Local Decision-Making in Multi-Agent Systems

Maike Kaufman

Balliol College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas Term 2010

This is my own work (except where otherwise indicated)

Candidate: Maike KAUFMAN

Signed:.....

Date:.....

Maike Kaufman
Balliol College

DPhil Thesis
Michaelmas Term 2010

Local Decision-Making in Multi-Agent Systems

Summary

This thesis presents a new approach to local decision-making in multi-agent systems with varying amounts of communication. Here, local decision-making refers to action choices which are made in a decentralized fashion by individual agents based on the information which is locally available to them.

The work described here is set within the multi-agent decision process framework. Unreliable, faulty or stochastic communication patterns present a challenge to these settings which usually rely on precomputed, centralised solutions to control individual action choices.

Various approximate algorithms for local decision-making are developed for scenarios with and without sequentiality. The construction of these techniques is based strongly on methods of Bayesian inference. Their performance is tested on synthetic benchmark scenarios and compared to that of a more conservative approach which guarantees coordinated action choices as well as a completely decentralized solution. In addition, the method is applied to a surveillance task based on real-world data.

These simulation results show that the algorithms presented here can outperform more traditional approaches in many settings and provide a means for flexible, scalable decision-making in systems with varying information exchange between agents.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor, Stephen Roberts, for his feedback and guidance and for allowing me the freedom to pursue my own ideas while shielding me from as much administrative overhead as possible.

This research was carried out within the ALADDIN project, which was jointly funded by BAE Systems and the Engineering and Physical Sciences Research Council (EPSRC).

Chapter 1

Introduction

1.1 Motivation

Artificial Intelligence, the discipline concerned with designing and building intelligent machines, is becoming increasingly influential to a wide range of Engineering applications. Since the advent of the field in the early second half of the 20th century, the presence of intelligent automated solutions to complex tasks has rapidly increased up to a point where they are vital to the smoothness and reliability of everyday life. Regardless of whether one is driving a car or using a personal computer, the process will most likely be aided by intelligent artificial devices, be it automatic lane detection or a sophisticated email spam filter. Although their user may be unaware of their presence in many cases, their adaptive capabilities are heavily relied upon. As this dependence increases and the wealth of available information exceeds an individual's ability of processing it, the need for automated task solutions becomes ever more pressing.

Distributed problem solvers, or multi-agent systems, have become a fast-growing area of research within the Artificial Intelligence community. Being applied to everything from air traffic control [Cammarata et al., 1988] to peer-to-peer based systems [Zhang et al., 2004] they offer solutions to a wealth of real-world problems. As the tasks that are being solved by employing intelligent machines are becoming increasingly complex, isolated artificial problem solvers are beginning to reach the limits

of their capabilities. Using a society of individual agents, either physical or virtual, to tackle these tasks therefore naturally suggests itself as an approach to the problem. Here, there is a distinction to be made between *distribution* and *decentralization*: In a distributed system agents are assigned (possibly independent) sub-tasks which they can tackle individually. In a distributed system on the other hand, agents take individual approaches towards solving a common task. The systems considered in this work fall into the latter category.

One challenge that comes with designing such multi-agent systems is that of agent coordination. While a single problem-solving agent need only interact with its environment, a group of agents occupying the same space will invariably influence each others spheres through their respective actions. Coordination between agents is therefore often ensured by allowing information to be exchanged between them. Alternatively, a completely decentralized solution can be sought under which agents will be less informed but independent of information obtained from others.

Agent communication may be implemented in may different ways, which lead to varying communication patterns. Some examples include mono-or bi-directional, symmetric or asymmetric and fixed or variable communication arrangements. Similarly, irregularities in communication can have different characteristics: information might be delayed, corrupted, partially received or missing entirely. The work presented here is concerned with agent systems with bi-directional communication patterns in which information is either exchanged instantaneously without loss or entirely unsuccessful.

One way of formalising such an agent system is a multi-agent decision process. These frameworks allows joint actions to be precomputed, but comes at the cost of high computational complexity. Furthermore, the exact schedule of agent communication must be known prior to run-time in order to compute the optimal actions. Both aspects limit the scope of the framework. In particular, many scenarios exist in which

the availability of communication is not known with certainty or cannot be guaranteed. Consider, for example, the ALADDIN[1] project through which the work carried out here was funded. The aim of this project is to develop techniques for dealing with the uncertainty and dynamics inherent in distributed and decentralized systems. The particular application domain of interest is disaster management. Here, many scenarios in which communication will be sparse and irregular are imaginable.

Rather than making restricting assumptions about the availability of communication, it would be desirable to develop a treatment which can handle sparse, irregular and/or non-deterministic communication patterns. Achieving this would require a decision-making algorithm which is flexible enough to function under varying amounts of information about the system and at the same time lean enough to allow widespread application.

This thesis will describe decision-making algorithms based on local computations carried out by individual agents. This approach has two advantages: by decentralizing the choice of actions, the complexity of the respective computations can be greatly reduced. In addition, the agents are able to make use of information obtained from others whenever it is available, but need not rely on it at every point in time.

1.2 Overview

The work presented in this thesis draws from different areas of study, the two main disciplines being Bayesian probability theory and agent systems.

Chapter 2 provides an review of classical and Bayesian probability theory and outlines the inference methods used in this work.

Existing foundations of agents and multi-agent systems are described in Chapter 3. In particular the multi-agent decision process used throughout this thesis is introduced

[1] ALADDIN stands for "autonomous learning agents for decentralised data and information networks"

there.

Chapter 4 discusses the particular challenges of communication in multi-agent systems and how they are commonly dealt with. It then introduces the novel approach towards decision-making in systems with sparse communication taken in this work, which is based on allowing differing beliefs about the state of the world. A formal treatment of centralized and localized decision-making in this setting is developed. As a result two main challenges of variable communication in agent systems become apparent: an infinite regress in agent reasoning and the infinite time-horizon of sequential decision-making.

The complexity of the latter motivates a temporary focus on one-step scenarios in the following 3 chapters. Chapter 5 describes one existing and two new benchmark algorithms on which to test the decision-making algorithms developed in the following chapter. It also summarizes other existing decision-making algorithms against which their performance can be tested.

Chapter 6 develops three novel algorithms for approximate local decision-making in multi-agent systems. One algorithm is based on exhaustive iteration, one uses a static heuristic and one is based on learning iteratively from past observations. All algorithms are tested on the benchmark problems.

In Chapter 7 the heuristic algorithm is successfully applied to a non-synthetic data set. The particular task considered is the coordination of a set of agents which record surveillance data from several cameras.

Chapter 8 builds on the previous chapters to extend the treatment to include settings with sequentiality. In these scenarios the inference steps needed will turn out to be more complex than in the one-step case. The algorithm developed is tested on sequential variants of the benchmark tasks introduced in Chapter 5.

Finally, Chapter 9 concludes the thesis.

Chapter 2

Probability Theory

2.1 Introduction

The work presented here will rely strongly on methods of *probability theory* to deal with the uncertainty encountered in agent systems. Uncertainty is an inherent property of most real-world settings, caused by imperfect and incomplete information about the underlying physical systems. Information about these systems is obtained by observing data. This will in general be noisy (imperfect) and only a finite (incomplete) amount will be available. Performing inference about such environments requires a framework which allows describing and quantifying the respective uncertainty. Probability Theory provides a mathematically sound means to do just that.

This chapter will give an overview over the basic terminology and rules of probability theory, which will be used throughout the remainder of this thesis. For a more detailed treatment of probability theory see e.g. [Jaynes, 2003, MacKay, 2003]

2.2 Classical Probability Theory

A discrete or continuous variable whose value depends on the outcome of some event is denoted a *random variable*. Let x be such a random variable with possible outcomes $\{x_1, \ldots, x_k\}$ which may be finite or infinite and discrete or continuous. The probability

that x takes on the value x_i is denoted $p(x = x_i)$, which will be abbreviated as $p(x_i)$. In the *classical* approach to probability theory this probability is given by the frequency with which the event $x = x_i$ is observed in a series of trials of length N in the limit of $N \to \infty$:

$$p(x_i) = \frac{N_{x_i}}{N} \tag{2.1}$$

The notation $p(x)$ is used to refer to the probability distribution over x. This probability distribution can have an arbitrary functional form but is subject to two general constraints:

1. The probability must always be greater or equal to zero, $p(x_i) \geq 0 \; \forall \; x_i$ and
2. The total probability of all possible event outcomes must be normalised to one,

$$\sum_x p(x) = 1 \quad \text{or} \quad \int p(x)\, dx = 1 \tag{2.2}$$

depending on whether the set of possible outcomes is discrete or continuous

Given two random variables x and y the *joint probability* of observing the pair $\{x_k, y_l\}$ is denoted by $p(x = x_k, y = y_l)$. Similarly, the *conditional probability* $p(x = x_k | y = y_l)$ indicates the probability of encountering the value $x = x_k$ given that the value $y = y_l$ has been observed. Again, the abbreviated expression for these probabilities is given by $p(x_k, y_l)$ and $p(x_k|y_l)$ respectively.

In analogy to $p(x)$, the classical conditional probability is given by the fraction of observations $y = y_l$ for which $x = x_k$ is also true:

$$p(x_k|y_l) = \frac{N_{x_k y_l}}{N_{y_l}}, \; N_{y_l} \to \infty \tag{2.3}$$

The normalisation constraint imposed on the distribution $p(x)$ must similarly hold for the joint distribution $p(x, y)$:

$$\sum_x \sum_y p(x, y) = 1 \tag{2.4}$$

Combining these two normalisation requirements gives the important sum rule of probabilities

$$p(x) = \sum_y p(x, y) \tag{2.5}$$

where $p(x)$ is now called the *marginal* probability distribution over x.

The equally fundamental product rule of probabilities can be derived from the definitions for the joint and conditional probabilities:

$$p(x, y) = \frac{N_{xy}}{N} = \frac{N_{xy}}{N_y}\frac{N_y}{N} = p(x|y)p(y) = p(y|x)p(x) \tag{2.6}$$

Note that due to commutativity this is interchangeable to $p(x, y) = p(y|x)p(x)$ and hence

$$p(y|x) = \frac{p(x|y)p(y)}{p(x)} = \frac{p(x|y)p(y)}{\sum_y p(x|y)p(y)} \tag{2.7}$$

This equality is known as *Bayes' Theorem* which enables expressing conditional probabilities in terms of joint and marginal probabilities.

The definitions and rules given above form the backbone of the probabilistic methods used in this work. From them, further concepts may be derived:

The *expectation, mean* or *average* of a function $f(x)$ under the probability distribution $p(x)$ is given by

$$\mathbb{E}[f(x)] = \sum_x p(x)f(x) \tag{2.8}$$

for a discrete random variable x and similarly

$$\mathbb{E}[f(x)] = \int p(x)f(x)\,dx \tag{2.9}$$

for a continuous one. In classical probability theory this can be interpreted as the average value of a random variable which is observed in a large number of trials. To describe the variation of the value of $f(x)$ around the mean the *variance* of $f(x)$ is

defined to be

$$\begin{aligned} \mathrm{var}[f(x)] &= \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2] \\ &= \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2 \end{aligned} \tag{2.10}$$

Note that the variance of the random variable x may itself be considered as a special case of this general definition with $f(x) = x$. The variance can be extended to two random variables $f(x)$ and $f(y)$ for which the *covariance* is given by

$$\begin{aligned} \mathrm{var}[f(x), f(y)] &= \mathbb{E}_{f(x),f(y)}[(f(x) - \mathbb{E}[f(x)])(f(y) - \mathbb{E}[f(y)])] \\ &= \mathbb{E}_{f(x),f(y)}[f(x)f(y)] - \mathbb{E}[f(x)]\mathbb{E}[f(y)] \end{aligned} \tag{2.11}$$

2.3 Bayesian Probability Theory

The definitions given in Section 2.2 are based on interpreting probabilities as the frequencies with which event outcomes are observed. Thus, for the concept of probabilities to be meaningfully defined, the events considered must be repeatable. In the *Bayesian* interpretation of probability theory on the other hand, probabilities are interpreted as measures of uncertainty about the truth of a proposition, which enables its use beyond repeatable events. This interpretation of probability theory does not affect any of the variables defined in the previous section, as it does not change the mathematical nature of probability distributions but merely the way in which they are interpreted.

In classical logic a statement such as “this car is red” can be either true or false. Interpreting probability theory as an extension to classical logic allows introducing degrees of belief about the truth of such a statement. In Bayesian probability theory, to say that the probability of the car being red is 0.2 is tantamount to saying “I am quite certain the car is not red”. Stating that the probability of it being red is 0.5 is equivalent to saying “I am uncertain about whether the car is red or not”. Such an

interpretation of probabilities no longer relies on the repeatability of a given event and therefore allows its application beyond the scope of frequencies.

As an example, consider quantifying the probability of an uncertain event, for example whether or not it will rain this afternoon. This is not a repeatable event, so calculating this probability from frequencies is not possible. Moreover, this is a setting in which the degree of belief about the proposition "it will rain this afternoon" might change as further information becomes available. For example, the sight of large dark clouds forming at midday will suggest that the statement is becoming more probable. The Bayesian interpretation of probability theory in combination with Bayes' Theorem provides a means with which to quantify this change in belief.

In accordance with Equation (2.7) Bayes' Theorem is given by

$$p(y|\mathcal{D},\mathcal{I}) = \frac{p(\mathcal{D}|y,\mathcal{I})p(y|\mathcal{I})}{p(\mathcal{D}|\mathcal{I})} \tag{2.12}$$

Where y is a statement ("it will rain this afternoon") about which to express a degree of belief. $\mathcal{I}$ refers to the totality of information which is available even before any observations are made and which sets the conditions under which the inference is performed (e.g. "we are situated in Britain"). $\mathcal{D}$ is any relevant information relating to the event which has been obtained from data ("clouds at midday"). The probability $p(y|\mathcal{I})$ denotes the *prior* belief about whether or not y will be true, in this case whether or not it will rain. This prior will depend on $\mathcal{I}$, the information which is available from first principles and will therefore differ for individual settings: the prior for rain in Britain will be a different one from the prior for rain in Spain. How much the belief about the proposition is altered by the data is dependent on how likely the data is, assuming the proposition were true (here, how likely one is to observe clouds when it rains) or if it were false. The probability $p(\mathcal{D}|y,\mathcal{I})$ is hence referred to as the *likelihood*. Bayes' theorem then allows to compute the conditional probability $p(\text{rain}|\text{clouds})$ that

it will rain, given that clouds are visible:

$$p(\text{rain}|\text{clouds}) = \frac{p(\text{clouds}|\text{rain})p(\text{rain})}{p(\text{clouds})} \tag{2.13}$$

More abstractly, it allows updating the prior belief about the probability of rain, given new data about its occurrence. To summarise, the individual terms in Bayes' Theorem are interpreted as

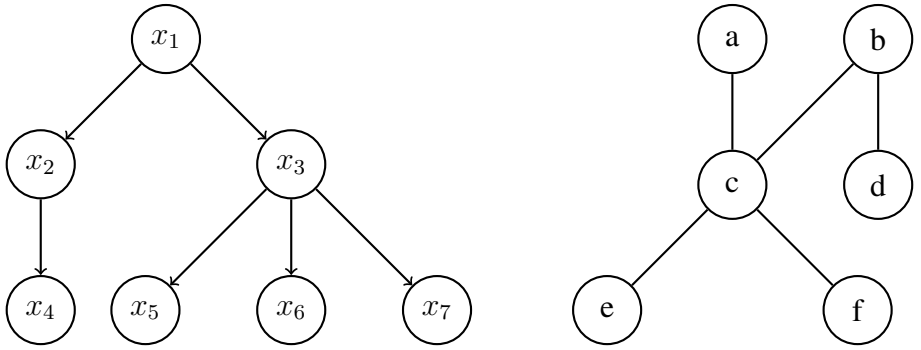
$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}} \tag{2.14}$$

Bayes' Theorem also allows incorporating further information into the belief at a later time: should further data become available, the posterior takes on the role of the prior with the new data leading to a new posterior. The posterior distribution can therefore be thought of as a current belief about the proposition, which in general only holds until new information becomes available.

To facilitate the computation of the posterior, the prior is often chosen as the *conjugate prior* to the likelihood: this is a choice of prior which ensures that the prior and the posterior will have the same functional form.

2.3.1 Graphical Models

The process of obtaining the probability of some statement given the information available by using Bayes' rule is referred to as *Bayesian Inference*. In many applications the settings are more complex than the example outlined above and performing inference on them can prove a challenging task. While successive application of Bayes' Rule is always a theoretical option it is often not viable or at least not efficient for practical purposes. Indeed, efficient inference is a large area of study in its own right which is beyond the scope of this introduction. This section will instead illustrate one popular inference framework, which has been used within this work. *Graphical models* provide a convenient way of visualising and describing interdependencies between random variables as well as facilitating efficient inference algorithms.

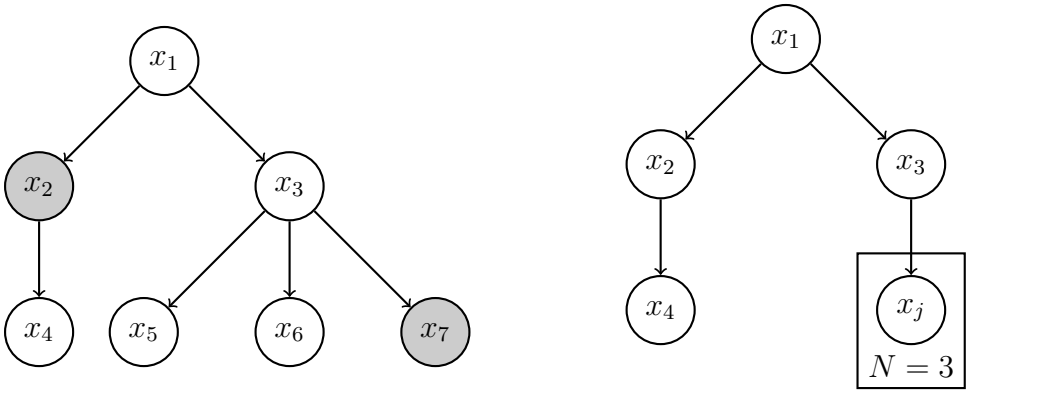


(a) Bayesian network (b) Markov random field

Figure 2.1: Examples of directed and undirected graphical models: Bayesian network and Markov Random Field

A graphical model is a graph whose structure is determined by the interdependence of variables in an inference problem: it consists of *nodes*, representing the variables and *edges* representing the interdependencies between the nodes. In a *directed* graphical model or *Bayesian network* edges carry an associated direction which identifies the conditional dependencies between variables. Here, a node which is dependent on other nodes is called a *child* while the corresponding higher node is called a *parent*. By contrast, edges in an *undirected* graphical model or *Markov random field* carry no directions (see figure 2.1 for examples of both graph types). A subset of graph nodes for which there exists an edge between any two members of the set is called a *clique*.

Graphical models are a practical means for visualising interdependencies in complex systems: from the graph it immediately becomes clear which variables depend on others and which are independent of each other. This also enables the development of general inference algorithms, which depend on abstract features of the graph (for example connectivity) rather than particularities of the inference problem at hand. Different inference algorithms exist for directed and undirected graphical models. For the work presented here, only Bayesian networks will be of interest, as they are particularly suited to expressing causal relationships between the random variables in a system.



(a) Shading of observed variables (b) Shortened visualisation of repeated nodes

Figure 2.2: Example of observed variables and repeated nodes in a Bayesian network.

Graphical models also allow for visualisation of further graph characteristics, such as observed variables and multiple nodes which have the same dependencies. Observed variables are shaded in the graph to distinguish them from unobserved ones. Multiple nodes with identical dependencies are replaced with a single plate which represents the repetition. See Figure 2.2 for an example of both cases.

