision-making processes. *Journal of Artificial Intelligence Research*, 33:109–147, 2008.

M. P. Georgeff, B. Pell, M. E. Pollack, M. Tambe, and M. Wooldridge. The belief-desire-intention model of agency. In *ATAL '98: Proceedings of the 5th International Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages*, pages 1–10, 1999.

A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In *Advances in Neural Information Processing Systems 20*, 2008.

P. J. Gmytrasiewicz and P. Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

P. J. Gmytrasiewicz and E. H. Durfee. A rigorous, operational formalization of recursive modeling. In *Proc. of the International Conference on Multiagent Systems*, pages 125–132, 1995.

P. J. Gmytrasiewicz, S. Noh, and T. Kellogg. Bayesian update of recursive agent models. *User Modeling and User-Adapted Interaction*, 8(1-2):49–69, 1998.

C. V. Goldman and S. Zilberstein. Optimizing information exchange in cooperative multi-agent systems. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 137–144, 2003.

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

C. V. Goldman and S. Zilberstein. Communication-based decomposition mechanisms for decentralized MDPs. *Journal of Artificial Intelligence Research*, 32:169–202, 2008.

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

C. Grappiolo, S. Whiteson, G. Pavlin, and B. Bakker. Integrating distributed Bayesian inference and reinforcement learning for sensor management. In *FUSION 2009: Proceedings of the Twelfth International Conference on Information Fusion*, pages 93–101, July 2009.

B. J. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.

B. J. Grosz and C. Sidner. Plans for discourse. In *Intentions in Communication*. MIT Press, 1990.

C. Guestrin, D. Koller, and R. Parr. Max-norm projections for factored MDPs. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 673–680, 2001a.

C. Guestrin, D. Koller, and R. Parr. Solving factored POMDPs with linear value functions. In *IJCAI '01 workshop on Planning under Uncertainty and Incomplete Information*, pages 67–75, 2001b.

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

C. Guestrin, S. Venkataraman, and D. Koller. Context specific multiagent coordination and planning with factored MDPs. In *Proc. of the National Conference on Artificial Intelligence*, pages 253–259, 2002b.

C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

E. A. Hansen and Z. Feng. Dynamic programming for POMDPs using a factored state representation. In *Proc. of the International Conference on Artificial Intelligence Planning Systems*, pages 130–139, Apr. 2000.

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

M. Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

J. Hefferon. *Linear Algebra*. CreateSpace, 2008. URL `http://joshua.smcvt.edu/linearalgebra/`.

J. Hoey, A. von Bertoldi, P. Poupart, and A. Mihailidis. Assisting persons with dementia during handwashing using a partially observable Markov decision process. In *In Proceedings of the International Conference on Vision Systems*, 2007.

R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.

R. A. Howard and J. E. Matheson. Influence diagrams. In *The Principles and Applications of Decision Analysis, Vol. II.*, pages 719–763. Strategic Decisions Group, 1984/2005. Reprinted, Decision Analysis 2.

K. Hsu and S. Marcus. Decentralized control of finite state Markov processes. *IEEE Transactions on Automatic Control*, 27(2):426–431, Apr. 1982.

M. N. Huhns, editor. *Distributed Artificial Intelligence*. Pitman Publishing Ltd., 1987.

N. R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence*, 75(2):195–240, 1995.

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

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

M. J. Kearns. Graphical games. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, Sept. 2007.

M. J. Kearns, M. L. Littman, and S. P. Singh. Graphical models for game theory. In *Proc. of Uncertainty in Artificial Intelligence*, pages 253–260, 2001.

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

H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa. RoboCup: The robot world cup initiative. In *Proc. of the International Conference on Autonomous Agents*, 1997.

H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjoh, and S. Shimada. 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*, pages 739–743, Oct. 1999.

J. R. Kok and N. Vlassis. Using the max-plus algorithm for multiagent decision making in coordination graphs. In *RoboCup-2005: Robot Soccer World Cup IX*, Osaka, Japan, July 2005.

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

D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45(1):181–221, Oct. 2003.

D. Koller and R. Parr. Computing factored value functions for policies in structured MDPs. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1332–1339, 1999.

D. Koller and R. Parr. Policy iteration for factored MDPs. In *Proc. of Uncertainty in Artificial Intelligence*, pages 326–334, 2000.