To illustrate the construction of a Bayesian network, let $x_1, \ldots, x_n$ be a set of random variables with known interdependencies. The directed graph can be constructed from this information in the following way: each variable is represented by a graph node. For each dependency relation, an edge is inserted between the respective variable node and those variables on which it depends, with the direction of this edge pointing to the dependent node. The joint probability distribution over all n variables is then given by

$$p(x_1, \ldots, x_n) = p(x_1) \prod_{i=2}^{n} p(x_i | \operatorname{pa}(x_i)) \tag{2.15}$$

where $\operatorname{pa}(x_i)$ denotes the parent nodes of i. For the example graph shown in Figure 2.1a this would give:

$$p(x_1, \ldots, x_7) = p(x_1)p(x_2|x_1)p(x_3|x_1)p(x_4|x_2)p(x_5|x_3)p(x_6|x_3)p(x_7|x_3) \tag{2.16}$$

More generally the distribution over variables in the graph can be expressed in terms of factored functions of subsets of variables X_k:

$$p(x_1, \ldots, x_n) = \prod_k f_k(X_k) \tag{2.17}$$

From this joint distribution, marginals can be obtained in the usual way, by summing or integrating over variables that are not of interest, e.g:

$$p(x_i, x_j, x_k) = \sum_{l \neq i,j,k} p(x_1, \ldots, x_n) \tag{2.18}$$

in practise, this is not always the most efficient way of obtaining a marginal distribution. Appendix A details an efficient inference algorithm for directed acyclic graphs, i.e. graphs which have no closed loops. This allows calculating marginals in a particularly systematic way.

2.4 Decision Theory

Probability theory as described so far provides a framework for inferring the probabilities of propositions. In most applications this is only part of the problem at hand: given the probability distribution over a variable, the question remains of how to act upon this knowledge. Solving this task is the subject of *decision theory*.

Return to the example in Section 2.3 which considered the proposition "it will rain this afternoon". Now assume that one has two possible actions to decide between: whether or not to take an umbrella along on a walk. The outcome of this decision will clearly depend on how probable the proposition is. Intuitively, the more likely rain is the more one should want to carry an umbrella. On the other hand, the convenience gained from taking the umbrella along depends on whether or not it actually does rain: if the weather remains dry it will be of no use, worse even, it will be unnecessary baggage. Formally, this can be expressed through the concept of *utility*: different

outcomes yield different amounts of utility to the decision-maker. In the example considered here, carrying the umbrella when it does rain will provide a high utility while bringing it along when it does not rain offers a low or even negative utility. Similarly, not taking an umbrella along if the weather is good will have high utility while not bringing it if it rains will have very low utility. Which action to take therefore not only depends on the probability of different outcomes but also on their utilities.

Let $\{x_1, \ldots, x_k\}$ be the set of possible values of a random variable x and let $\{a_1, \ldots, a_k\}$ be a set of possible actions. Further, let $u(x, a)$ denote the utility gained from taking an action for a given value of the variable. For a given outcome the optimal action is the one which maximises the utility function. In a probabilistic setting in which the value of the random variable is not known with certainty, the optimal action is instead given by the one which maximises the expected utility under the variable's probability distribution:

$$a^* = \arg\max_{a_i} \int_{x_j} p(x_j|\mathcal{D})u(x_j, a_i)\, dx_j \tag{2.19}$$

where $\mathcal{D}$ is the data which has been observed.

Chapter 3

Agent Systems

3.1 Introduction

The second area of study from which the work presented here draws heavily can be broadly referred to as *multi-agent systems*. Various research communities have developed different notions about what this area encompasses and this chapter will provide an introduction to the concepts and methods commonly employed. It will begin by providing a short overview over the development of the field over the past few decades and reviewing basic terminology. This will be followed by a section on single agent and multi-agent *decision processes*, which form the basis for this work. The chapter will close with a summary of basic concepts of *game theory*, which will be used in later chapters.

3.1.1 Historic Overview

Artificial Intelligence (AI) as a research area originated in the mid-20th century as a field concerned with the study and design of single intelligent entities. Rather than treating problem solvers in an integrated way, early AI approached the task of creating intelligent machines by exploring individual aspects of intelligence, such as reasoning, vision, problem solving etc. In particular the area of planning, concerned with making a choice of action, became the predecessor of agent-based systems. This approach was

based mostly on symbolic reasoning. As its application began to encounter limitations due to complexity issues, a first generation of state/action based agent systems evolved [Jennings et al., 1998]. These were shown to be successful in several applications and subsequently became increasingly refined.

As agent tasks became more complex single agent-based solutions again began to suffer from resource bottlenecks. This resulted in the advancement of the study of distributed systems. Such early work was mainly concerned with designing cooperative agent systems which were applied successfully to some first real-world tasks, including air traffic control [Cammarata et al., 1988] and distributed vehicle monitoring [Durfee, 1988]. Throughout the development of the field it has become evident that the successful coordination of plans, intentions and responsibilities and the sharing of resources between agents are among the most important issues of cooperative agents [Jennings, 1996, Jennings et al., 1998].

More recently the study of multi-agent systems has been extended to consider systems of self-interested agents, sparking a whole new area of study. In these systems, effective agent negotiation poses one of the main design challenges. Economics-based methodologies (e.g. game-theoretic approaches or auctions) have been employed successfully in many cases to mediate between agent interests in an effective way.

Multi-agent systems continue to be an active field of study as the design of reliable intelligent agents, especially systems of interacting agents, remains a nontrivial task [Jennings et al., 1998].

3.2 Basic Agent Concepts

Although the terms “agent” and “multi-agent system” are commonly used throughout the literature and have been referenced within various communities for quite some time, no unanimous definition of what an agent or a multi-agent system is exists. This section

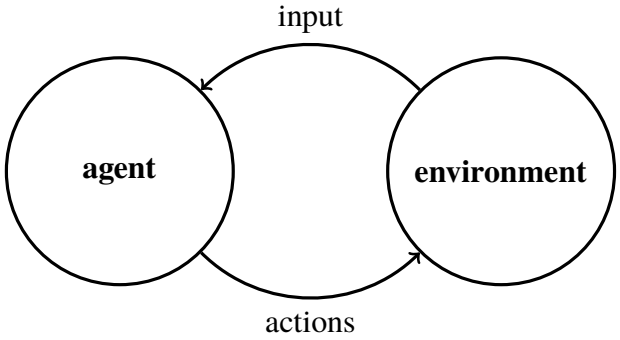


Figure 3.1: Schematics of agent-environment interaction: agents perform actions which affect the environment state. In turn the state of the environment influences an agent's choice of action

will aim to describe some of the more general concepts commonly used to characterise these systems. For a comprehensive review of Artificial Intelligence, in particular multi-agent systems, see e.g. [Russell and Norvig, 1995, Weiss, 1999a, Wooldridge, 2001].

3.2.1 Intelligent Agents

Following [Weiss, 1999b] the most elementary characteristic of an agent is its autonomy, that is, the ability to act without external interference. More precisely, "an agent is a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objective". This general definition reflects two important aspects of agents, namely

1. An agent is only meaningfully defined within an environment and as such cannot be studied in isolation

2. An agent is designed to meet a specific objective, which sets the criteria by which to evaluate its performance

For a schematic view of the agent-environment interdependence see Figure 3.1.

The environment on which an agent acts can have various properties which can be classified according to the following high-level taxonomy [Russell and Norvig, 1995]:

- *Observable vs. partially observable:* in an accessible environment it is possible for an agent to obtain complete information about the environment's state
- *Deterministic vs. non-deterministic:* a deterministic environment is one in which an action has a guaranteed reproducible effect, i.e. there is a deterministic connection between an agent's action and the following environment state
- *Episodic vs. non-episodic:* in an episodic environment the interaction between agent and environment takes place in self-contained segments (episodes) in which an agent's performance does not depend on previous episodes
- *Open vs. closed:* a closed environment will only be affected by the actions undertaken by the agent. By contrast, an open environment is influenced additionally by external factors.
- *Discrete vs. continuous:* a discrete environment comprises a fixed finite number of states while a continuous environment can undergo an infinite number of states

The simple definition of an agent given above makes no statements concerning the intelligence of autonomous agents. For an agent to be an intelligent one, it must satisfy additional criteria. Some of these might include [Weiss, 1999b, Wooldridge and Jennings, 1995]:

- *reactivity:* agents perceive and act upon their environment, choosing actions according to their design objectives
- *pro-activeness:* agents choose their actions in a goal-directed manner, that is they take the initiative to fulfil their design objectives

- *social ability:* agents are capable of interacting with others in a target-oriented fashion

A more comprehensive definition could encompass *mobility, veracity, benevolence* and *rationality* as secondary agent characteristics. However, the assumption of benevolence, i.e. the display of altruistic behaviour towards other agents imposes a certain restriction on how agents may be designed and has therefore been relaxed in many approaches.

Return to the second requirement stated in the first definition, namely that agents act in accordance to their design objective. Colloquially this could be referred to as agents "doing the Right Thing". This behaviour of choosing the right action is often referred to as *agent rationality*. The right action is the one which yields the greatest success, but in order to determine successful behaviour some quantitative *performance measure* will be needed. A rational agent can therefore be described as an agent which chooses the best possible action, given its performance measure determined by the design objective.

3.2.2 Formal Description

The aforementioned properties can be formalised by developing a more abstract description of an agent and its environment. Using this will allow a more quantitative treatment of its characteristics and performance.

Let the environment in which the agent acts be characterised by a (possibly continuous) set of environment states $S = \{s_1, s_2, \ldots\}$ and let the agent be able to choose from a (possibly continuous) set of actions $A = \{a_1, a_2, \ldots\}$. Let $\bar{s} = (s^1, s^2, s^3, \ldots)$ denote a sequence of environment states and let $\bar{S}$ refer to the set of all possible sequences of states. An agent can then be understood as a function Ag which maps sequences of environment states $\bar{s}$ onto actions:

$$\mathrm{Ag} : \bar{S} \rightarrow A \tag{3.1}$$

with the special case of a simple reflex agent whose choice of action depends exclusively on the current state of the system:

$$\mathrm{Ag} : S \rightarrow A \tag{3.2}$$

Finding the best mapping from states to actions is the challenge that lies at the heart of the study of multi-agent systems. Its solution depends both on the agent's performance measure as well as the characteristics of the environment and finding it is in general non-trivial.

As noted in the previous section the environment can be characterised by deterministic or stochastic behaviour. This characteristic is formalised by the environment's *transition function* or *transition probability function*, which determine the succession of states the environment undergoes. If the environment is deterministic, the transition function is of the form

$$T : S \times A \rightarrow S \tag{3.3}$$

In the case of a non-deterministic environment, the transition probability function defines a function mapping from states and actions onto a probability distribution over states:

$$p_T : S \times S \times A \rightarrow [1, 0] \tag{3.4}$$

where the probability of a state s' following a previous state s and action a is given by the conditional probability $p(s'|s, a)$. Recall the concept of utility, which was defined in section 2.4. This idea is directly applicable to decision-making in agent systems by assigning a utility $u(s)$ to each possible environment state s. A rational agent will then act in order to maximise its utility. In deterministic environments this is a matter of straightforward evaluation:

$$a^* = \arg\max_a \, u(s'(a, s)) \tag{3.5}$$

where the state $s'(s,a) = T(s,a)$ is the state which follows s when taking action a. In non-deterministic environments the following state is not known with certainty and the agent will have to maximise the expected utility instead:

$$a^* = \arg\max_a \mathbb{E}[u(s')|a] \tag{3.6}$$

where the expected utility is given by

$$\mathbb{E}[u(s')|a] = \sum_{s'} p_T(s'|s,a)u(s') \tag{3.7}$$

For an agent in a given environment a *policy* specifies which action to choose for any given environment state it might encounter. Consequently the optimal policy is the one which maximises (expected) utility for any possible environment state.

3.3 Single Agent Decision Process

The mathematical frameworks for describing multi-agent systems used in this work are all straightforward extensions of the single agent case. This section will therefore review the standard formulation of a single agent making action choices in an uncertain environment.

3.3.1 Markov Decision Processes

A *Markov decision process (MDP)* [Bellman, 1957a, Howard, 1960] is defined as a tuple $\{S, A, p, R\}$ where:

- $S = \{s_1, s_2, \ldots\}$ is a finite set of environment states
- $A = \{a_1, a_2, \ldots\}$ is a finite set of actions
- $p : S \times S \times A \to [0,1]$ is a state transition probability function where $p(s_n|s_m, a_k)$ denotes the probability for ending up in state s_n after taking action a_k in state s_m

- $R : S \times A \to \mathbb{R}$ is a reward function where $R(s_m, a_k)$ is the reward obtained for taking action a_k in state s_m

This decision process satisfies the Markovian property: given the current state and a constant number n of previous states, the probability of transitioning to the next state is conditionally independent of any other previous (that is "older") states. Usually, the term MDP refers to first order Markovian processes in which the current state only depends on the directly preceding state.

MDPs can be interpreted as an abstract formulation of an agent interacting with an uncertain environment: at each time-step, the environment is in one of the possible states. The agent observes this environment state and acts upon it. The environment then stochastically transitions to the next state.

As discussed in Section 3.2.2, an agent's performance is always measured with respect to some performance measure. With the above definitions this can be defined in the following way: Let a policy π be a mapping from states onto actions, $\pi : S \to A$. Then the goal in solving a given MDP is to find the optimal policy π^*, which maximises the expected future reward gained by an agent acting according to it. The expected reward, also known as the *value function* under policy π, is given by

$$V_\pi(s^0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s^t, \pi(s^t))\right] \tag{3.8}$$

where γ is a discount factor $0 < \gamma < 1$ which ensures the expectation value to remain finite and expresses a preference for immediate rewards over future rewards. If the expectation value is given by an infinite sum as in Equation (3.8) the MDP is said to have an *infinite time-horizon*. It is also possible to use a *finite time-horizon* value function, for which the discount factor can be equal to one, $0 < \gamma \leq 1$

$$V_\pi(s^0) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(s, \pi(s))\right] \tag{3.9}$$

Working within the finite-time horizon framework comes with some disadvantages. Most noticeably the finite time-horizon policy will no longer be stationary, that is it will not only depend on the state but also on the time-step at which this state is encountered. As a result, the infinite time-horizon case has been chosen for the work presented here.

Re-naming s^0 as s the expected reward given in Equation (3.8) can be written as follows:

$$\begin{aligned} V_\pi(s) &= R(s,\pi(s)) + \mathbb{E}\left[\sum_{t=1}^{\infty}\gamma^t R(s^t,\pi(s^t))\right] \\ &= R(s,\pi(s)) + \mathbb{E}\left[\gamma R(s^1,\pi(s^1)) + \sum_{t=2}^{\infty}\gamma^t R(s^t,\pi(s^t))\right] \\ &= R(s,\pi(s)) + \sum_{s^1\in S} p(s^1|s,\pi(s))\left[\gamma R(s^1,\pi(s^1)) + \sum_{t=2}^{\infty}\gamma^t R(s^t,\pi(s^t))\right] \end{aligned} \tag{3.10}$$

The expression in large brackets is similar to the first line except for a multiplication by γ. The value function can therefore be written as

$$V_\pi(s) = R(s,\pi(s)) + \gamma\sum_{s'\in S} p(s'|s,\pi(s))V_\pi(s') \tag{3.11}$$

where s^1, the state which follows s, has been re-named as s'.

It has been shown [Howard, 1960] that there exists a stationary policy π^*, which is optimal for any possible starting state. The value function under this policy, V^* is given by

$$V^*(s) = \max_a\left[R(s,a) + \gamma\sum_{s'\in S} p(s'|s,a)V^*(s')\right] \tag{3.12}$$

and the optimal policy is given by

$$\pi^* = \arg\max_a\left[R(s,a) + \gamma\sum_{s'\in S} p(s'|s,a)V^*(s')\right] \tag{3.13}$$

In addition to the value function V it can often be convenient to define the *Q-values* of a MDP which are given by

$$Q(s,a) = R(s,a) + \gamma\sum_{s'\in S} p(s'|s,a)V^*(s') \tag{3.14}$$

Hence the Q-values give the expected discounted sum of rewards when taking action a in state s and acting optimally from s' onward.

Equation (3.12) is known as the Bellman optimality equation and defines a system of $|S|$ equations with $|S|$ unknowns. Due to the nonlinearity of the $\max$ operator, however, solving this set of equations is not straightforward.

3.3.2 MDP Solutions

There exist two main approaches to solving MDPs: algorithms for known system dynamics (p and R) and algorithms which can find the optimal policy and learn the unknown dynamics. The latter field is known as *reinforcement learning*. The work presented here is concerned with systems for which the dynamics are known.

In principle Equation (3.12) could be solved by exhaustively searching the space of all possible policies π, to find the one which yields the maximum value function. However, this approach has prohibitive complexity for even a small set of states and actions, so that it is infeasible for most scenarios.

Value Iteration

Bellman [Bellman, 1957a] introduced a recursive approach to solving Equation (3.12), called *value iteration*. This algorithm is based on initialising all value functions to an arbitrary starting value $V^0(s)$ and iteratively updating the current estimate $V^t(s)$ via

$$V^{t+1}(s) = \max_a [R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V^t(s')] \tag{3.15}$$

This can be interpreted as a backward calculation for an agent which starts out at time-step t and uses the value function at that time to construct the value function for the next time-step, $t+1$. The starting value $t = 0$ can then be understood as the agent having arrived at its last time step. It is therefore usually initialised to zero. Note that for the infinite time-horizon setting this requires taking the limit $t \to \infty$.

Value Iteration has been shown to converge to the optimal value function [Bellman, 1957b]. If the Bellman error magnitude $|V^{t+1}(s) - V^t(s)|$ is smaller than ϵ the maximum error in the value function estimate is bounded by

$$\max_{s \in S} |V_{\pi(t+1)} - V^*(s)| < 2\epsilon \frac{\gamma}{1-\gamma} \tag{3.16}$$

where $\pi(t+1)$ denotes the best policy for the value function estimate V^{t+1} [Williams and Baird, 1994].

Policy Iteration

It is often the case that the policy converges more quickly than the value function, that is that $\pi(t) = \pi^*$ is obtained well before V approaches V^*. *Policy iteration* [Howard, 1960] makes use of this circumstance by iteratively finding the optimal policy rather than the optimal value function. This is achieved by alternating the following steps:

1. Initialise π^0
2. Given π^t, calculate the corresponding value function V^t according to Equation (3.11) until convergence
3. Calculate a new policy π^{t+1}, using Equation (3.13) given the approximate value function V^t
4. Repeat from 2 until there is no further change in policy

3.3.3 Partially Observable Markov Decision Processes

In the MDP framework described above, the states are assumed to be fully observable at each time-step and the only source of uncertainty is the stochasticity of the state transition function. In many settings this need not be the case. In particular, the agent may not have full access to the environment's state at any given time and hold only

partial information about it instead. For example this might be the case when an agent can only make noisy observations which do not uniquely define the environment state.

A *partially observable Markov decision process (POMDP)* [Sondik, 1971, Sondik, 1978] is defined as a tuple $\{S, A, O, p_T, p_O, R\}$ where:

- $S = \{s_1, s_2, \ldots\}$ is a finite set of environment states
- $A = \{a_1, a_2, \ldots\}$ is a finite set of actions
- $O = \{o_1, o_2, \ldots\}$ is a finite set of observations
- $p_T : S \times S \times A \rightarrow [0, 1]$ is the transition probability function where $p_T(s_n|s_m, a_k)$ denotes the probability for ending up in state s_n after taking action a_k in state s_m
- $p_O : O \times S \rightarrow [0, 1]$ is the observation probability function where $p_O(o_l|s_m)$ is the probability of making observation o_l in state s_m
- $R : S \times A \rightarrow \mathbb{R}$ is the reward function where $R(s_m, a_k)$ is the reward obtained for taking action a_k in state s_m

Similar to the MDP, the POMDP can be interpreted as an abstract formulation of an agent interacting in an even more uncertain environment. Again, the system is in one of the possible states at any given time-step. In contrast to the MDP the agent here does not have full information about this state. Instead it must use the partial observation o made at that point in time to infer a distribution over possible states. The choice of action can then be based on this belief. Let b be a probability distribution over S and let $b^t(s)$ denote the probability over current states of the system at time t. Then after taking an action a, transitioning to a new (unknown) state s' and making observation

o' the belief over the current state is updated via Bayes' Rule to give:

$$b^{t+1}(s') = \frac{p_O(o'|s')\ p(s')}{\sum_{s'} p_O(o'|s')\ p(s')} \tag{3.17}$$

$$= \frac{p_O(o'|s')\ \sum_s p_T(s'|s,a) b^t(s)}{\sum_{s'} p_O(o'|s') \sum_s p_T(s'|s,a) b^t(s)} \tag{3.18}$$

While the aim in solving a MDP was to find a mapping from states to actions, the challenge in solving a POMDP lies in finding a mapping from the belief space to actions. To solve for the optimal policy, the POMDP can be re-cast as a belief-state MDP, $\{B, A, T, R_B\}$ (see e.g. [Kaelbling et al., 1998]) where

- B is the continuous set of belief states
- $A = \{a_1, a_2, \ldots\}$ is the finite set of actions defined by the POMDP
- $T : B \times O \times A \rightarrow B$ is the transition function of the belief states, where $T = b^{t+1}$ as defined above
- $R_B : B \times A \rightarrow \mathbb{R}$ is the belief state reward function, derived from the POMDP reward function as $R_B(b,a) = \sum_s b(s) R(s,a)$

This re-definition allows the POMDP to be solved with similar techniques as introduced for the MDP. However, the belief state space is real-valued and continuous and therefore introduces additional complexity. Given a belief sate b and a policy π, the expected discounted reward under π is given by

$$V_\pi(b) = R_B(b,a) + \gamma \sum_{b' \in B} \sum_{o' \in O} p_O(o'|b,a) p(b'|b,a,o') V_\pi(b') \tag{3.19}$$

The finite-horizon POMDP optimal value function has been shown to be piecewise-linear and convex [Sondik, 1971, Sondik, 1978], which means that it can be expressed as a finite linear combination of vectors. This has lead to a number of exact finite-horizon solution algorithms which are POMDP variants of value iteration [Sondik,

1971, Smallwood and Sondik, 1973, Cassandra et al., 1997, Kaelbling et al., 1998] and policy iteration [Sondik, 1971, Sondik, 1978, Hansen, 1998a, Hansen, 1998b] described in section 3.3.2. The complexity of solving for optimal policies in a finite-horizon POMDP is PSPACE-complete[1] [Papadimitriou and Tsitsiklis, 1987]. As a result the application of exact solution algorithms is limited to very small problem settings.

In the infinite-horizon case the value function will still be convex but in general not piecewise-linear. The value function can however be approximated arbitrarily closely by a succession of finite-horizon functions V^t for $t \rightarrow \infty$ [Sondik, 1978]. Approximate solutions have therefore been developed both for the finite- and the infinite-horizon case. These approaches are mostly based on limiting either the policy space, e.g. [Littman, 1994, Meuleau et al., 1999, Peshkin et al., 2001] or approximating the value function, e.g. [Pineau et al., 2003, Spaan and Vlassis, 2005].

3.4 Multi-Agent Systems

Not all tasks lend themselves well to being solved by a single autonomous agent. For reasons such as resource constraints, computational complexity or decentralization of data, an individual agent might be insufficient or ineffective at solving the application. Instead, one might wish to employ a group of agents working independently or together to solve a given task. Such an approach provides the possibility of designing more complex and powerful problem solvers.

[1] PSPACE problems are those solvable by a deterministic Turing machine using $\mathcal{O}(p(n))$ space, where $p(n)$ is a polynomial of n. A PSPACE-complete problem is one for which every other PSPACE problem has a polynomial time many-to-one reduction to it. Complexity classes with respect to time are defined in a similar way, e.g. P denotes the class of problems which are solvable by a deterministic Turing machine in $\mathcal{O}(p(n))$ time.

3.4.1 Distributed Systems

A *distributed system* or *distributed computing* is a very general concept describing an environment consisting of multiple, possibly heterogeneous components. A distributed system could be anything from a client-server network to a parallellised computing algorithm, to a collection of mobile robots.

A *multi-agent system* on the other hand, is a more specific example of a distributed system. Loosely following [Sycara, 1998] it can be defined by three key characteristics, namely

1. Agents have incomplete information about the system or insufficient capabilities for solving a task autonomously
2. The system exhibits no global control
3. Data is decentralized within the system

This definition addresses some of the shortcomings of single agent systems mentioned above and highlights how multi-agent systems can provide a framework for solving tasks in which individual agents fail: they are designed from first principles to act in uncertain, decentralized environments. The following section will provide a more formal treatment of the multi-agent framework used in this work. Multi-agent systems can be designed to be either self-interested or cooperative with the focus of this work being on cooperative agent systems.

3.5 Multi-Agent Decision Process

The single-agent formalisms described in Section 3.3 can easily be extended to model systems containing multiple agents. These treatments have attracted much interest and are increasingly used to design cooperative decentralized multi-agent systems [Kael-

bling et al., 1998, Pynadath and Tambe, 2002, Seuken and Zilberstein, 2008]. Depending on the amount of interaction between agents and the observability of the global state, systems can be classified by different frameworks. A summary of all approaches described below can be found in [Pynadath and Tambe, 2002].

Let a *multi-agent decision process* be a tuple $\{N, S, A, O, p_T, p_O, R, B\}$ where:

- N is a set of agents indexed by i
- $S = \{s_1, s_2, \ldots\}$ is a set of global states
- $A_i = \{a_{i1}, a_{i2}, \ldots\}$ is a set of local actions available to agent i
- $A = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots\}$ is a set of joint actions with $A = A_1 \times A_2 \times \ldots \times A_n$
- $O_i = \{o_{i1}, o_{i2}, \ldots\}$ is a set of local observations available to agent i
- $O = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots\}$ is a set of joint observations with $O = O_1 \times O_2 \times \ldots O_n$
- $p_T : S \times S \times A \rightarrow [0, 1]$ is the joint transition probability function where $p_T(s_q|s_p, \boldsymbol{a}_k)$ is the probability for arriving in state s_q when taking action $\boldsymbol{a}_k$ in state s_p
- $p_O : O \times S \rightarrow [0, 1]$ is the observation probability function where $p_O(\boldsymbol{o}_k|s_l)$ is the probability of making observation $\boldsymbol{o}_k$ in state s_l
- $R : S \times A \rightarrow \mathbb{R}$ is the reward for taking action $\boldsymbol{a}_k$ in a state s_p
- $\boldsymbol{b} = (b^1, \ldots, b^n)$ is the vector of local belief states with $\boldsymbol{b} \in B$, the set of joint belief states

Note that the reward matrix is not agent-specific but rather common to all agents in the system. As a result the multi-agent decision process is a cooperative setting where agents have identical interests and benefit equally from choosing a certain action.

Let $\bar{o}_i$ denote a sequence of local observations. In this setting a local policy π_i is defined as a mapping from local observation histories to individual actions, $\pi_i : \bar{o}_i \rightarrow a_i$. As an agent's belief state is sufficient in summarising the observation histories, the local policy can also be defined as a mapping from local belief states to local actions, $\pi_i : b_i \rightarrow a_i$. Let a joint policy π either be a mapping from global states to joint actions or a mapping from local belief states to joint actions, $\pi : S \rightarrow A$ and $\pi : B \rightarrow A$ respectively. Which definition is used will depend on the observability of the global state and the exchange of information between agents. Again, the goal is to find the optimal joint policy, using the discounted sum of future rewards as a measure for optimality.

In general, agents' observations made at a given time will not all be the same and they will thus hold differing information about the system. This introduces a possibility of exchange of information between agents, which will depend on whether or not they can communicate. In addition to their observability these systems can therefore also be classified along their communication patterns. The limit cases commonly encountered are:

Multi-agent MDP (MMDP) [Boutilier, 1999] If agents have guaranteed and free communication among each other and the observation probability defines a one-to-one mapping from states to joint observations, the system is collectively observable and effectively reduces to a single-agent MDP. The problem simplifies to finding a joint policy π from global states to joint actions.

Multi-agent POMDP [Pynadath and Tambe, 2002] If agents have guaranteed and free communication and the observation probability defines a one-to-many mapping from states to joint observations, the system is collectively partially observable. This scenario is equivalent to a single-agent POMDP and the optimal policy is defined as a mapping from joint belief states to actions.

Decentralized MDP (dec-MDP) [Bernstein et al., 2002] If agents do not exchange their observations and the observation probability defines a one-to-one mapping from states to joint observations, the process is jointly observable but locally only partially observable. The aim is to find the optimal joint policy consisting of local policies $\pi = (\pi_1, \ldots, \pi_n)$.

Decentralized POMDP (dec-POMDP) [Bernstein et al., 2002] If agents do not exchange their observations and the observation probability defines a one-to-many mapping from states to joint observations, the process is both jointly and locally partially observable. As with the dec-MDP the problem lies in finding the optimal joint policy comprising local policies.

Note that the last two cases are not equivalent to multiple parallel single-agent POMDPs. Because the transition, observation and reward functions are joint between agents their local actions will in general affect not only their own dynamics but those of the other agents as well. Modelling the individual decision-problems as local POMDPs would not take these interdependencies into account.

Finding solutions for the first two cases is straightforward insomuch as they reduce to single-agent cases. Any of the solution algorithms discussed in Sections 3.3.2 and 3.3.3 may be applied, although the same limitations due to complexity apply (in particular in the POMDP case). Decentralized (PO)MDPs have been shown to be NEXP-complete[2] [Bernstein et al., 2002, Pynadath and Tambe, 2002], which makes them even harder to solve than single agent POMDPs. Finding approximate dec-(PO)MDP solutions is still a very active area of research, see e.g. [Amato et al., 2006, Amato et al., 2007, Bernstein, 2005, Boularias and Chaib-draa, 2008, Chechetka and Sycara, 2007, Emery-Montemerlo et al., 2004, Hansen, Eric A. and Bernstein, Daniel S.

[2]NEXP problems are those which can be solved by a nondeterministic Turing machine in $\mathcal{O}(2^{p(n)})$ time where $p(n)$ is a polynomial of n. An NEXP-complete problem is one for which every other NEXP problem has a polynomial time many-to-one reduction to it.

and Zilberstein, Shlomo, 2004,Nair et al., 2003,Oliehoek and Vlassis, 2007a,Oliehoek and Vlassis, 2007b,Oliehoek et al., 2008,Oliehoek et al., 2009,Roth et al., 2007,Seuken, 2007, Szer and Charpillet, 2006]

3.6 Game Theory

Multi-agent systems are often confused with or assumed to be identical to *game theory*. While the two approaches have many similarities their definitions and objectives have subtle differences. Game theory studies the interactions of self-interested decision-makers. As such its objective is different from that of the Artificial Intelligence community which aims to design decision-makers and their environment such that their interactions have the desired outcomes. Nonetheless, game theoretic concepts can be of great use when designing multi-agent systems and studying their characteristics.

In this thesis, the use of game theoretical methods will allow the statement of convergence results for one of the decision-making algorithms developed in Section 6.7. The current section will therefore provide a brief overview over basic terminology of game theory and review the methods needed within this work. For a comprehensive introduction to game theory, see for example [Fudenberg and Tirole, 1991]. A game-theoretic treatment of multi-agent systems is given in [Shoham and Leyton-Brown, 2009].

3.6.1 Basic Definitions

Let an action profile be a vector $\boldsymbol{a} = (a_1, \ldots, a_n)$ where agent i takes action a_i and $\boldsymbol{a}_{-i}$ denotes the actions chosen by all agents other than i. A *strategic game* is the very basic form of a game, which is defined as a tuple $g = N, A, u$, where:

- N is a finite set of self-interested *players*
- A is a set of joint *actions* $A = A_1, \times A_2 \times \ldots \times A_n$

- A_i is the set of individual actions available to agent i
- u is a *payoff function*, $u : A \rightarrow \mathbb{R}$ where $u(\boldsymbol{a}) = (u_1(\boldsymbol{a}), \ldots, u_n(\boldsymbol{a}))$ determines every player's preferences over action profiles $\boldsymbol{a}$

Agents are again assumed to act rationally, that is to choose the action which yields the highest payoff. In contrast to the multi-agent decision problem defined in Section 3.5, they will now in general have differing payoff functions (utilities). As a result they will in general have competing rather than cooperative preferences.

The action profile $\boldsymbol{a}^*$ is a *Nash equilibrium* if for every player i the following holds:

$$u_i(\boldsymbol{a}^*) \geq u_i(a_i, \boldsymbol{a}^*_{-i}) \; \forall a_i \tag{3.20}$$

A Nash equilibrium is therefore a steady state, at which no player has reason to change their action, because it is the best response to all other players' actions at the equilibrium. However, it need not be a state at which agents receive maximum joint payoff. This is because Nash equilibria are not always *Pareto optimal*, which denotes states in which no player can be made better off without making another player worse off.

As an example consider the prisoner's dilemma in figure 3.2. The unique Nash equilibrium in this game is for both players to defect, even though they could both receive higher payoff by cooperating. The visualisation of a game by its payoff matrix is referred to as the *normal form*.

Similar to the notion of policies introduced in the context of multi-agent systems, game theory defines the concept of a *strategy*. A deterministic strategy in which a player chooses his or her actions with certainty is referred to as a *pure strategy*. The collective single-action choices of all players in such a game are called a *pure strategy profile*. Players might also decide to choose randomly from their set of possible actions. Let $\Pi(X)$ denote the set of all probability distributions over set X. Then player i's

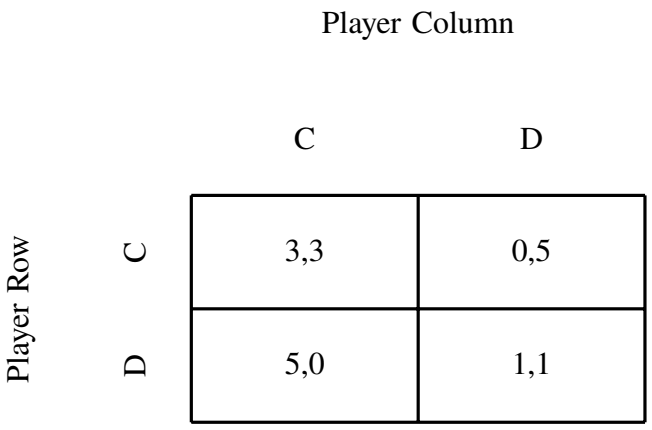


Figure 3.2: The prisoner's dilemma game. Two players, row and column, choose between defecting (D) and cooperating (C) actions. The numbers in the panels show the payoff gained by each player for the four possible action profiles. The first amount is the payoff obtained by the row player, the second the amount gained by the column player. The Nash equilibrium of this game lies in the lower right corner and is reached when both players choose to defect.

set of *mixed strategies* is given by $S_i = \Pi(A_i)$, where A_i is its set of possible actions. The *mixed strategy profile* is similarly used to refer to the collective stochastic action choices of all players.

3.6.2 Concepts Relevant to this Work

The strategic game introduced in the previous section does not have a notion of time attached to it: in its very basic form only one incident of this game is studied. Such games are therefore referred to as *single stage games*. Normal form games can in principle be used to describe games which are played more than once, but do not always lend themselves well to doing so, as they do not emphasise their temporal nature. The *extensive form* is an alternative representation of games with sequentiality which explicitly represents this aspect. An extensive form game with perfect information is defined as a tuple $g = N, A, q, u$ where

- N is a set of self-interested players

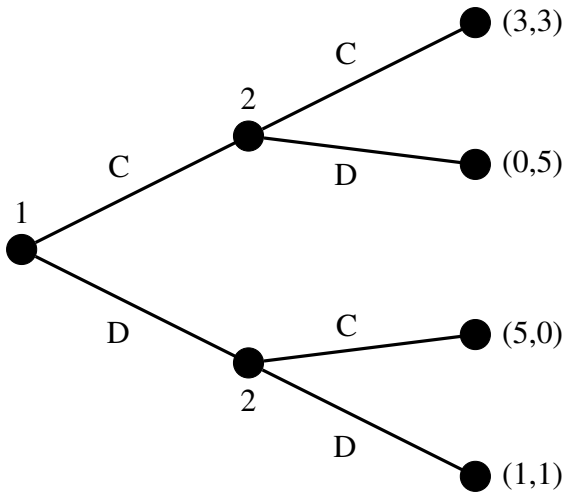


Figure 3.3: The extensive form representation of the prisoner's dilemma. Players are now labelled 1 and 2 rather than "row" and "column". Non-terminal graph nodes represent decision points of individual players in the game. Each path through the tree corresponds to one action sequence. The tuples at the terminal nodes give the payoff obtained by each of the players for the respective action sequence.

- A is a set of possible joint *action sequences*
- q is a *player function* that determines which of the players can choose an action at which point in the sequence
- u is the players' payoff functions for all possible action sequences

The extensive form can be visualised by a game tree. See figure 3.3 for an extensive form representation of the prisoner's dilemma. This representation suggests a sequentiality of action choice for the players in the game. Note however, that the extensive form can be used both for games with sequential as well as simultaneous action choices.

A *repeated game* is a special form of the extensive form game in which a particular single stage game is repeated for a finite or infinite number of steps. This allows the study of how players' action choices are affected by previous observation histories and their belief about their opponent's future actions. The single stage game which forms the basis of the repeated game is simply referred to as the game's *stage game*. The

concept of repeated games will be used to state convergence results about one of the decision-making algorithms presented in Chapter 6.

In an extensive form game, the best response for a player will in general depend on the particular point in the game. The concept of player's strategies therefore needs to be adapted to reflect the sequential nature of these games. In an $N-$player game, let $h = (a_{11}, a_{21}, \ldots, a_{N,1}, a_{12}, a_{22}, \ldots)$ be a finite or infinite action history of an extensive form game and let H be the set of all possible action histories. Then a strategy s_i for player i is a mapping from action histories to player i's possible actions that determines which action the player will choose at any point in the game: $s_i : H \rightarrow A_i$. Similar to the normal form game, a strategy profile s is given by all individual players' strategies, $s = (s_1, \ldots, s_N)$

The final framework to be considered here is that of *stochastic games*, which are a generalisation of both repeated games and multi-agent decision processes. A stochastic game is a tuple $\{G, p_T, R\}$ where:

- G is a set of stage games
- $p_T : G \times G \times A \rightarrow [0, 1]$ is the probability function which determines the transition between games
- $R = (R_1, \ldots, R_N)$ is a vector of individual payoff functions with $R_i : G \times A \rightarrow \mathbb{R}$.

Thus, a stochastic game which has only one stage game simplifies to a repeated game while a multi-agent decision process is a stochastic game with identical payoff functions for all players.

The convergence results for the decision-making algorithms developed in section 6.7 will rely heavily on existing convergence statements for repeated games. The next sections will therefore expand on those statements in detail.

3.6.3 Subjectivity and Equilibrium in Repeated Games

Consider n self-interested players, indexed by i, in a slightly modified version of an infinitely repeated game, g. Each player has a countable set of actions A_i, a countable set of possible consequences (outcomes) C_i, a utility function $u_i : A_i \times C_i \to \mathbb{R}$ and a discount parameter γ_i. The use of a set of consequences can be motivated as follows. An agents' goal is to choose a sequence of actions which maximises the current value of its expected utility. This will in turn depend on the possible consequences that might follow each action. Which consequences are possible will depend on two factors: the stochasticity (nature) of the game and the actions chosen by the other agents during the same round. Let t denote the current time index. Based on the joint vector of actions at each time-step, nature chooses a consequence vector $\boldsymbol{c}^t = (c_1^t, \ldots, c_n^t)$. Each agent is reported its individual consequence and collects the payoff $u_i(a_i^t, c_i^t)$. It then proceeds to choose a_i^{t+1}, the next action.