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

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

F. R. Kschischang, B. J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

H. W. Kuhn. Extensive games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(10):570–576, 1950.

H. W. Kuhn. Extensive games and the problem of information. *Annals of Mathematics Studies*, 28:193–216, 1953.

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

L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Proc. of the European Conference on Machine Learning*, pages 656–671, 2008.

V. Lesser, C. L. Ortiz Jr., and M. Tambe, editors. *Distributed Sensor Networks: A Multiagent Perspective*, volume 9. Kluwer Academic Publishers, May 2003.

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

J. Liu and K. P. Sycara. Exploiting problem structure for distributed constraint optimization. In *Proc. of the International Conference on Multiagent Systems*, pages 246–253, 1995.

Y. Liu and P. Stone. Value-function-based transfer for reinforcement learning using structure mapping. In *Proc. of the National Conference on Artificial Intelligence*, pages 415–20, 2006.

H.-A. Loeliger. An introduction to factor graphs. *IEEE Signal Processing Magazine*, 21 (1):28–41, 2004.

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

R. T. Maheswaran, C. M. Rogers, R. Sanchez, P. A. Szekely, G. Gati, K. Smyth, and C. VanBuskirk. Multi-agent systems for the real world. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 1281–1282, 2009.

S. M. Majercik and M. L. Littman. Contingent planning under uncertainty via stochastic satisfiability. *Artificial Intelligence*, 147(1-2):119–162, 2003.

J. Marecki and M. Tambe. On opportunistic techniques for solving decentralized Markov decision processes with temporal constraints. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 1–8, 2007.

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

N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L. P. Kaelbling, T. Dean, and C. Boutilier. Solving very large weakly coupled Markov decision processes. In *Proc. of the National Conference on Artificial Intelligence*, pages 165–172, 1998.

P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161:149–180, 2005.

J. M. Mooij. *Understanding and Improving Belief Propagation.* PhD thesis, Radboud University Nijmegen, May 2008a.

J. M. Mooij. libDAI: library for discrete approximate inference, 2008b. URL `http://www.jorismooij.nl/`.

K. P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, UC Berkeley, Computer Science Division, July 2002.

K. P. Murphy and Y. Weiss. The factored frontier algorithm for approximate inference in DBNs. In *Proc. of Uncertainty in Artificial Intelligence*, pages 378–385, 2001.

R. Nair and M. Tambe. Hybrid BDI-POMDP framework for multiagent teaming. *Journal of Artificial Intelligence Research*, 23:367–420, 2005.

R. Nair, M. Tambe, and S. Marsella. Team formation for reformation. In *Proc. of the AAAI Spring Symposium on Intelligent Distributed and Embedded Systems*, 2002.

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

R. Nair, M. Tambe, and S. Marsella. Team formation for reformation in multiagent domains like RoboCupRescue. In *Proc. of RoboCup-2002 International Symposium*, 2003b.

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

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

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

J. F. Nash. Equilibrium points in N-person games. *Proc. of the National Academy of Sciences of the United States of America*, 36:48–49, 1950.

F. Oliehoek and A. Visser. A hierarchical model for decentralized fighting of large scale urban fires. In *AAMAS'06 Workshop on Hierarchical Autonomous Agents and Multi-Agent Systems (H-AAMAS)*, pages 14–21, May 2006.

F. Oliehoek and N. Vlassis. Dec-POMDPs and extensive form games: equivalence of models and algorithms. IAS technical report IAS-UVA-06-02, Intelligent Systems Lab, University of Amsterdam, Amsterdam, The Netherlands, Apr. 2006.

F. Oliehoek, M. T. J. Spaan, and N. Vlassis. Best-response play in partially observable card games. In *Benelearn 2005: Proceedings of the 14th Annual Machine Learning Conference of Belgium and the Netherlands*, pages 45–50, Feb. 2005a.

F. Oliehoek, N. Vlassis, and E. de Jong. Coevolutionary Nash in poker games. In *BNAIC 2005: Proceedings of the 17th Belgian-Dutch Conference on Artificial Intelligence*, pages 188–193, Oct. 2005b.

F. A. Oliehoek and N. Vlassis. Q-value functions for decentralized POMDPs. In *Proc. of The International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 833–840, May 2007a.

F. A. Oliehoek and N. Vlassis. Q-value heuristics for approximate solutions of dec-POMDPs. In *Proc. of the AAAI spring symposium on Game Theoretic and Decision Theoretic Agents*, pages 31–37, Mar. 2007b.