Let $z = (\boldsymbol{a}^1, \boldsymbol{c}^1, \ldots)$ be an infinite play path and let $\boldsymbol{h}^t = (\boldsymbol{a}^1, \boldsymbol{c}^1, \ldots, \boldsymbol{a}^t, \boldsymbol{c}^t)$ be a finite history of length t. Define $\boldsymbol{h}_i^t = (a_i^1, c_i^1, \ldots, a_i^t, c_i^t)$ to be its projection onto an individual agent. Let an agent's mixed strategy be defined as the probability of choosing an action at time $t+1$ given its individual history up to time t: $\sigma_i^{t+1} = p(a_i^{t+1}|\boldsymbol{h}_i^t)$ and let $\boldsymbol{\sigma}_i^t$ denote the action choices made up to (and including) time t. The joint choices of action can then be described in the same way: Let $\sigma^{t+1} = (\sigma_1^{t+1}, \ldots, \sigma_n^{t+1})$ be the joint probability of actions at time $t+1$ and let $\boldsymbol{\sigma}^t$ denote all action choices up to time t.

To construct the probability over histories assume the empty history $\boldsymbol{h}^0$ has probability one. Then the joint probability of observing history $\boldsymbol{h}^t$, followed by actions $\boldsymbol{a}^{t+1}$ and consequences $\boldsymbol{c}^{t+1}$ can iteratively be constructed as

$$p(\boldsymbol{h}^t, \boldsymbol{a}^{t+1}, \boldsymbol{c}^{t+1}) = p(\boldsymbol{h}^t|\boldsymbol{\sigma}^t) \cdot \prod_i \sigma_i^{t+1} \cdot p(\boldsymbol{c}^{t+1}|\boldsymbol{a}^{t+1}) \tag{3.21}$$

To determine its strategy at a given time-step, an agent should use this probability to reason about future consequences and rewards. But in general, the other agent's strategies will not be known to it, making such straightforward computation impossible. The agent must therefore maintain a belief over others' strategies and choose local actions based on the expectation values that result from this belief. However, the only variable which is ultimately of interest to an agent is its expected utility which depends on others' actions through the possible consequences. Rather than reason explicitly about others' strategies, an agent can therefore reason directly over possible consequences and base its choice of actions on that belief instead.

The joint influence of nature's choice and the other players' strategies on agent i's outcome of consequences can be summarised by an *environment response function*

$$e_i^{t+1} = p(c_i^{t+1}|\boldsymbol{h}_i^t, a_i^{t+1}) \tag{3.22}$$

which gives the probability of obtaining consequence c_i^{t+1} after observing history $\boldsymbol{h}_i^t$ and taking action a_i^{t+1}. If σ_{-i}^{t+1}, the remaining agents' strategies at this time, are known it is easy to construct the induced environment response function e_i (assuming the stochasticity of nature's choice is also known). If this is not the case, each agent can only hold a personal belief about its true environment response function. This belief is referred to as the *subjective environment response function* $\bar{e}_i$ which gives the subjective probability which agent i assigns to the possible consequences (and which need not be the same as the true probability over consequences):

$$\bar{e}_i^{t+1} = p_i(c_i^{t+1}|\boldsymbol{h}_i^t, a_i^{t+1}) \tag{3.23}$$

This belief is updated according to Bayes' Rule whenever the agent obtains new information.

Throughout the game, all agents will then base their choice of action on their subjective environment response functions, which induces a joint strategy vector $\boldsymbol{\sigma}$.

This in turn induces the objective (true) environment response functions.

In general the subjective strategies chosen by agents will be correlated, due to the interdependence of consequences. This allows the use of a *correlated equilibrium* [Aumann, 1974] in describing the system, a solution concept which is more general than the Nash equilibrium described in section 3.6.1 and can lead to higher overall rewards. To this end a correlation device is introduced to the game: Let $M = M_1 \times \ldots \times M_n$ be a set of possible messages to the agents in the game and let $p(M)$ be a probability distribution over these messages. Replace the empty history $\boldsymbol{h}^0$ with a message $\boldsymbol{m}$ drawn from $p(M)$, communicate to each agent its private "share" $m_i \in M_i$ of $\boldsymbol{m}$ and let the initial choice of strategy depend on this message: $\sigma_i^1 = p(a_i^1|m_i)$, then the resulting play of agents will be correlated.

Using the concepts defined above an agent's induced subjective probability over its own play paths can be expressed as

$$p(\boldsymbol{h}_i^{t+1}|\sigma_i, \bar{e}_i) = p(\boldsymbol{h}_i^t|\sigma_i, \bar{e}_i)\, \sigma_i^{t+1}\, \bar{e}_i^{t+1} \tag{3.24}$$

where $p(\boldsymbol{h}_i^0) = p(m_i)$. The true (objective) probability distribution over play paths is obtained in the same way, by replacing $\bar{e}_i^{t+1}$ with e_i^{t+1}.

Agent i's expected utility for a given local infinite play path z_i is defined as

$$u_i(z_i) = \sum_t \gamma_i^{t-1}\, u(a_i^t, c_i^t) \tag{3.25}$$

As a function of its environment response function and strategy this gives

$$u_i(\sigma_i^t, e_i^t) = \int u_i(z_i)\, p(z_i|\sigma_i^t, e_i^t)\, dz_i \tag{3.26}$$

With these definitions the criteria for an equilibrium of the repeated game can be formulated:

Let g be an infinitely repeated game and let (M, p) be a correlation device as described above. Let $\sigma = (\sigma_1, \ldots, \sigma_n)$ be a vector of strategies and let $\bar{\boldsymbol{e}} = (\bar{e}_1, \ldots, \bar{e}_n)$

be a vector of subjective environment response functions. Then the game is in a *subjective correlated equilibrium* if for each player i and each possible message m_i the following two conditions hold:

1. Subjective optimisation: σ_i is optimal with respect to $\bar{e}_i$, that is $\sigma_i = \arg\max_{\sigma_i} u_i(\sigma_i, e_i)$
2. Correlated uncontradicted beliefs: $p(z_i|\sigma_i, \bar{e}_i) = p(z_i|\sigma_i, e_i)$

The agents in the game described above will in general not converge to stationary strategies. It is therefore necessary to expand the equilibrium concept, which will require the concept of *closeness* of two probability distributions: For a given $\epsilon > 0$ and two probability distributions p and q, let q be ϵ-close to p if for any event x, $|p(x) - q(x)| \leq \epsilon$.

Let (M, p, g) be a correlated game as described above, with a vector of correlated strategies $\boldsymbol{\sigma}$ and a vector of correlated subjective environment response functions $\bar{\boldsymbol{e}}$ and $\epsilon > 0$. Then the game is in a *subjective correlated ϵ-equilibrium* if for every player i and possible message m_i the following two conditions hold:

1. Subjective optimisation: σ_i is optimal with respect to $\bar{e}_i$
2. Correlated ϵ-uncontradicted beliefs: with probability q greater than $1 - \epsilon$, a message vector m will be chosen with $p(z_i|\sigma_i, \bar{e}_i)$ ϵ-close to $p(z_i|\sigma_i, e_i)$

Kalai and Lehrer [Kalai and Lehrer, 1995] show that for rational agents and subject to certain conditions the repeated game described above converges to a subjective correlated ϵ-equilibrium. Stating the result requires the concept of *compatibility*:

The evolution described by $(\sigma_i, \bar{e}_i)$ is said to be compatible with the one generated by (σ_i, e_i) if the underlying probability distribution $p(z_i|\sigma_i, e_i)$ is absolutely continuous with respect to $p(z_i|\sigma_i, \bar{e}_i)$, $p(z_i|\sigma_i, \bar{e}_i) \gg p(z_i|\sigma_i, e_i)$. This means that for every event z_i,

$$p(z_i|\sigma_i, e_i) > 0 \Rightarrow p(z_i|\sigma_i, \bar{e}_i) > 0 \tag{3.27}$$

In this setting, compatibility ensures that events considered impossible under an agent's current belief really are impossible. Colloquially, this can be referred to as $p(z_i|\sigma_i, \bar{e_i})$ having a grain of truth.

Finally, the convergence result given by [Kalai and Lehrer, 1995] for the correlated game defined above can be stated: If $p(z_i|\sigma_i, \bar{e_i})$ is compatible with $p(z_i|\sigma_i, e_i)$ then for every $\epsilon > 0$ there is a time T such that at all times $t \geq T$ the players are in a subjective correlated ϵ-equilibrium. This convergence result is based on the concept of *merging of opinions*, [Blackwell and Dubins, 1962, Kalai and Lehrer, 1994] which will be outlined in the following section.

3.6.4 Merging of Opinions

Let $(\Omega, \mathcal{B})$ be a measurable space of outcomes. This means that Ω is a set of outcomes and $\mathcal{B}$ is a σ-algebra on Ω, which ensures that probability distributions can be defined over Ω and subsets of it. Let μ and $\tilde{\mu}$ be two probability distributions on $(\Omega, \mathcal{B})$ and assume that μ is the true distribution over outcomes while $\tilde{\mu}$ denotes some subjective belief over outcomes held by a rational agent. Let a *partition* of a set X be a collection of subsets of X which are both mutually exclusive and collectively exhaustive. Let $\{\mathcal{P}_t\}$ be a sequence of finite or countable partitions of Ω, where $\mathcal{P}_t$ can be interpreted as the information which is available to an agent at time t. Assume that if at time t, the true state is $\omega \in \Omega$, the agent is told $P_t(\omega)$, the element (i.e. subset) of $\mathcal{P}_t$, which contains ω. Let the partition P of a set X be a *refinement* of the partition Q of X if every element of P is a subset of an element of Q. Assume that the following restrictions on the partition sequence $\{\mathcal{P}_t\}$ hold:

1. $\mathcal{P}_{t+1}$ refines $\mathcal{P}_t$. This ensures that information is always cumulative

2. Let $\mathcal{F}_t$ denote the $\sigma-$algebra generated by $\mathcal{P}_t$ and let $\mathcal{F}$ be the $\sigma-$algebra generated by all $\mathcal{F}_t$. Then require $\mathcal{F} = \mathcal{B}$. This means that the events which are

defined under the information sequence are those which are defined under the measurable space

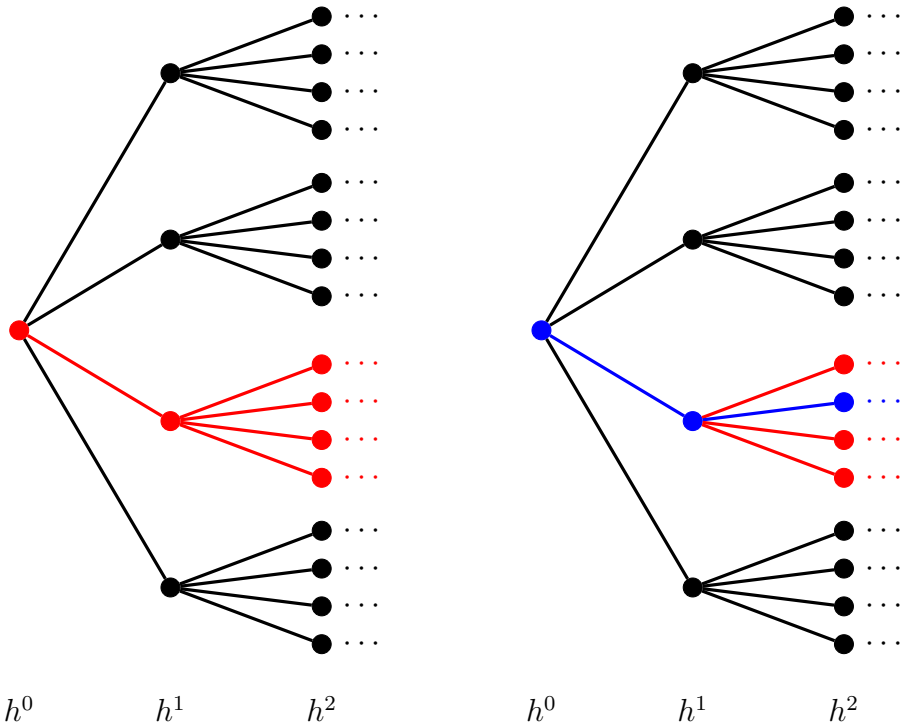
With the information gained at time t the agent can update its belief over outcomes, to be the posterior distribution $\tilde{\mu}(\cdot|P_t(\omega))$. The true distribution on the other hand will be given by $\mu(\cdot|P_t(\omega))$.

In this setting $\tilde{\mu}$ merges to μ if for every $\epsilon > 0$, information sequence $\mathcal{P}_t$ and almost every ω there is a time $T(\epsilon, \omega)$ such that for every time $t \geq T(\epsilon, \omega)$, $\tilde{\mu}(\cdot|P_t(\omega))$ is ϵ-close to $\mu(\cdot|P_t(\omega))$.

Blackwell and Dubins [Blackwell and Dubins, 1962] showed that given the assumptions above the following convergence result holds: $\tilde{\mu}$ merges to μ in the information sequence $\{\mathcal{P}_t\}$ if μ is absolutely continuous with respect to $\tilde{\mu}$: $\tilde{\mu} \gg \mu$.

3.6.5 Merging of Opinions in Repeated Games

The concepts introduced in the previous section can be applied to the repeated game described in section 3.6.3. Here, Ω is the set of infinite local (i.e. agent-specific) play paths. $\tilde{\mu}$ and μ are the approximate and true agent-based probability distributions over infinite play paths. These are induced by the subjective and objective environment response functions $\bar{e}_i$ and e_i respectively. A state ω is an infinite play path z_i and at time t, $P_t(\omega)$ describes the set of possible infinite play paths, defined by the history $h^t = (a_i^1, c_i^1, \ldots)$ while $\{\mathcal{P}_t\}$ describes the partitioning of the set of all possible play paths given the observed history. See figure 3.4 for an example.



(a) Two partitions after observing h^1 (b) Three partitions after observing h^2

Figure 3.4: Partitioning of the set of infinite play paths in the repeated game assuming four possible action-consequence pairs at every time-step. Different colours mark the paths which define the partitions given the current history: black for the set of infinite play paths possible under h^0, red for the set of infinite play paths possible under h^1 and blue for those possible under h^2.

Chapter 4

Agent Communication

4.1 Introduction

The previous chapter gave an overview of multi-agent systems and how decision-making can be realised within them. Decision-making in these systems is closely linked to the topic of agent communication, as communication determines how much information is exchanged between agents. The amount of information available to an individual agent will in turn influence the decision-making process. This chapter will describe the particular decision-making scenarios considered in this work and introduce the general approach taken towards solving them. It will begin by describing the problem setting and the way in which communication and decision-making are commonly designed within it. The strengths and weaknesses of this method will be highlighted. Subsequently the alternative approach proposed in this thesis based on individual local decision-making will be introduced.

4.2 Communication in Multi-Agent Decision Processes

The setting within which decision-making and communication will be studied here is the multi-agent decision process described in Section 3.5. Recall that this framework describes a set of cooperative agents with local observations and actions in a stochastic

environment. The aim is to find a joint policy which maximises the discounted sum of expected future rewards. As described in the previous chapter, most instances of this decision-problem assume either full synchronisation or no communication at all between agents. In the full-communication case, in which agent communication is assumed to be free and guaranteed, the decision-making problem reduces to that of a single agent. In the zero-communication case agents exchange no information between each other and act locally based on a pre-computed decentralized policy. Solving this decentralized decision-problem has significantly higher computational complexity (NEXP-complete) than the centralised case (P-complete or PSPACE-complete depending on the observability) [Pynadath and Tambe, 2002]. However, full and zero communication are only the limit cases on a whole spectrum of possible communication patterns. It is easy to imagine scenarios in which a formal description of partial communication would be desirable. Communication might come at a cost, creating a need to balance the value of increased information against the cost of obtaining it. Communication channels might be insecure or unreliable, causing some communication attempts among agents to fail. There is a wealth of realistic settings which fall somewhere between the centralised and decentralized case and some work has focused on scenarios with more flexible amounts of communication. In these treatments the system is usually modelled as alternating between time-steps of agent synchronisation (full communication) and episodes of zero communication. Such studies have been carried out with different motivations. Some employ communication as a means to reduce the complexity of computing a policy for the decentralized case [Goldman and Zilberstein, 2008, Nair et al., 2004]. Others aim to reduce the communication overhead inherent in full centralisation by avoiding redundant communications [Roth et al., 2005a, Roth et al., 2005b]. Sparse communication has also been studied from the viewpoint of finding a (near-)optimal communication policy in scenarios where

communication comes at a cost [Becker et al., 2005, Xuan et al., 2001]. Some of these approaches make limiting assumptions about the multi-agent system at hand, such as transition independence (which requires agents to have individual transition functions that are independent of one another). The focus in most of this work lies on deciding when to communicate, by pre-computing communication policies or by developing algorithms for on-line reasoning about when or what to communicate. Such treatment is valuable for scenarios in which inter-agent communication incurs some cost but is otherwise reliably available as it ensures that communication is only carried out when needed.

4.3 Unforeseen Sparse Communication

Apart from settings in which reducing communication between agents is a choice, some systems will impose involuntary communication restrictions on agents. If the nature of such restrictions is not deterministic or the time of their occurrence known beforehand, the agents again face a complex decision-making problem. But it is fundamentally different from the problem setting considered so far: the question then is not when or what to communicate but how to choose an action when communication is unexpectedly unavailable. Such a situation could arise in many different scenarios. Agents might be situated in a hostile environment where monitoring from opposing agents will make communication too risky at times. The communication channel between agents might be unreliable, such as a network connection which has recurring downtime. Or agents might have a limited transmission range allowing them to communicate only with those currently close enough to receive transmissions. Many such settings are conceivable in which communication might be unavailable for all agents or could just be missing between a few individuals in a larger system.

In more general terms these situations can be described as settings in which vary-

ing amounts of information exchange are available between agents over time, without the exact pattern being known beforehand. When agents have access to differing information, their local beliefs about the world will in general also differ. The question at hand is therefore what constitutes good decision-making in such scenarios and how it can be implemented. Ideally, agents should be able to make use of communication whenever it is available and yet be capable of making robust local action choices when it is not or only partially possible. Again the aim is to find the best joint action given the situation faced, where the best action is defined as the one which maximises overall expected future rewards.

Before embarking on the questions outlined above it is worthwhile to look at alternative approaches to solving this problem. One strategy could be to make up for the loss of information, which might be achieved by making the exchange of information between agents possible in a way that differs from communication. This approach has for example been used in methods of *plan recognition* [Huber and Durfee, 1993, Huber and Durfee, 1995] which are based on observing an agent's action and inferring its long-term goal or plan from them rather than communicating them directly. A more game-theoretic approach to the same scenario is presented in [Genesereth et al., 1986, Rosenschein and Genesereth, 1985].

While such an approach conveniently makes up for missing communication its downside lies in imposing a new assumption about availability of information (based on observations) which may not hold in all cases. Essentially, this approach shifts the problem of incomplete information due to communication failure to the problem of incomplete information from observational failures. The constraints which cause sparse communication may well apply to the availability of observing others' actions as well. Consider for example an agent trying to communicate with its counterpart which is sitting behind a wall blocking their communication channel. In such an easily

conceivable scenario an agent will not be able to substitute action observations for missing communication and hence must rely on a more versatile strategy.

Substituting observations for communication also limits the scope of possible systems to those in which the concept of observing another's action is meaningfully defined. Multi-agent systems need not comprise physical agents and in many scenarios which can abstractly be described by a multi-agent framework agents will not be able to observe each others' actions (e.g. in a system of software agents).

Another possibility of dealing with missing communication is complete decentralization: rather than rely on communication being available when it is not, agents could simply act according to a (near-)optimal decentralized policy at all times. This has the advantage of making the agents truly autonomous and very robust. On the other hand it unnecessarily wastes any possibilities agents have of sharing information by making agents completely isolated from one another.

Little work has been done on unpredicted sparse communication in the setting of multi-agent decision problems so far. One approach addresses stochastic communication delays in centralised POMDPs [Spaan et al., 2008]. However, it does not present a very general treatment, as the method focuses on agent systems which have just a small number of successive time-steps without any communication between steps of full synchronisation. Some work has been done on local decision-making in the context of dec-POMDPS, where approximate local decision-making can be used to speed up policy computation [Emery-Montemerlo et al., 2004, Roth et al., 2005a, Roth et al., 2005b]. As such these methods focus on how agents should choose local actions when there is no communication at all and do not lend themselves well to include situations with partial communication.

Settings in which agents have differing amounts of information about the environment pose a fundamental problem to decision-making, in particular in cooperative

scenarios. The aim remains for agents to act coordinately in order to achieve the highest possible joint reward. But if some agents now have more information about the environment than others, and might not even know which of the others holds how much information, it is not immediately clear how they go about choosing their best joint action. If agents are in a setting where each participant is assigned its own convex sub-problem to the overall task, differing amounts of information about the world will not pose a problem. In such a case each agent can tackle its local problem independent of the action choices of others. In a general setting where agents' tasks are inter-related and the rewards assigned to individual actors depend on a joint choice of action this will no longer be the case. Here, action choices based on differing beliefs about the world are not necessarily coordinated in a sense that they are made under full knowledge of others' choices of actions at the same time. The only way to ensure that agents choose such a coordinated action is for them to have the same beliefs about the world. This would mean that agents would have to discard any information which gives them an "advantage" over others, i.e. which would affect the others' belief about the global state if it were communicated. But that is an unsatisfactory solution, as it requires agents to ignore what might be valuable information. On the other hand, if agents use all information that is available to them locally, they will act as individually more informed decision-makers but might end up with an action which is globally much worse than what they could have achieved coordinately. Very fundamentally there is a trade-off to be made between the information gained from knowing that all decision-makers are equally ignorant and will choose actions in a coordinated fashion and the information gained from local observations or partial communications. It is not immediately clear how the decision between these two possibilities is to be made.

In all of the work cited above, guaranteed coordination is favoured over the use of locally available information. This choice is usually made from first principles without

elaborating why or weighing the two possibilities off against each other. However, anecdotal proof exists that using locally available information can improve the performance of an existing algorithm in the context of dec-POMDPs [Chechetka and Sycara, 2007].

4.4 Approach Taken Here

None of the approaches described so far provides a general framework for decision-making in agent settings with varying information exchange. In particular there is further need to investigate the trade-off between agent coordination and local information in such scenarios. The approach taken in this work therefore differs in several ways from the work on sparse communication outlined above:

Firstly, a different stance is taken on the issue of agent coordination vs. the use of local information. From an information-theoretic standpoint it is unsatisfactory to throw away local information along the decision-making process even if it is to ensure overall coordination. The working hypothesis here is that there are many scenarios in which using all locally available information will lead to better overall performance because agents make more well-informed decisions. This could mean that in many cases local decision-making is to be favoured over guaranteed coordination and not the other way around.

Secondly, the approach taken here is intended to be as general as possible. This means in particular that not only full and zero communication cases are to be studied, but also settings with more complex communication patterns. These could for example include individual communication links between any two agents or communication with a nearest neighbour who is unknown at prior time-steps. In addition, as few assumptions as possible are made about the structure of the scenarios, for example no transition independence is necessary.

Lastly, the aim of this work is to develop scalable algorithms. Many of the current dec-(PO)MDP solutions still do not scale to systems with more than very few agents, observations and states. They are therefore very limited in their applications. Rather than aim to develop near-optimal algorithms which have slightly lower complexity than the exhaustive solution, the approach taken here is to obtain as good a performance as possible at much lower computation cost.

The overall aim is therefore to develop good approximate algorithms for local decision making in the agent system described.

4.5 Exact Treatment of Sparse Communication

To explore the possibilities of decision-making based on local information consider first the general, exact case. Assume a set of agents indexed by i, which are acting in a decision process with known dynamics similar to those described in chapter 3. At any given point in time all agents will hold some amount of information about the state of the system. Depending on whether any communication has taken place, this might be the full amount of information available at that time, only the amount obtainable from local observations or some amount in between. Let $I_i = I_{i,obs} + I_{i,comm}$ denote the information which is locally available to agent i. This information can be used to successively update the agent's personal belief b_i over the global state:

$$b_i(s|I_i) = \frac{p(I_i|s) \cdot b_i(s)}{p(I_i)} \tag{4.1}$$

Where $b_i(s)$ is the personal belief held before obtaining information I_i and the likelihood $p(I_i|s)$ depends on the probability of local observations as well as the probability of communications between agents. In general the information I_i held locally will differ between agents and as a result the agents will hold differing local beliefs b_i about the global state of the system. Assume, without loss of generality, that some prior knowledge $\mathcal{I}$, about how much information agents are expected to have access to during a

course of a scenario, is available. Let $\boldsymbol{b} = (b_1, \ldots, b_n)$ and $\pi(\boldsymbol{b}) = (\pi_1(b_1), \ldots, \pi_n(b_n))$ denote the vectors comprising all local agent beliefs over states and the local agent policies respectively. Then the expected future reward in a given state under this mixed joint belief is given by

$$V_\pi(s, \boldsymbol{b}) = R(s, \pi(\boldsymbol{b})) + \gamma \sum_{s'} p(s'|s, \pi(\boldsymbol{b})) \sum_{\boldsymbol{b}'} p(\boldsymbol{b}'|\boldsymbol{b}, s, \mathcal{I}) V_\pi(s', \boldsymbol{b}') \tag{4.2}$$

and the expected future reward given only the belief state is given by

$$V_\pi(\boldsymbol{b}) = \sum_{s} p(s|\boldsymbol{b}) \left\{ R(s, \pi(\boldsymbol{b})) + \gamma \sum_{\boldsymbol{b}'} p(\boldsymbol{b}'|\boldsymbol{b}, \mathcal{I}) V_\pi(\boldsymbol{b}') \right\} \tag{4.3}$$

Note that $p(\boldsymbol{b}'|\boldsymbol{b}, s, \mathcal{I})$ allows assigning probabilities to all possible future belief states, according to the prior knowledge about information exchange between agents. In the case of full communication for example, $p(\boldsymbol{b}'|\boldsymbol{b}, s, \mathcal{I}) = 0$ for all $\boldsymbol{b}'$ for which $b'_i \neq b'_j$. Notice also that this formulation can in principle encompass any additional sources of uncertainty (e.g. observation noise) which might affect an agent's belief state. These influences can be incorporated into the distribution over future beliefs and thus need not be considered separately.

In theory, the way to determine the best joint policy is by finding the combination of local policies that maximise equation (4.3). It can easily be seen however, that this equation is highly intractable: in the limit of no communication it will reduce to the standard equation for decentralized decision processes. These have been shown to be NEXP-complete [Bernstein et al., 2002]. Considering not only the zero-communication case but all possible scenarios of partial (and full) communication that might occur at future time-steps will only increase the computational complexity of finding a solution. The complexity of Equation (4.3) is therefore lower bounded by the complexity of solving the decentralized case. (This is assuming that it is known that full communication will not be the case. Solving for the centralised scenario has lower computational complexity, namely that of solving a single-agent MDP or POMDP, which are P-complete

and PSPACE-complete respectively [Pynadath and Tambe, 2002].)

As the search for the optimal combination of local policies cannot be solved exhaustively, the challenge lies in finding suitable approximate solutions. The remainder of this thesis will explore several possibilities for such approximate algorithms which allow for local agent decision making based on individual belief states.

4.6 Towards Local Decision-Making

Equation (4.3) provides an exact but intractable formulation of the value function of a multi-agent system in which agents have varying amounts of information available locally. One cause for its complexity is the centralised nature of this approach: it requires enumerating all possible combinations of local beliefs and local policies and calculating the expected discounted reward under each of them. If it were instead possible to obtain the optimal combination of local policies in a decentralized fashion, that is for each agent locally, the computation could be simplified. In general this will not be possible, as a globally optimal solution is not always obtainable from local computations. However, the analysis in the following chapters will show, that this can provide a very effective approach in practise.

The value function given by Equation (4.3) is calculated in a centralized fashion for a setting in which agents hold differing beliefs b_i about the global state and act upon these local beliefs. Instead, consider calculating a local solution π_i to $V_{\pi_i}(b_i)$, the local expected reward given agent i's belief over the global state:

$$V_{\pi_i}(b_i) = \sum_{s} p(s|b_i) \sum_{\boldsymbol{b}_{-i}} p(\boldsymbol{b}_{-i}|b_i) \sum_{\pi_{-i}} p(\pi_{-i}|\pi_i, \boldsymbol{b_{-i}}) \cdot \{R(s, \pi(\boldsymbol{b})) + \gamma \sum_{b_i'} p(b_i'|\boldsymbol{b}, s, \mathcal{I}) V_{\pi}(b_i')\} \tag{4.4}$$

where $\boldsymbol{b} = (b_1, \ldots, b_n)$ is the vector comprising local beliefs, $\pi(\boldsymbol{b}) = (\pi_1(b_1), \ldots, \pi_n(b_n))$ is the joint policy vector and $\boldsymbol{b}_{-i}$ and π_{-i} are the respective joint vectors without the

ith component. In addition to marginalising over the global state, a summation over other agents' possible beliefs and policies is also necessary. This computation requires the probability of other agents' beliefs given agent i's belief, $p(\boldsymbol{b}_{-i}|b_i)$, as well as the probability of others' policies given their belief and agent i's policy, $p(\pi_{-i}|\pi_i, \boldsymbol{b_{-i}})$. With this the total reward under policy π_i as expected by agent i is given by

$$\begin{aligned} V_{\pi_i} &= \sum_{b_i} p(b_i) V_{\pi_i}(b_i) \\ &= \sum_{b_i} \sum_{s} p(s) p(b_i|s) V_{\pi_i}(b_i) \end{aligned} \tag{4.5}$$

Calculating the value function in Equation (4.4) requires marginalising over all possible belief states and policies of other agents for an infinite time-horizon and will in general still be intractable. Facilitating this kind of localised decision-making therefore requires further approximations which allow computing a solution to this decision problem.

In particular, all possible belief states of all agents are considered in Equation (4.4). In reality, each agent will only encounter a fraction of the belief states that are in principle possible under the system's dynamics. One idea might therefore be to compute policies on-line rather than to pre-compute a mapping for all belief states.

To approximate (4.4), two problems must be addressed: firstly, the infinite horizon treatment of the setting is a cause for intractability and must somehow be simplified. In contrast to fully centralised or decentralized decision processes this also involves finding an approximation to the vast number of possible information exchanges that might occur at future times. Secondly, there is the challenge of approximating the marginalisation over all other agents' belief states and policies at the current time-step. The latter is in itself a complicated task even without taking the infinite time-horizon into account. The following chapters will therefore focus on approximations to the second problem in scenarios where sequentiality need not be considered. Understanding which factors influence the quality of such approximations in these simplified settings will then allow

to approach both problems together in Chapter 8.

Chapter 5

One-Step Scenarios

5.1 Introduction

To understand the dynamics and influencing factors of approximate local decision-making, consider the class of simplified one-step scenarios. While this is a strong restriction on the set of possible scenarios, it is one which allows focusing on studying the basic aspects of local decision-making. With a better understanding of these it will then be possible to extend the study to sequential scenarios.

The decision processes studied here will continue to have a probabilistic transition between successive states. However, in contrast to the general case introduced in Section 3.5, the probability of transitioning to a state s will be independent of the previous state and action. In particular, all states will be equally probable under the transition matrix. This effectively reduces the setting to a one-step scenario with a uniform prior distribution over possible states.

As previously described, communication will be assumed to be free but only available at some time-steps and possibly not between all agents. This will force agents to sometimes make decisions without exchanging information while at other times communication will be possible and allow more informed decision-making. Depending on the number of agents in such a system, different communication scenarios are possible: In the simplest case, the system will comprise two agents and successful communi-

cation will result in fully synchronised information about the world. Similarly, the failure of communication will mean that agents only have access to the information obtained from local observations. In a system with more than two agents, this alternation between being fully synchronised and fully local is again possible. In addition, any communication pattern which lies between full and zero communication is also feasible. For example, agents might have individual communication links which could fail independently, leaving them with partial information about others' observations.

The remainder of this chapter will outline three benchmark scenarios on which the approximate algorithms presented in the next chapter will be tested. It will also describe alternative solution algorithms to which their performance can be compared.

5.2 Benchmark Scenarios

The decision-making algorithms derived in the following chapter will be applied to three different one-step benchmark scenarios: a modified version of the *tiger problem*, a meeting problem as well as a team monitoring problem. Each of these scenarios captures different possible characteristics and challenges of a multi-agent system. As such they can provide both a motivation for the type of problems that might be addressed with flexible decision-making algorithms as well as a first test of their performance within well-defined settings.

5.2.1 The Tiger Problem

The first benchmark scenario is a modified version of the tiger problem. This setting was first introduced in [Kaelbling et al., 1998] in the context of single-agent POMDPs and has since been used in various forms as a benchmark problem for dec-POMDP solution techniques [Amato et al., 2007,Nair et al., 2003,Nair et al., 2004,Oliehoek and Vlassis, 2007b,Roth et al., 2005a,Roth et al., 2005b,Seuken, 2007,Szer and Charpillet,

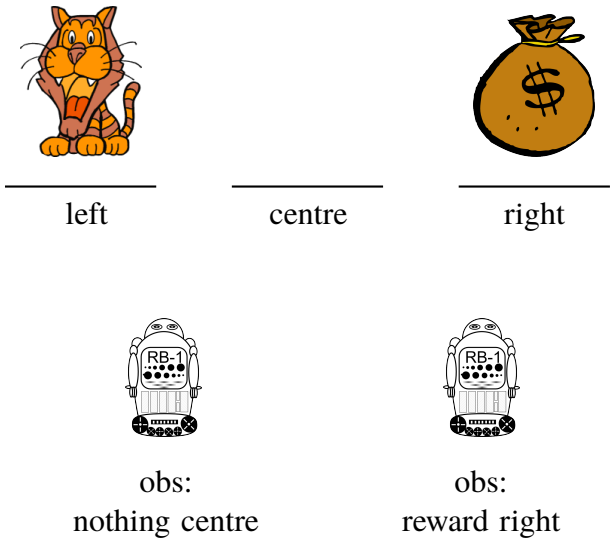


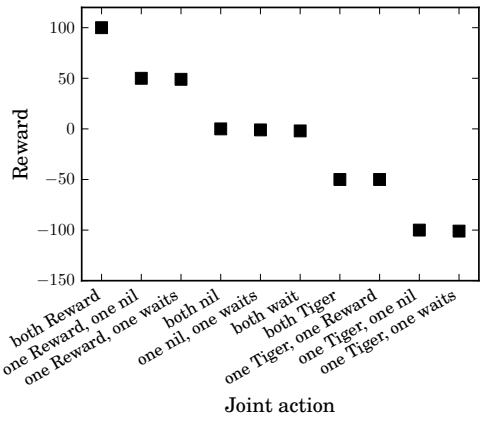
Figure 5.1: Visualisation of the tiger problem with two agents both choosing between three doors. In this example the first agent has observed that there is nothing behind the centre door while the second agent has observed the reward behind the right door.

2006]. For a comprehensive description of the initial multi-agent formulation of the problem see [Nair et al., 2003]. The two-agent scenario has been modified in the following way (see figure 5.1 for a visualisation): Two agents are faced with three doors, behind which sit a tiger, a reward or nothing. At each time step both agents must choose to open one of the doors or to pass their opportunity and wait. These actions are carried out deterministically and after both agents have chosen their actions, an identical reward is received depending on their joint action. The matrix which determines the reward payout is commonly known to both agents. After agents have received their reward the configuration behind the doors is randomly re-set to a new state. Prior to choosing an action the agents are both informed about the contents behind one of the doors, but never both about the same door. This somewhat constructed setup is interesting to study, for the following reason: if agents can exchange their observations before making their respective decisions, the problem becomes fully observable and the optimal choice of action for each agent is straightforward. If, on the other hand,

they cannot exchange their observations, they will hold incomplete information about the global state. The local observation received by each agent will enable it to form a belief over where the tiger and the reward are located, but because agents obtain different observations, these beliefs will differ between them.

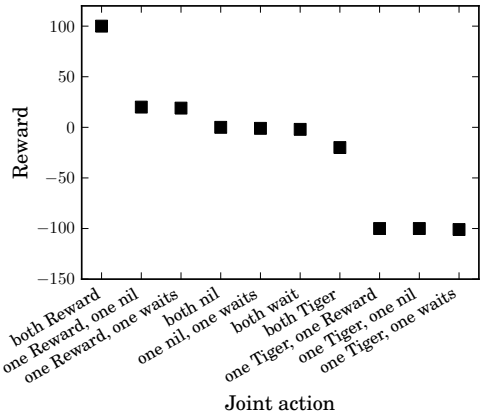
The tiger problem has been implemented with three different settings for the reward matrix. In all three cases the maximum reward/lowest penalty is awarded to coordinated actions: if agents both choose the door behind which lies the reward, they receive maximum payoff. Similarly, when they both choose the door which houses the tiger, they receive a reduced penalty, which reflects the assumption that together they are more well-equipped to fend off the tiger. The three reward settings differ in the amount of payoff received when both choose the tiger as well as the amount obtained when their actions are not coordinated. This could for example be the case when one agent chooses a good action (the door with the reward behind it) while the other agent chooses a lesser one (e.g. the door behind which lies the tiger). For such joint actions, different reward scenarios are possible: agents could receive some fraction of the payoff to reward them for getting one of the actions "right". They might receive a partial penalty to account for one of them having chosen a bad action or, in the limit case, they might obtain maximum penalty. Figure 5.2 summarises the different reward settings used in the tiger problem.

The tiger problem is one in which agents make local (i.e. partial) observations which are correlated and which thus provide information about which observation the other agent might have made. In addition, it is a scenario in which the optimal policy given both observations is symmetric: assuming the state were known the best local action for both agents would be to choose the door behind which the reward is located.



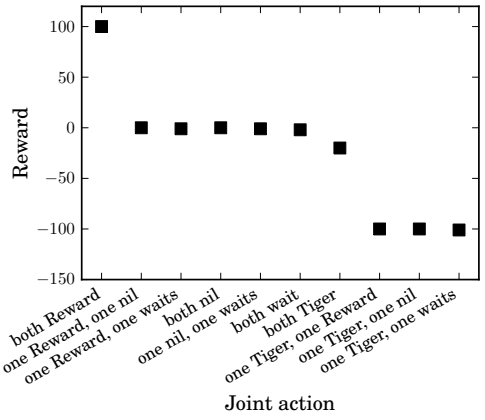
Actions	Rewards
both choose tiger	−50
both choose reward	100
both choose nil	0
both wait	−2
one tiger, one nil	−100
one tiger, one reward	−50
one tiger, one waits	−101
one nil, one waits	−1
one nil, one reward	50
one reward, one waits	49

(a) "Local good" setting: some reward for uncoordinated actions



Actions	Rewards
both choose tiger	−20
both choose reward	100
both choose nil	0
both wait	−2
one tiger, one nil	−100
one tiger, one reward	−100
one tiger, one waits	−101
one nil, one waits	−1
one nil, one reward	20
one reward, one waits	19

(b) "Similar" setting: very little reward for uncoordinated actions



Actions	Rewards
both choose tiger	−20
both choose reward	100
both choose nil	0
both wait	−2
one tiger, one nil	−100
one tiger, one reward	−100
one tiger, one waits	−101
one nil, one waits	−1
one nil, one reward	0
one reward, one waits	−1

(c) "Coordinated good" setting: penalty/no reward for uncoordinated actions

Figure 5.2: Reward settings used for the tiger problem, differing in the amount of payoff awarded for uncoordinated actions. In all cases the highest reward/lowest penalty was awarded for coordinated actions. The right-hand column shows the rewards in tabular form with differing regions shaded while the left-hand column shows the same settings as plots. The slope of the plot can intuitively be understood as a measure for how much uncoordinated actions are rewarded.