F. A. Oliehoek, E. D. de Jong, and N. Vlassis. The parallel Nash memory for asymmetric games. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 337–344, July 2006.

F. A. Oliehoek, J. F. Kooij, and N. Vlassis. A cross-entropy approach to solving Dec-POMDPs. In *International Symposium on Intelligent and Distributed Computing*, pages 145–154, Oct. 2007a.

F. A. Oliehoek, M. T. J. Spaan, and N. Vlassis. Dec-POMDPs with delayed communication. In *AAMAS Workshop on Multi-agent Sequential Decision Making in Uncertain Domains*, May 2007b.

F. A. Oliehoek, N. Vlassis, and M. T. J. Spaan. Properties of the QBG-value function. IAS technical report IAS-UVA-07-04, Intelligent Systems Lab, University of Amsterdam, Amsterdam, The Netherlands, Feb. 2007c.

F. A. Oliehoek, J. F. Kooi, and N. Vlassis. The cross-entropy method for policy search in decentralized POMDPs. *Informatica*, 32:341–357, 2008a.

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

F. A. Oliehoek, M. T. J. Spaan, S. Whiteson, and N. Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *Proc. of The International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 517–524, May 2008c.

F. A. Oliehoek, S. Whiteson, and M. T. J. Spaan. Lossless clustering of histories in decentralized POMDPs. In *Proc. of The International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 577–584, May 2009.

F. A. Oliehoek, S. Whiteson, and M. T. J. Spaan. Exploiting agent and type independence in collaborative graphical Bayesian games. 2010. (article under submission).

J. M. Ooi and G. W. Wornell. Decentralized control of a multiple access broadcast channel: Performance bounds. In *Proc. of the 35th Conference on Decision and Control*, pages 293–298, 1996.

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

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

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

J. P. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1446–1451, 2007.

J. Pearl. *Probabilistic Reasoning In Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

L. Peshkin. *Reinforcement Learning by Policy Search*. PhD thesis, Brown University, 2001.

L. Peshkin, K.-E. Kim, N. Meuleau, and L. P. Kaelbling. Learning to cooperate via policy search. In *Proc. of Uncertainty in Artificial Intelligence*, pages 307–314, 2000.

M. Petrik and S. Zilberstein. Average-reward decentralized Markov decision processes. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1997–2002, 2007.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1025–1032, 2003.

P. Poupart. *Exploiting Structure to Efficiently Solve Large Scale Partially Observable Markov Decision Processes*. PhD thesis, Department of Computer Science, University of Toronto, 2005.

W. B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Wiley's Series in Probability and Statistics. Wiley-Interscience, 2007.

M. L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., 1994.

D. V. Pynadath and M. Tambe. Multiagent teamwork: Analyzing the optimality and complexity of key theories and models. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 873–880, 2002a.

D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16:389–423, 2002b.

D. V. Pynadath and M. Tambe. An automated teamwork infrastructure for heterogeneous software agents and humans. *Autonomous Agents and Multi-Agent Systems*, 7(1-2):71–100, 2003.

Z. Rabinovich, C. V. Goldman, and J. S. Rosenschein. The complexity of multiagent systems: the price of silence. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 1102–1103, 2003.

B. Rathnasabapathy, P. Doshi, and P. Gmytrasiewicz. Exact solutions of interactive POMDPs using behavioral equivalence. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 1025–1032, 2006.

I. Romanovskii. Reduction of a game with complete memory to a matrix game. *Soviet Mathematics*, 3:678–681, 1962.

M. T. Rosenstein, Z. Marx, L. P. Kaelbling, and T. G. Dietterich. To transfer or not to transfer. In *NIPS'05 Workshop on Inductive Transfer: 10 Years Later*, 2005.

A. Rosenthal. Nonserial dynamic programming is optimal. In *STOC '77: Proceedings of the ninth annual ACM symposium on Theory of computing*, pages 98–105, 1977.

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

M. Roth, R. Simmons, and M. Veloso. Decentralized communication strategies for coordinated multi-agent policies. In A. Schultz, L. Parker, and F. Schneider, editors, *Multi-Robot Systems: From Swarms to Intelligent Automata*, volume IV. Kluwer Academic Publishers, 2005b.

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

N. Roy, G. Gordon, and S. Thrun. Planning under uncertainty for reliable health care robotics. In *Proc. of the Int. Conf. on Field and Service Robotics*, 2003.