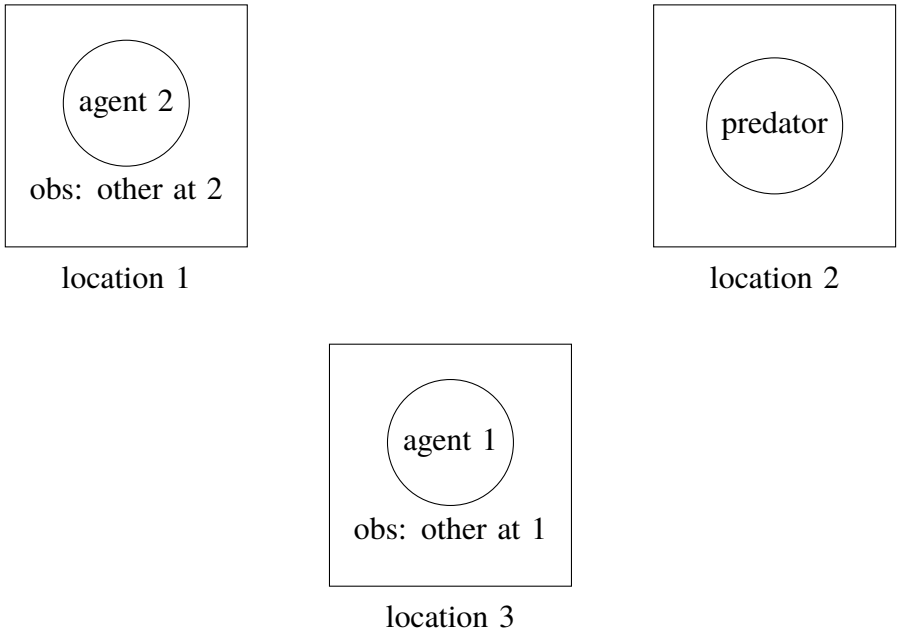
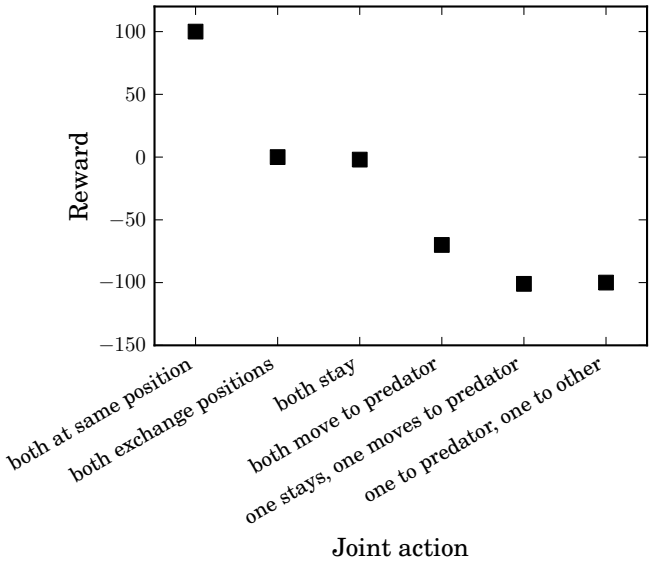


Figure 5.3: Visualisation of the meeting scenario with three locations occupied by two agents and a predator. In this example the agents are at locations 1 and 3. Agent 1 has observed agent 2 to be at location 2 although it is at location 3. Both agents know that their local observations might be wrong.

5.2.2 The Meeting Problem

The second benchmark scenario used, is one in which agents aim to meet in a common location. Consider three possible locations which are occupied by two friendly agents and a predator. At each time-step the friendly agents observe which site they are located at with certainty. In addition, they make a noisy observation about where the other friendly agent is located: with probability p the location observed is false. In this case the other agent is in truth located at the remaining possible location, while the position at which it is observed to be is occupied by the predator (see figure 5.3 for a visualisation). Agents can choose to move to one of the two other locations or to remain at their current position. These action choices are carried out deterministically. As with the tiger problem the scenario becomes fully observable if agents are able to exchange their observations (more precisely the information about their own location). After



(a) Plot representation of reward matrix

Actions	Rewards
One stays, other joins	100
Both exchange position	0
Both stay	−2
Both move to predator	−70
One stays, one moves to predator	−101
One moves to predator, one to others' position	−100

(b) Tabular representation of reward matrix

Figure 5.4: Reward settings for the meeting problem.The lower sub-figure shows the rewards in tabular form with differing regions shaded while the upper sub-figure shows the same settings as plots. The slope of the plot can intuitively be understood as a measure for how much uncoordinated actions are rewarded.

carrying out their action choices the scenario is randomly re-set to one of the possible configurations. Figure 5.4 shows the implemented reward matrix for all possible joint actions: a high reward is obtained if both agents end up at the same location and avoid the predator, but no reward or a penalty is received if agents act uncoordinatedly. Varying the amount of reward received for partially uncoordinated actions had very little effect in this scenario. Hence, only one reward setting was used. Note that in

contrast to the tiger problem this is a setting in which individual local observations are independent given the state. Another difference is that the optimal joint strategy given the global state is asymmetric: maximum reward is obtained when one agent remains in its place while the other moves to meet it.

Particular characteristics of the first two benchmark problems

As two-agent systems the first two benchmark problems presented here have the particular property of alternating between full synchronisation and no exchange of information at all. This means that in studying the performance of local decision-making algorithms only the zero-communication time-steps need to be considered: for those with full communication, the centralised policy can be used and the problem solution is straightforward. This also allows benchmarking the performance of local approximate algorithms against the performance of alternative solutions, such as a dec-(PO)MDP policy, as will be described below.

5.2.3 The Monitoring Problem

The third benchmark problem is a joint monitoring task: four agents are deployed to identify passing aerial objects, each of which could either be a friend, a civilian or an enemy. Each of the four agents can observe one characteristic of the target, such as its velocity, its build, its altitude or its equipment carried. No individual characteristic is sufficient for determining the target's identity. Instead, the occurrence likelihoods are commonly known to agents (e.g. the probability of observing a certain velocity if the target is a civilian) along with a prior distribution over target identities. In contrast to the two first applications this is a setting in which even with full exchange of information, the problem does not become jointly observable. At every time-step each agent must choose between three possible actions: attack, standby or send a friendly signal. Three different settings for the reward matrix have been implemented.

Again, the highest reward is given to coordinated actions: regardless of the identity of the target, the highest reward is obtained when three agents react to the target and one agent remains on standby (this can be interpreted as one agent staying alert in case additional information becomes available). If the target is a friend, three agents should send friendly messages while one is on standby. If it is a civilian all agents should remain on standby while with an enemy, three agents should attack while one remains on standby. For a full overview over the rewards used in each setting of this scenario see appendix B.

Because this scenario comprises more than two agents, different communication settings are possible. In addition to the scenario in which agents alternate between full and zero communication, intermediate communication patterns may also occur. These will be studied more closely in the next chapter.

5.3 Benchmark Algorithms

The benchmark problems described above will allow testing the performance of the decision-making algorithms developed in the following chapter. To do so two benchmark algorithms will also be implemented on the same problems. First, the reward obtained under the approximate algorithms for the zero-communication case can be compared to that of the fully decentralized algorithm. Secondly, results will be compared to those of a guaranteed coordinated approach similar to the one described in Section 4.3.

5.3.1 Fully Decentralized

The fully decentralized algorithm is a solution to the corresponding dec-MDP or dec-POMDP, in which it is assumed that no communication at all takes place between agents. As described in Section 3.5, numerous approximate solution techniques ex-

ist. The performance of this algorithm is of particular interest when compared to the first two scenarios described here (the tiger problem and the meeting problem): as two-agent systems they alternate between full agent synchronisation and zero communication. To measure the quality of an approximate algorithm only the time-steps at which no communication takes place need therefore be studied. In a one-step scenario this is equivalent to being in a dec-(PO)MDP and the fully decentralized solutions consequently gives an upper bound for the maximum reward which may be achieved by an approximate algorithm. Similarly, the algorithm provides a lower bound for the performance one would hope to achieve in settings with other communication patterns: if partial (but unplannable) communication is possible between agents, the approximate algorithms developed should perform better than the fully decentralized ones, as agents have greater or equal amounts of information available from others when making their action choices.

The exhaustive way of calculating the optimal decentralized algorithms would be to iterate over all possible combinations of local policies π_i and possible local observations o_i, choosing the combination of local policies which maximises the expected reward:

$$V_\pi = \sum_s p(s)R(s, (\pi_1(o_1), \ldots, \pi_n(o_n))) \tag{5.1}$$

where the discount factor has been chosen close enough to zero to consider only the current time-step. In practise this approach is prohibitively costly and fails to remain tractable for the benchmark scenarios presented here. Instead, a well-known approximate algorithm for finding decentralized policies has been used for this work. Joint equilibrium-based search for policies, JESP, [Nair et al., 2003] is a locally optimal solution algorithm for decentralized (PO)MDPs. The particular flavour used here is exhaustive JESP, which computes the best local policy for agent i while holding the policies of all other agents j fixed. Initial agent policies are drawn randomly and the best responses are calculated iteratively. The algorithm terminates when the policies

of all agents remain unchanged (i.e. when an equilibrium is reached). The search for agent i's best local policy is performed exhaustively and as a result the expected value under the joint policy is guaranteed to never decrease. Thus the algorithm will always terminate at a local maximum. To avoid getting trapped in bad local maxima, random restarts can be used to calculate different solutions to the search, choosing the one which yields the highest expected reward.

5.3.2 Guaranteed coordination

The second benchmark algorithm is a guaranteed coordinated approach, as described in Section 4.3. The aim of this algorithm is to guarantee that agents make coordinated action choices even when communication is unavailable. This will in general only be possible when all agents hold the same belief over the global state at the time of decision-making. Under synchronised beliefs each agent can determine the optimal joint action based on this belief and choose the local action which corresponds to that particular joint action. Since all agents hold the same belief, they will all carry out the same calculation in parallel. In contrast, agents which hold differing beliefs about the world will in general arrive at different answers as to which joint action is the optimal one to choose. Thus, to guarantee identical beliefs agents may not take local information into account which would alter their belief about the global state. This means that agents must ignore their local observation and use the commonly held prior distribution over states to calculate the optimal joint action by maximising the expected reward:

$$V(\boldsymbol{a}) = \sum_{s} p(s)R(s, \boldsymbol{a}) \tag{5.2}$$

As noted in Section 4.3, this guaranteed coordination comes at the cost of discarding potentially valuable information, thus resulting in a decision-making process which might be overall less informed. This is particularly pertinent in settings where individ-

ual communication links might fail while others remain available: under this approach all agents would be forced to discard their current knowledge of observations, even in the case where all agents but one have been able to exchange information.

5.4 Local decision-making in One-Step scenarios

Recall from equation (4.5) the formulation for a single agent i's expected reward under its belief state b_i, which was given by

$$V_{\pi_i}(b_i) = \sum_s p(s|b_i) \sum_{\boldsymbol{b}_{-i}} p(\boldsymbol{b}_{-i}|b_i) \sum_{\pi_{-i}} p(\pi_{-i}) \cdot \{R(s, \pi(\boldsymbol{b})) + \gamma \sum_{b_i'} p(b_i'|\boldsymbol{b}, s, \mathcal{I}) V_\pi(b_i')\} \tag{5.3}$$

In the case of a one-step scenario this equation simplifies to

$$V_{\pi_i}(b_i) = \sum_s p(s|b_i) \sum_{\boldsymbol{b}_{-i}} p(\boldsymbol{b}_{-i}|b_i) \sum_{\pi_{-i}} p(\pi_{-i}) R(s, \pi(\boldsymbol{b})) \tag{5.4}$$

Where the assumption has been that γ is sufficiently close to zero to ignore later time-steps. The optimal local policy for agent i is then given by

$$\pi_i^* = \arg\max_{\pi_i} V_{\pi_i}(b_i) \tag{5.5}$$

The following chapter will explore approximate solutions to equations (5.4) and (5.5).

Chapter 6

Approximate Algorithms for One-step Scenarios

6.1 Introduction

The previous chapters have set the framework for sparse communication considered in this work. This chapter will study the general characteristics of approximate local decision-making in these settings and then develop three algorithms which provide local policies in one-step scenarios. Each approximation has been applied to the benchmark problems described in chapter 5 and compared to the performance of the benchmark algorithms outlined there.

6.2 Modelling Others' Action Choices

Recall Equation (5.4) in chapter 5 which gives the local value function under an agent's local belief state in a one-step scenario:

$$V_{\pi_i}(b_i) = \sum_{s} p(s|b_i) \sum_{\boldsymbol{b}_{-i}} p(\boldsymbol{b}_{-i}|b_i) \sum_{\pi_{-i}} p(\pi_{-i}) R(s, \pi(\boldsymbol{b})) \tag{6.1}$$

The aim now is to find an approximate formulation of this equation which avoids a full marginalisation over all other agents' possible belief states and policies.

Note that the optimal local one-step policy of an individual agent is simply the best response to the possible local actions the other agents could be choosing at that

point in time. The full marginalisation over others' local beliefs and possible policies therefore amounts to a marginalisation over all actions possibly taken by other agents at that time-step. Calculating this requires knowledge of the probability distribution over the others' action choices, given agent i's local belief b_i. In general this probability will also depend on the current state. Let $\boldsymbol{a}_{-i}$ denote the joint action of all agents other than i. Assuming the joint distribution $p(s, \boldsymbol{a}_{-i}|b_i)$ over the current state and others' actions is known, Equation (6.1) can be reformulated as

$$V_i(a_i|b_i) = \sum_s \sum_{\boldsymbol{a}_{-i}} p(s, \boldsymbol{a}_{-i}|b_i) R(s, a_i, \boldsymbol{a}_{-i}) \tag{6.2}$$

This expression in terms of $p(s, \boldsymbol{a}_{-i}|b_i)$ reduces the computational complexity of finding the best local policy π_i: instead of marginalising over all local beliefs and policies of other agents for each belief state b_i the agent only needs to consider the set of possible joint actions of others. Its own best response is consequently given by

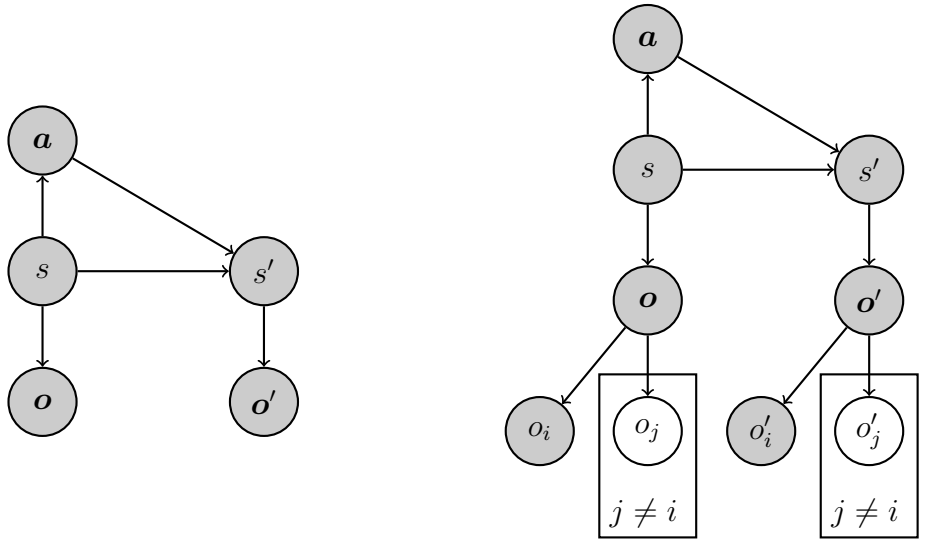
$$a_i^* = \arg\max_{a_i} [V_i(a_i|b_i)] \tag{6.3}$$

However, Equation (6.2) assumes that the joint distribution $p(s, \boldsymbol{a}_{-i}|b_i)$ is known. This is not the case and the analysis below will show that finding its exact form will again turn out to be intractable.

Agent i's belief over the current state and other agents' choice of actions can be expanded as

$$p(s, \boldsymbol{a}_{-i}|b_i) = p(\boldsymbol{a}_{-i}|s, b_i) p(s|b_i) = p(\boldsymbol{a}_{-i}|s, b_i) b_i(s) \tag{6.4}$$

which requires knowledge of the agent's belief over the global state b_i as well as the probability over the other agents' actions. Both of these will depend on how much information has been exchanged between agents. The following treatment will therefore first look at systems with synchronised communication and then extend to general communication patterns. In accordance with much of literature referenced here, synchronized communication will refer to full communication throughout this work.



(a) Centralised case (b) As seen from agent i

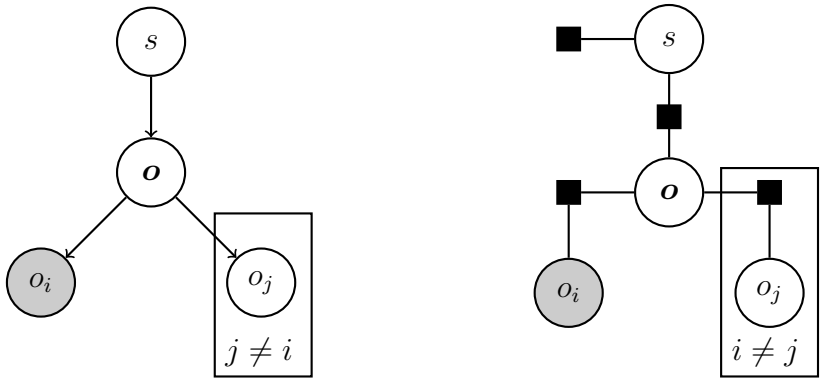
Figure 6.1: Graphical model representation of the centralised one-step multi-agent decision problem

6.2.1 Synchronised Communication

Consider a system in which agents are either fully synchronised or have no exchange of information at all. In such cases one need only consider those time-steps with zero communication. In these situations each agent will only have access to its local observation o_i. Because the system is assumed to have no sequentiality, an agent's belief will be fully defined by this local observation. Equation (6.4) then becomes

$$p(s, \boldsymbol{a}_{-i}|o_i) = p(\boldsymbol{a}_{-i}|s, o_i)p(s|o_i) \tag{6.5}$$

So far there has been no mention of how to compute the local belief state $p(s|o_i)$. This is a matter of straightforward Bayesian inference based on the knowledge of the system's dynamics including the prior distribution over states. One convenient way of solving this computation is by casting the scenario as a graphical model and using standard solution algorithms to obtain the marginal distribution $p(s|o_i)$. For prerequisites about graphical models and standard inference algorithms see Section 2.3.1 and appendix A.



(a) Directed graphical model

(b) Factor graph representation

Figure 6.2: Directed graphical model and factor graph as seen by agent i under missing communication

To begin with, consider the graphical representation of a centralised multi-agent decision process shown in Figure 6.1a. The environment starts out in a state s, which generates a joint observation $\boldsymbol{o}$. By way of the centralised policy the state also generates a joint action $\boldsymbol{a}$. Together the state and action determine the stochastic transition to the next state s'. Figure 6.1b shows what this graph looks like from the perspective of agent i which only observes its own local node variable o_i. The information about the remaining local observations is obtained from communication with others, which effectively makes the state observable.

Now consider the case in which there is no communication between agents at a given time-step. In the one-step scenarios considered for now, the current state does not depend on the previous state and action. The graphical model as seen from agent i therefore simplifies to the one shown in Figure 6.2a, where all variables other than the local observation are now unobserved. Note that this graph is now a tree which allows straightforward use of the sum-product algorithm [Pearl, 1988] to obtain the marginal distribution over s. To this end the directed graph is turned into a factor graph, as shown in Figure 6.2b. From here the local marginals $p(s|o_i)$ can be calculated for each

agent by passing messages through the graph.

This covers one part of Equation (6.5), that is the calculation of the posterior probability over the state. Obtaining an expression for the conditional probability of all others' actions is less simple: Assuming $p(\boldsymbol{a}_{-i}|s, o_i)$ were known, agent i could calculate its expectation of rewards given by equation (6.2) and choose the local action which maximises this value according to

$$a_i^* = \arg\max_{a_i} \sum_s p(s|o_i) \sum_{\boldsymbol{a}_{-i}} p(\boldsymbol{a}_{-i}|s, o_i) R(s, a_i, \boldsymbol{a}_{-i}) \tag{6.6}$$

At the same time, the other agents will be executing the same calculation, based on their local observations. The probability of agent j's local action is therefore given by

$$p(a_j|o_j) = \delta(a_j - \arg\max_{a_{j'}} \sum_s p(s|o_j) \sum_{\boldsymbol{a}_{-j}} p(\boldsymbol{a}_{-j}|s, o_j) R(s, a_{j'}, \boldsymbol{a}_{-j})) \tag{6.7}$$

Agent i can express the probability distribution over the other agents' joint actions as

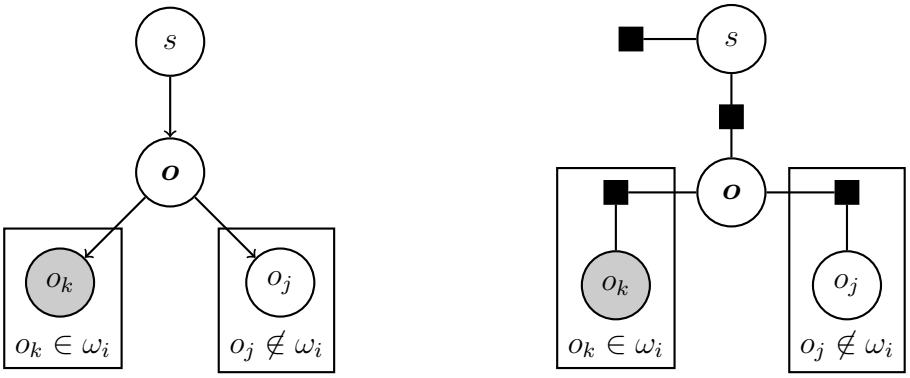
$$p(\boldsymbol{a}_{-i}|s, o_i) = \sum_{\boldsymbol{o}_{-i}} p(\boldsymbol{o}_{-i}|o_i) \prod_{j \neq i} p(a_j = \boldsymbol{a}_{-i}^{(j)}|\boldsymbol{o}_{-i}^{(j)}) \tag{6.8}$$

where the probability $p(a_j = \boldsymbol{a}_{-i}^{(j)}|\boldsymbol{o}_{-i}^{(j)})$ is given by

$$p(a_j = \boldsymbol{a}_{-i}^{(j)}|\boldsymbol{o}_{-i}^{(j)}) = \begin{cases} 1 & \text{if } \boldsymbol{a}_{-i}^{(j)} = \arg\max_{a_{j'}} \sum_s p(s|o_j) \sum_{\boldsymbol{a}_{-j}} p(\boldsymbol{a}_{-j}|s, o_j) R(s, a_{j'}, \boldsymbol{a}_{-j}) \\ 0 & \text{otherwise.} \end{cases} \tag{6.9}$$

and $\boldsymbol{o}_{-i}$ is used to denote the joint observation excluding agent i. Note that the conditional expression in Equation (6.9) contains the probability over all actions other than j, $p(\boldsymbol{a}_{-j}|s, o_j)$, which includes action a_i. Agent i's expression for the other agents' action choices is therefore directly dependent on its own current decision-making process and vice verca.

If agents were to actively reason about the choices being made by others, this interdependence would lead to an infinite regress with one agent reasoning about another agent reasoning about itself, and so on. The remaining challenge in developing



(a) Directed graphical model (b) Factor graph representation

Figure 6.3: Directed graphical model and factor graph for the general communication case as seen by an individual agent

local decision-making algorithms for one-step scenarios therefore lies in finding an approximate solution to this infinite regress.

6.2.2 General Communication

In the above formulation agents were assumed to only have access to their local observations. In a generalised setting (e.g.one with individual communication links between agents) this treatment needs to be adapted.

Within this work the assumption is that communications between agents convey local observations. Thus, after a communication is carried out the participating agents have full knowledge of the others' current local observations. An alternative approach could be to allow agents to communicate different variables, for example their current belief over the global state or the action they plan to choose. Communicating the current belief would require merging an individual agent's belief with those received from others to give an updated joint belief. Exchanging local observations during communication allows agents to compute this updated belief directly by taking the others' local observations into account. If agents were to communicate action choices rather than

observations, the local actions received from others could influence the choice of action just made (and communicated) by an individual agent. Such a communication scheme would therefore require some mechanism to ensure that action choices converge to a common point. It is not immediately clear that such a mechanism exists and whether it would require additional communication between agents. Here, local observations are therefore the communicated variable of choice

Let $\omega_i = \{o_i, \{o_j\}|j \text{ communicated with } i\}$ denote the total amount of information available to agent i, which comprises its own local observation as well as those local observations from others which it has obtained from communication. Further, let $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n)$ be the joint vector of information held by agents individually and let $\boldsymbol{\omega}_{-j}$ denote the joint vector of information held by all agents other than j. Then the optimal local action as seen by i is given by

$$a_i^* = \arg\max_{a_i} \sum_s p(s|\omega_i) \sum_{\boldsymbol{a}_{-i}} p(\boldsymbol{a}_{-i}|s, \omega_i) R(s, a_i, \boldsymbol{a}_{-i}) \tag{6.10}$$

where $p(\boldsymbol{a}_{-i}|s, \omega_i)$ is given by

$$p(\boldsymbol{a}_{-i}|s, \omega_i) = \sum_{\boldsymbol{\omega}_{-i}} p(\boldsymbol{\omega}_{-i}|\omega_i) \prod_{j \neq i} p(a_j = \boldsymbol{a}_{-i}^{(j)}|\boldsymbol{\omega}_{-i}^{(j)}) \tag{6.11}$$

which is Equation (6.8) with ω_i substituted for o_i. Note, however, that the probability of the joint observation $p(\boldsymbol{\omega}_{-i}|\omega_i)$ follows a more complex expression here: it depends not only on the local observation probabilities but also on the probability of information exchange between agents:

$$p(\boldsymbol{\omega}_{-i}|\omega_i) = \sum_{\boldsymbol{o}} p(\boldsymbol{o}|\omega_i)\ p(\boldsymbol{\omega}_{-i}|\boldsymbol{o}) \tag{6.12}$$

where $p(\boldsymbol{\omega}_{-i}|\boldsymbol{o})$ depends on how many of the local observations that make up $\boldsymbol{o}$ were communicated.

Obtaining the posterior probability $p(s|\omega_i)$ in equation (6.10) is done in the same way as described in section 6.2.1 by passing messages through a graph. The difference

here is that due to the more general communication links, an agent might now use more than just its local observation in the inference process. The updated graphical models for this case are shown in Figure 6.3. only variable communicated

6.3 Iterative Approximation

The first approximate approach to the infinite regress considered here is based on finding an iterative solution to Equations (6.6) and (6.10) respectively. To begin with, consider the simpler case of fully synchronised communication. Equations (6.6) and (6.8) define a system of coupled equations: for known values of $p(a_j|o_j)$, calculating a_i^* is straightforward. One approach to solving these equations can therefore be to initialise all action probabilities $p(a_i|o_i)$ to some starting value and iteratively update the expressions until they converge. For a single optimal action, this must be done for all of the other agents' possible observations o_j. Solving for local policies in this manner is still a centralized approach. However, the problem at hand has now been decentralized to a certain extent: instead of considering the solution to a single central value function as in Equations (4.2) and (4.3), the aim is to find solutions to the individual value functions of single agents.

There are infinitely many ways of initialising the action probabilities and in general there is no guarantee that a certain, or indeed any, initialisation will converge to stationary policies. For practical purposes, methods such as over-relaxation [Young, 1950] can be used to increase the probability of convergence in such a system. For the benchmark scenarios considered in this work the algorithm always converged, suggesting that it can be an effective approach.

6.4 Results for Iterative Approximation

The iterative approximation was applied to the benchmark problems described in Chapter 5 and compared to the benchmark algorithms introduced in the same chapter. For this implementation of the iterative algorithm the action probabilities were initialised with a uniform distribution, that is, all actions were initially equally probable. See algorithm 6.1 for a full description of the policy generation algorithm.

Algorithm 6.1 Iterative algorithm for computing best local actions

for $i = 1$ to numagents **do**
  $p(a_i^0|o_i) \leftarrow 1/|A_i|$
end for
while converged $== False$ **do**
  max_diff $= 0$
  for $i = 1$ to numagents **do**
    for obs$= 1$ to numobs **do**
      $p(a_i^{new}|$ obs$) \leftarrow$ recalculate action prob()
      diff$\leftarrow p(a_i^{new}|obs) - p(a_i^{old}|obs)$
      if diff>max_diff **then**
        max_diff $\leftarrow$diff
      end if
      $p(a_i^{old}|$ obs$) \leftarrow p(a_i^{new}|$ obs$)$
    end for
  end for
  if max_diff $<$ convergence_diff **then**
    converged $\leftarrow True$
  end if
end while

6.4.1 Tiger Problem

The iterative algorithm was first applied to the tiger problem (see Section 5.2.1). Each scenario run consisted of 500 time-steps without communication.

Figure 6.4 shows the average reward obtained under each of the decision-making algorithms for the three settings of the reward matrix. Table 6.1 gives the respective p-values obtained in a two-tailed Welch t-test [Welch, 1947] comparing the averages

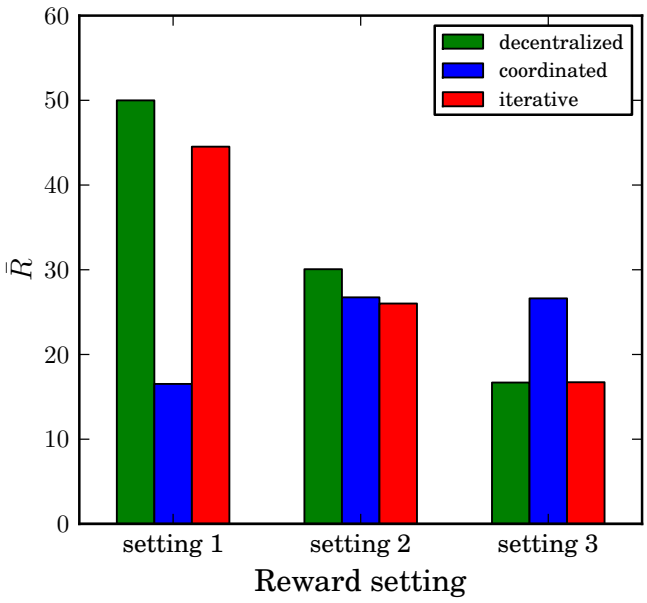


Figure 6.4: Average obtained reward under the decentralized (see Section 5.3.1), guaranteed coordinated (see Section 5.3.2) and iterative algorithms in the tiger problem. Data was obtained by averaging over 500 simulations with 500 time-steps each.

Algorithms		p-values		
		Setting 1	Setting 2	Setting 3
decentralized	coordinated	0.0	0.0	0.0
iterative	coordinated	0.0	3.37×10^{-6}	0.0
decentralized	iterative	0.0	0.0	0.82

Table 6.1: p-values obtained in a two-tailed Welch t-test comparing different algorithms in the tiger problem.

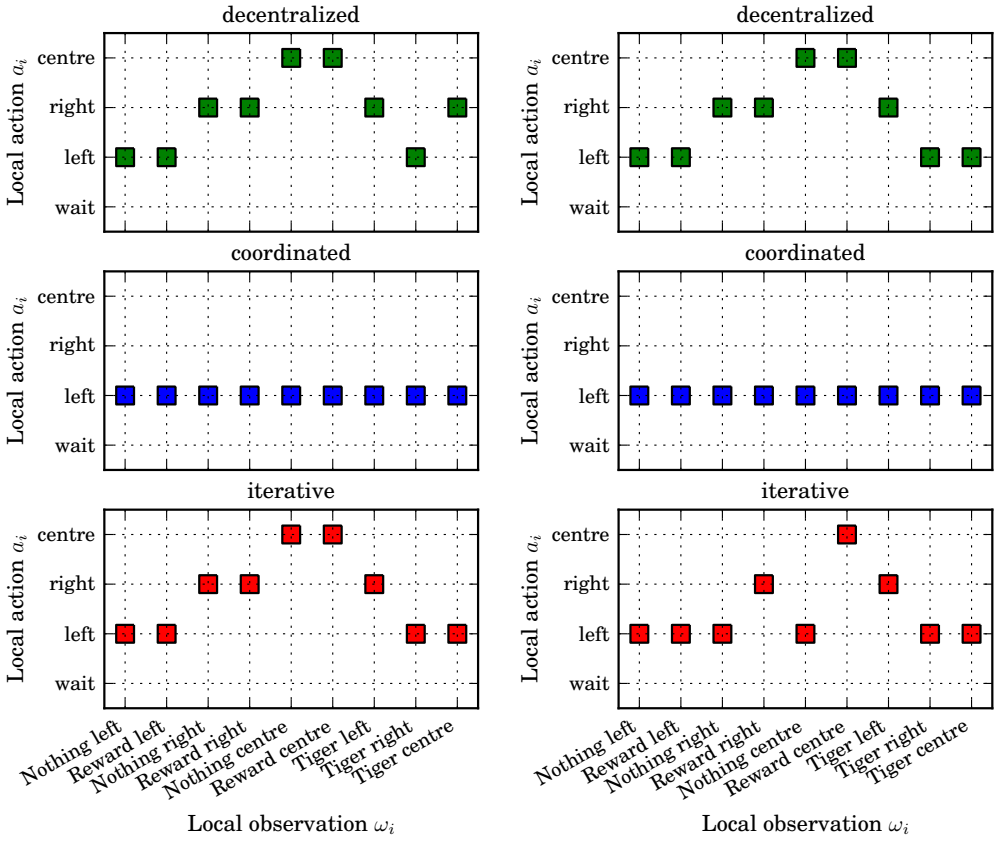
obtained under the different algorithms.

While the guaranteed coordinated approach outperforms the other two in the third setting, the rewards obtained are roughly equal in the second setting (with the coordinated algorithm preforming slightly better). In the first setting the iterative and the decentralized algorithms outperform the guaranteed coordinated one. Recall that in all settings coordinated actions were given the highest reward. The fact that the coordinated approach does not perform best in all of these settings can be seen as a

first confirmation of the initial hypothesis that guaranteeing agent coordination at the cost of discarding local information does not always produce the best performance.

In the first two settings the iterative algorithm performs nearly as well as the decentralized one. In the third setting it performs as well as the decentralized. The solution algorithm for the decentralized algorithm used here is an approximate one (see Section 5.3.1) and might not yield the globally optimal decentralized policy. The comparison between the iterative algorithm and the decentralized one can be informative nonetheless, as one might expect the approximate decentralized algorithm to be reasonably close to the exact solution. This would in turn give an upper limit for how well an approximate local algorithm might perform.

Figure 6.5 shows the policies resulting from the different decision-making algorithms for one of the reward settings. Under the coordinated algorithm a static action is chosen independent of the local observation. This is the expected behaviour, as agents ignore their local observations and must therefore choose the action which is optimal under the prior probabilities of global states. For the decentralized and the iterative algorithm, individual agent actions depend on the local observation. Note that in both cases the resulting policies differ between the agents, even though the optimal policy under full communication is symmetric. This is a result of agents successively updating their policies in response to one another and the uncertainty that results from not knowing the full joint observation. Take the iterative policy in which agent 1 is computing its best responses based on the initialised action probabilities of agent 2. This yields the policy shown in Figure 6.5a. Agent 2 subsequently calculates its best response to this policy. Consider the local observation where agent 2 observes nothing behind the centre door. In this case agent 1 could have observed the reward or the tiger behind either the left or the right door. According to its momentarily static policy this will result in a “left” or “right” action with equal probability. Because coordinated



(a) Agent 1 (b) Agent 2

Figure 6.5: Actions chosen by both agents for first setting of the tiger problem. See Figure 5.2 in chapter 5 for details of the reward settings used.

actions yield the highest reward, the best response for agent 2 is therefore to choose either “left” or “right” as well. In either case this will result in a different mapping to that from agent 1, which chose to play the “centre” action for the same observation.

6.4.2 Meeting Problem

Next, the algorithm was applied to the meeting problem (see Section 5.2.2). Again each scenario run considered of 500 time steps without communication.

Figure 6.6 shows the average reward obtained under the different decision-making algorithms. Table 6.2 gives the respective p-values obtained when comparing the

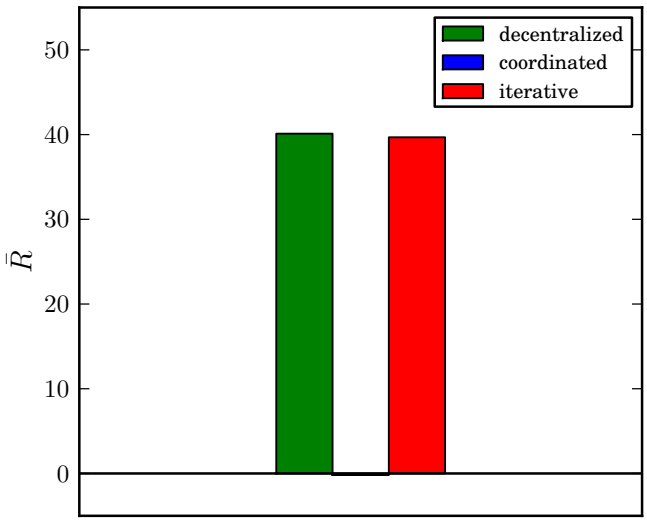


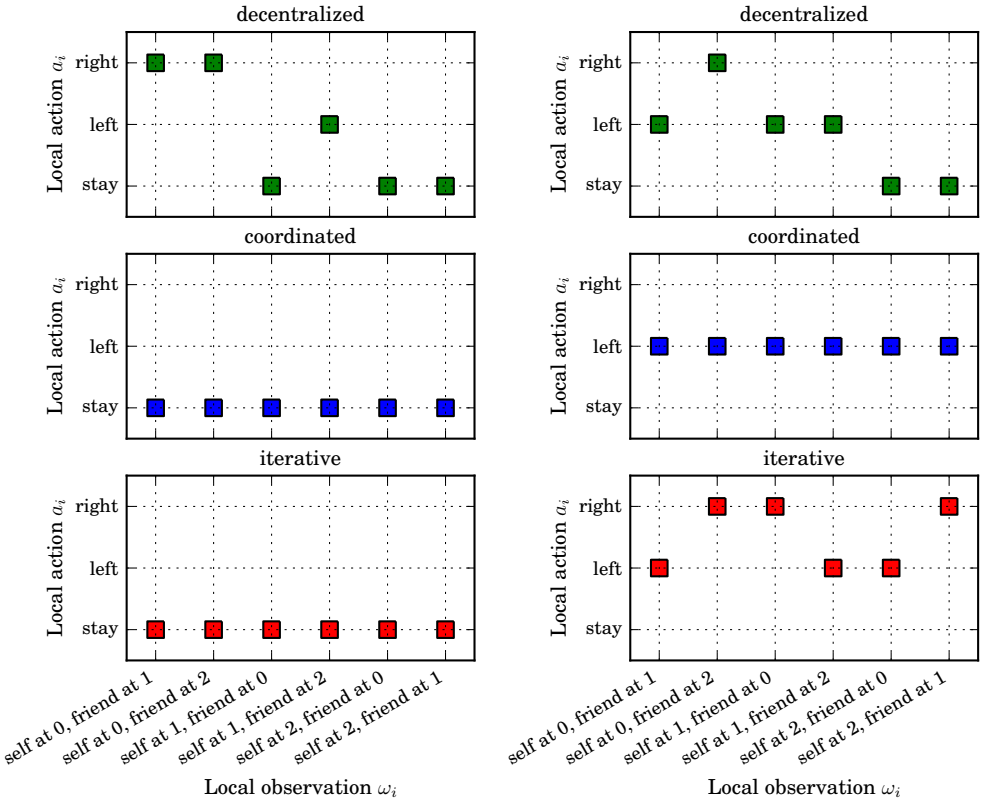
Figure 6.6: Average obtained reward under the decentralized, guaranteed coordinated and iterative algorithms in the meeting problem. Data was obtained by averaging over 500 simulations with 500 time-steps each.

Algorithms		p-values
decentralized	coordinated	0.0
iterative	coordinated	0.0
decentralized	iterative	0.08

Table 6.2: p-values obtained in a two-tailed Welch t-test comparing different algorithms in the meeting problem.

performance means in a two-tailed Welch t-test.