N. Roy, G. Gordon, and S. Thrun. Finding approximate POMDP solutions through belief compression. *Journal of Artificial Intelligence Research*, 23:1–40, 2005.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2nd edition, 2003. ISBN 0137903952.

F. C. Schoute. Symmetric team problems and multi access wire communication. *Automatica*, 14:255–269, 1978.

D. Schuurmans and R. Patrascu. Direct value-approximation for factored MDPs. In *Advances in Neural Information Processing Systems 14*, pages 1579–1586. MIT Press, 2002.

P. J. Schweitzer and A. Seidman. Generalized polynomial approximations in Markovian decision processes. *Journal of Mathematical Analysis and Applications*, 110:568–582, 1985.

O. G. Selfridge, R. S. Sutton, and A. Barto. Training and tracking in robotics. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 670–672, 1985.

S. Seuken and S. Zilberstein. Memory-bounded dynamic programming for DEC-POMDPs. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 2009–2015, 2007a.

S. Seuken and S. Zilberstein. Improved memory-bounded dynamic programming for decentralized POMDPs. In *Proc. of Uncertainty in Artificial Intelligence*, July 2007b.

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

J. Shen, R. Becker, and V. Lesser. Agent interaction in distributed MDPs and its implications on complexity. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 529–536, 2006.

Y. Shoham and K. Leyton-Brown. *Multi-Agent Systems*. Cambridge University Press, 2007.

R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

M. P. Singh. *Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications*. Springer-Verlag, 1994.

S. Singh, M. R. James, and M. R. Rudary. Predictive state representations: a new theory for modeling dynamical systems. In *Proc. of Uncertainty in Artificial Intelligence*, pages 512–519, 2004a.

S. Singh, V. Soni, and M. Wellman. Computing approximate Bayes-Nash equilibria in tree-games of incomplete information. In *Proc. of the 5th ACM conference on Electronic commerce*, pages 81–90, 2004b.

S. P. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3):323–339, May 1992.

E. J. Sondik. *The optimal control of partially observable Markov decision processes*. PhD thesis, Stanford University, 1971.

V. Soni, S. Singh, and M. Wellman. Constraint satisfaction algorithms for graphical games. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 423–430. ACM Press, 2007.

E. D. Sontag. *Mathematical control theory: deterministic finite dimensional systems*. Textbooks in Applied Mathematics. Springer-Verlag New York, Inc., 2nd edition, 1998.

M. T. J. Spaan. Cooperative active perception using POMDPs. In *AAAI 2008 Workshop on Advancements in POMDP Solvers*, July 2008.

M. T. J. Spaan and P. U. Lima. A decision-theoretic approach to dynamic sensor selection in camera networks. In *Int. Conf. on Automated Planning and Scheduling*, 2009.

M. T. J. Spaan and F. S. Melo. Interaction-driven Markov games for decentralized multiagent planning under uncertainty. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2008.

M. T. J. Spaan and F. A. Oliehoek. The MultiAgent Decision Process toolbox: software for decision-theoretic planning in multiagent systems. In *AAMAS Workshop on Multiagent Sequential Decision Making in Uncertain Domains*, pages 107–121, 2008.

M. T. J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

M. T. J. Spaan, G. J. Gordon, and N. Vlassis. Decentralized planning under uncertainty for teams of communicating agents. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 249–256, 2006.

M. T. J. Spaan, F. A. Oliehoek, and N. Vlassis. Multiagent planning under uncertainty with stochastic communication delays. In *Proc. of The International Conference on Automated Planning and Scheduling*, pages 338–345, Sept. 2008.

P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110 (2):241–273, June 1999.

P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Mar. 1998.

K. P. Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, 1998.

D. Szer and F. Charpillet. An optimal best-first search algorithm for solving infinite horizon DEC-POMDPs. In *Proc. of the European Conference on Machine Learning*, pages 389–399, 2005.

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

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

I. Szita and A. Lörincz. Factored value iteration converges. *Acta Cybernetica*, 18(4): 615–635, 2008.

M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7: 83–124, 1997.

N. Tao, J. Baxter, and L. Weaver. A multi-agent policy-gradient approach to network routing. In *Proc. of the International Conference on Machine Learning*, pages 553–560, 2001.

M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(1):1633–1685, 2009.

M. E. Taylor, P. Stone, and Y. Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(1):2125–2167, 2007.

M. E. Taylor, G. Kuhlmann, and P. Stone. Autonomous transfer for reinforcement learning. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 283–290, May 2008.

S. Thrun. Is learning the *n*-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems 8*, pages 640–646, 1996.

S. Thrun and L. Pratt, editors. *Learning to learn*. Kluwer Academic Publishers, 1998.

K. J. Tierney and J. D. Goltz. Emergency response: Lessons learned from the kobe earthquake. Technical report, Disaster Research Center, 1997. URL `http://dspace.udel.edu:8080/dspace/handle/19716/202`.

L. Torrey, T. Walker, J. W. Shavlik, and R. Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Proc. of the European Conference on Machine Learning*, pages 412–424, 2005.

B. van den Broek, W. Wiegerinck, and B. Kappen. Graphical models inference in optimal control of stochastic multi-agent systems. *Journal of Artificial Intelligence Research*, 32:95–122, 2008.

P. Varaiya and J. Walrand. On delayed sharing patterns. *IEEE Transactions on Automatic Control*, 23(3):443–445, June 1978.

P. Varakantham, R. Nair, M. Tambe, and M. Yokoo. Winning back the cup for distributed POMDPs: planning over continuous belief spaces. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 289–296, 2006.

P. Varakantham, J. Marecki, Y. Yabu, M. Tambe, and M. Yokoo. 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.

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

M. Wainwright, T. Jaakkola, and A. Willsky. Tree consistency and bounds on the performance of the max-product algorithm and its generalizations. *Statistics and Computing*, 14(2):143–166, 2004.

M. de Weerdt, A. ter Mors, and C. Witteveen. Multi-agent planning: An introduction to planning and coordination. In *Handouts of the European Agent Summer School*, pages 1–32, 2005.

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

W. Wiegerinck, B. van den Broek, and B. Kappen. Optimal on-line scheduling in stochastic multiagent systems in continuous space-time. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 1–8, 2007.

A. Wilson, A. Fern, S. Ray, and P. Tadepalli. Multi-task reinforcement learning: a hierarchical Bayesian approach. In *Proc. of the International Conference on Machine Learning*, pages 1015–1022, 2007.

S. J. Witwicki and E. H. Durfee. Flexible approximation of structured interactions in decentralized Markov decision processes. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 1251–1252, 2009.

M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2002.

M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

J. Wu and E. H. Durfee. Mixed-integer linear programming for transition-independent decentralized MDPs. In *Proc. of the International Joint Conference on Autonomous Agents and Multi Agent Systems*, pages 1058–1060, 2006.

Y. Xiang. *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*. Cambridge University Press, 2002.

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

M. Yokoo. *Distributed Constraint Satisfaction: Foundation of Cooperation in Multi-agent Systems*. Springer-Verlag, 2001.

E. Zermelo. Uber eine anwendung der mengenlehre auf die theorie des schachspiels. In *Proceedings of the Fifth International Congress of Mathematicians II*, pages 501–504, 1913.

L. S. Zettlemoyer, B. Milch, and L. P. Kaelbling. Multi-agent filtering with infinitely nested beliefs. In *Advances in Neural Information Processing Systems 21*, 2009.

Acknowledgments

First and foremost, I would like to thank my supervisor and co-promotor Nikos Vlassis. Not only was it he who provided excellent guidance in finding an interesting research topic, he was also the person who convinced me to start a Ph.D. in the first place. We have had long and enthusiastic brainstorm sessions during my master and the first years of my Ph.D. research. When Nikos moved to Greece, these sessions were replaced by emails and telephone conversations, equally long and enthusiastic. I am very grateful for all his feedback, since without it, this thesis would never have reached the level it did.

I am grateful to my promotor Frans Groen for his insightful comments and the monthly meetings we had and that kept me focused and on schedule, and to Arnoud Visser, who shielded me from many project administrative tasks, thereby allowing me to concentrate on research.

Special thanks go to all the coauthors I had the pleasure of working with. In particular, Matthijs Spaan and Shimon Whiteson deserve to be mentioned. The collaboration with Matthijs has been very fruitful. Especially the shared code base we used for experiments has been a great success and resulted in the Multiagent decision process toolbox. Shimon has given me great support during the last two years and has been a excellent tutor in clear reasoning and argumentation.

I would like to thank all colleagues of the intelligent autonomous systems group, the intelligent systems lab Amsterdam and the ICIS project for the interesting discussions we had and the stimulating environment they provided. Finally, I also would like to express my appreciation of the committee members for their feedback and their efforts spent.