In this scenario the guaranteed coordinated approach performs much worse than both the decentralized and the iterative algorithms. This is the case even for a setting in which high penalties are given for any but the desired coordinated action. The iterative algorithm again performs nearly as well as the decentralized algorithm. Figure 6.7 shows the local policies produced under the different decision-making algorithms. Again the coordinated algorithm results in a static policy while the decentralized an iterative algorithms produce asymmetric local policies.



(a) Agent 1 (b) Agent 2

Figure 6.7: Actions chosen by both agents for the meeting problem. See Figure 5.4 in chapter 5 for details of the reward setting used.

6.4.3 Monitoring Problem

Finally, the iterative algorithm was applied to the monitoring problem (see Section 5.2.3).

Figure 6.8 shows the average reward obtained under the different decision-making algorithms in this scenario. In all three settings the iterative algorithm performs significantly worse than the decentralized one. In the first two settings it performs much better than the coordinated approach while the difference is less pronounced in the third setting.

Figure 6.9 shows the actions chosen by all agents for all possible local observations.

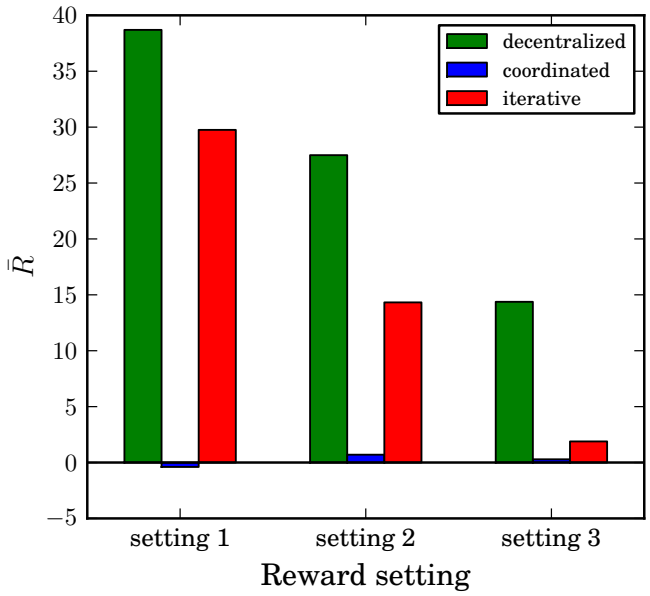
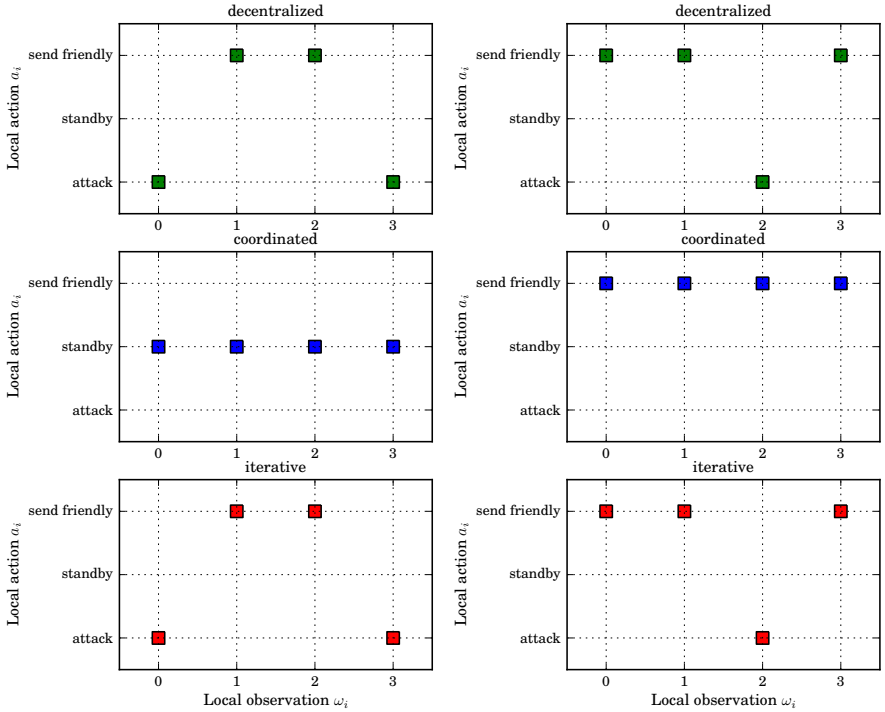


Figure 6.8: Average reward obtained per time-step with zero communication in the monitoring problem. Data was collected over 500 simulations with 50 time-steps each. The comparison of the iterative performance to the other algorithms in a two-tailed Welch t-test yielded a p-value of 0.0 for the first two settings. In the third setting the comparison with the coordinated algorithm yielded a p-value of 0.03.

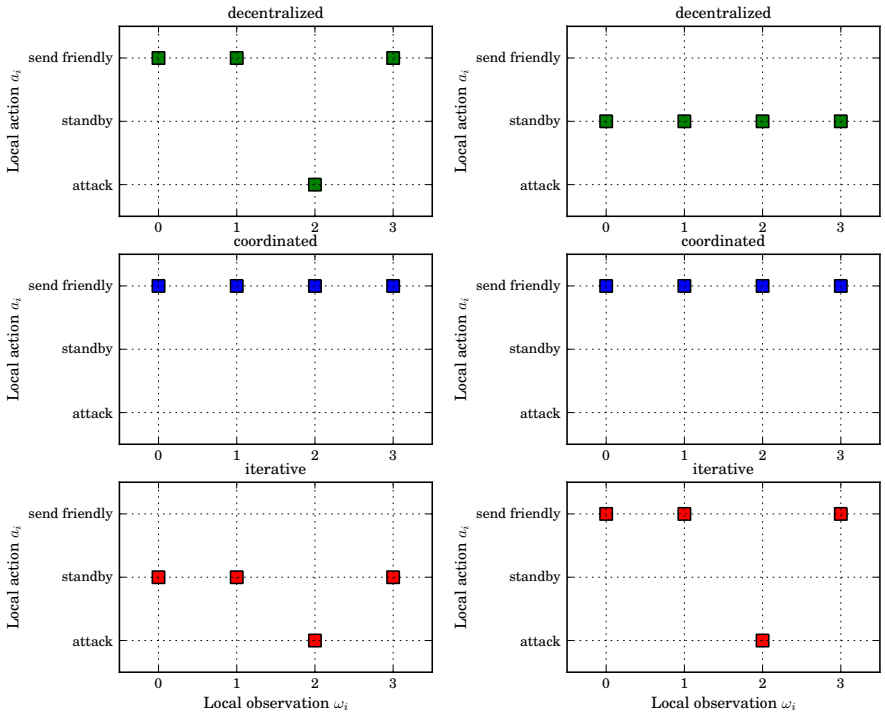
This plot shows why the overall performance of the iterative algorithm is worse than the decentralized one: while the first two agents choose locally optimal actions (i.e. the same ones as under the decentralized algorithm), the third and fourth agents differ from their respective optimal policies.

As discussed in Section 5.2.3 the monitoring problem can feature other communication patterns than the synchronised communication considered here so far. However, more general communication patterns greatly increase the complexity of calculating a local policy, because the space of possible local information held by a single agent, ω_i, increases (see section 6.2.2). On a Dell PowerEdge 1950 with 16 GB of RAM and 2 quad core CPUs running at 3.0 GHz each, the calculation of the iterative policies for a setting with individual communication links between agents required roughly a week of run-time. To compare the performance in all three settings would hence require several



(a) Agent 1

(b) Agent 2



(c) Agent 3

(d) Agent 4

Figure 6.9: Actions chosen by all four agents under the different decision-making algorithms in the first setting of the monitoring problem.

weeks of pre-calculations. As such a time-scale is impractical for most applications, other communication patterns were not considered here.

6.4.4 Discussion

The application of the iterative algorithm to the first two benchmark scenarios shows that it can produce good to near-optimal local policies in some settings with full synchronisation and small observation spaces. But the application to the monitoring problem also showcases the limitations of this algorithm: in settings with more agents and more general communication patterns, the observation space soon becomes too large for the iterative approach to be feasible. This is not very surprising, as the algorithm requires iterating over all possible combinations of observations in order to pre-compute all local policies. The iterative approach is therefore not suited for more complex scenarios and/or very general communication settings. An algorithm which scales to such settings must have lower computational complexity. The following section explores such an approach based on a simple heuristic with which policies can be pre-computed offline or, for larger scenarios, can be calculated on-line. This allows to compute local actions only for those observations which are actually encountered in the simulation.

6.5 Static Approximation

Return to Equation (6.4), from which the joint probability of the global state and the other agents' actions is given by

$$p(s, \boldsymbol{a}_{-i}|b_i) = p(\boldsymbol{a}_{-i}|s, b_i)b_i(s) \tag{6.13}$$

Assuming the uncertainty over other agents' actions is not directly dependent on the local belief b_i but rather due to the uncertainty over global states, the probability over

others actions becomes $p(\boldsymbol{a}_{-i}|s, b_i) = p(\boldsymbol{a}_{-i}|s)$. The joint probability can then be simplified to

$$p(s, \boldsymbol{a}_{-i}|b_i) = p(\boldsymbol{a}_{-i}|s)b_i(s) \tag{6.14}$$

This is equivalent to assuming that all other agents know the global state and that their policy given the global state is known. In the scenarios considered here this assumption does not hold, but the following analysis will show that it is nonetheless a useful approximation.

in practise, the distribution over others' actions given the global state, $p(\boldsymbol{a}_{-i}|s)$ is not known, because the other agents only have partial knowledge of the state. Simplifying the above expression therefore has not yet solved the infinite regress problem of reasoning over agents' actions. As an approximate solution consider the following approach. Rather than reasoning explicitly about the functional form of another agent's action probability $p(a_j|s)$, agent i might presume to know this distribution from prior assumptions. It can then calculate its best action given this postulated distribution and execute it, regardless of whether agent j's action probability is adequately represented by the assumed distribution. An agent's best local action then becomes

$$a_i^* = \arg\max_{a_i} \sum_s p(s|b_i) \sum_{\boldsymbol{a}_{-i}} p(\boldsymbol{a}_{-i}|s) R(s, a_i, \boldsymbol{a}_{-i}) \tag{6.15}$$

In this case there is no difference in treatment between synchronised and general communication, as the agents do not actively reason about the amount of local information available to others.

The algorithm outlined above can be understood as a first-order approximation to the iterative algorithm considered in the previous section: instead of re-adjusting the best responses to other agents' current policies all agents use the initial distribution over action probabilities as a basis for their calculations. This is a simple heuristic but has the great advantage of very low complexity: to compute the optimal local action,

the main calculation needed is the one which determines the local posterior distribution over global states, together with a summation over all possible joint actions of others'.

The resulting best response will depend on the static distribution used in the calculation. The following sections describe three possible priors and the rationale behind them.

6.5.1 Optimistic Approximation

An optimistic approach to approximating $p(a_j|s)$ would be to assume that agent j does indeed know the global state and hence chooses the local action a_j given by the joint centralised policy for that state. Under this assumption the distribution over actions is given by

$$p(a_j|s) = \begin{cases} 1 & \text{if } a_j = \pi(s)_j \\ 0 & \text{otherwise.} \end{cases} \tag{6.16}$$

This is similar to the approximation used in [Chechetka and Sycara, 2007].

6.5.2 Uniform Approximation

For the iterative algorithm a uniform distribution over action probabilities was chosen. The same choice can be made for the static approximation. It can be motivated by noting that in practise no prior knowledge of agent j's choice of action might be available. In such a case agent i will assume that all actions a_j are equally likely, regardless of the state. The distribution over actions is then given by

$$p(a_j|s) = \frac{1}{|A_j|} \tag{6.17}$$

and in the case of more than two agents

$$p(\boldsymbol{a}_{-i}|s) = \prod_{j \neq i} p(a_j|s)|_{a_j \in \boldsymbol{a}_{-i}} \tag{6.18}$$

6.5.3 Pessimistic Approximation

Finally, a pessimistic agent might assume that the interdependence of local decision-making will lead to sub-optimal action choices made by other agents. In the worst case this could mean that agent j will choose the worst possible action in a given state. Such an action choice is not very probable in most settings but is nonetheless worth studying, as it provides a lower bound for how badly the algorithm might perform.

Under this approximation the distribution over agent j's actions is given by

$$p(a_j|s) = \begin{cases} 1 & \text{if } a_j = \arg\min_{\boldsymbol{a}}(Q(s,\boldsymbol{a}))_j \\ 0 & \text{otherwise.} \end{cases} \tag{6.19}$$

where $V(s,\boldsymbol{a})$ refers to the value of taking an action in a given state under full communication.

Each of the distributions described above can be used to implement an approximate local decision-making algorithm by calculating the expected value of a local action under this distribution according to equation (6.15). See Algorithm 6.2 for a more detailed description.

Algorithm 6.2 Decision-making algorithm based on heuristic approximations

```
for i = 1 to numagents do
  p(a_i^0|s) ← static distribution
end for
for i = 1 to numagents do
  for obs = 1 to numobs do
    act ← calculate_best_response( )
  end for
end for
```

6.6 Results for Static Approximation

The heuristic decision-making algorithm has been implemented for all three approximations and applied to the benchmark scenarios.

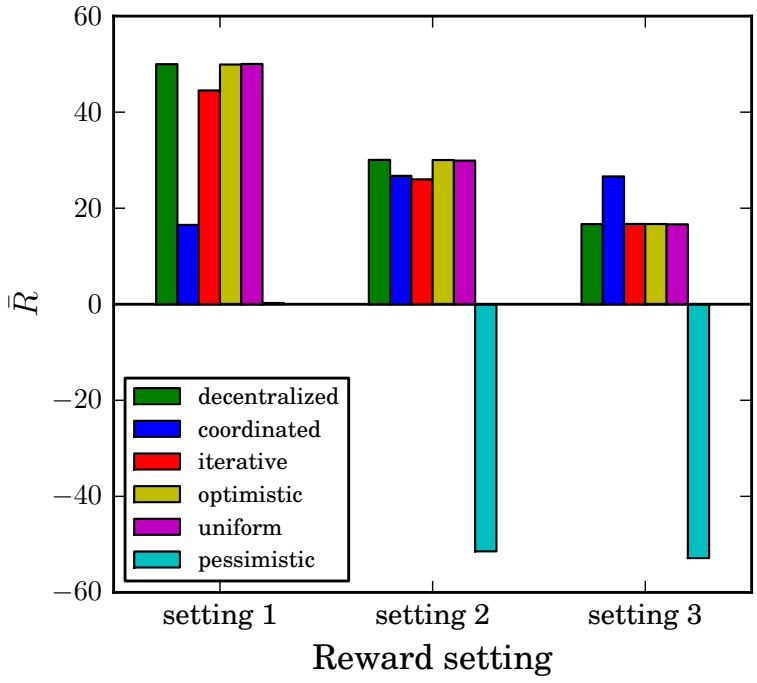


Figure 6.10: Average reward obtained under the decentralized, guaranteed coordinated, iterative and static algorithms in the tiger problem. Data points were obtained by averaging over 500 simulations consisting of 500 time-steps each.

6.6.1 Tiger Problem

The static decision-making algorithm was first applied to the tiger problem, using the three different action distributions discussed previously. As previously, each scenario run consisted of 500 time-steps without communication.

Figure 6.10 shows the performance of the approximate algorithms compared to the decentralized, guaranteed coordinated and the iterative approaches. Table 6.3 gives the respective p-values obtained in a two-tailed Welch t-test comparing the averages obtained under the different algorithms.

The pessimistic approach consistently performs worse than any of the other algorithms, while the optimistic and the uniform approaches achieve similar performance. Compared to the guaranteed coordinated algorithm the performance of the optimistic /

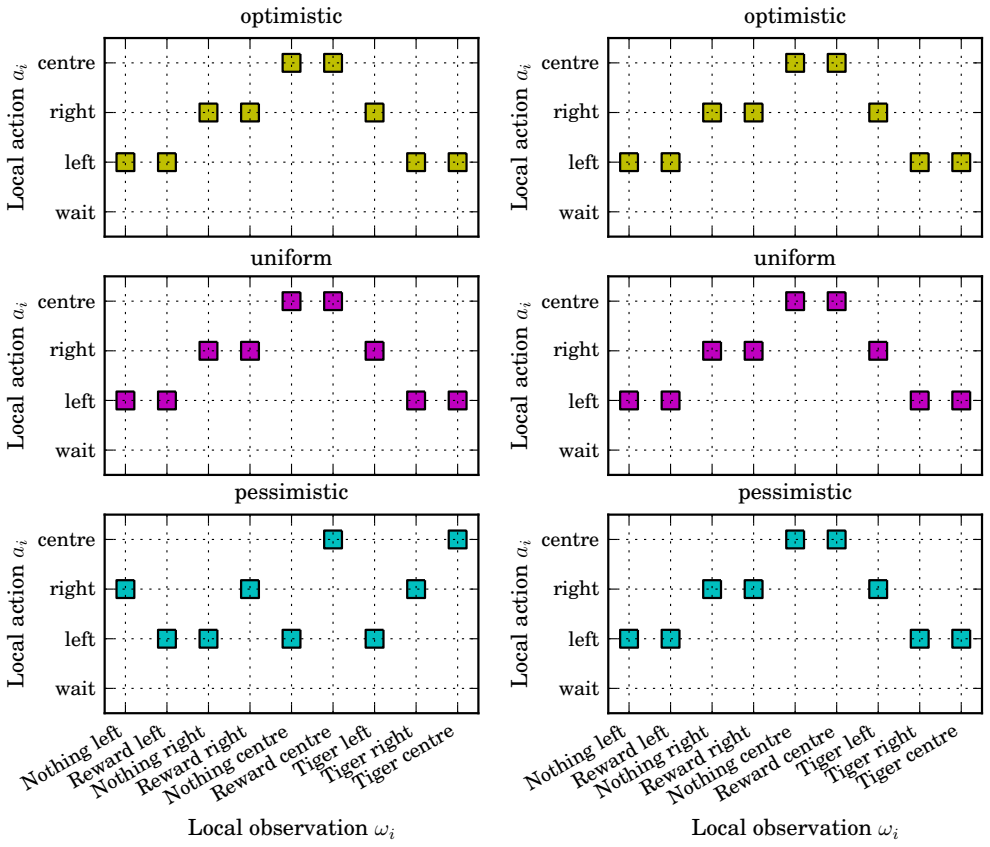
Algorithms		p-values		
		Setting 1	Setting 2	Setting 3
optimistic	coordinated	0.0	0.0	0.0
uniform	coordinated	0.0	0.0	0.0
pessimistic	coordinated	0.0	0.0	0.0
uniform	optimistic	0.13	0.19	0.69
uniform	decentralized	0.56	0.09	0.72
optimistic	decentralized	0.36	0.72	0.90

Table 6.3: p-values obtained in a two-tailed Welch t-test comparing different algorithms in the tiger problem.

uniform algorithms depends on the setting of the reward matrix. They clearly outperform it in the first setting, while achieving less average reward in the third. For setting 2 all algorithms (other than the pessimistic) obtain similar rewards with the decentralized, uniform and optimistic algorithms performing slightly better than the others. In comparison to the iterative approach the uniform and optimistic algorithms perform slightly better in the first two settings. Indeed, they show a similar performance to that of the decentralized algorithm.

Figure 6.11 shows the action choices made by both agents under the static algorithms for one setting of this scenario. The algorithm based on the optimistic approximation and the one based on the uniform approximation lead to the same local action choices, which are symmetric. The resulting policy is the same for both agents and is close to the policy found by the decentralized algorithm in the same scenario (see Figure 6.5)

The policy resulting from the pessimistic approximation is asymmetric on the other hand: while the second agent chooses to play the same actions as for the other two approximations, the first agent chooses a different policy. This might be surprising at first, but is due to the fact that the worst possible joint action (under full communication) is asymmetric. In particular, the first agent's worst action policy is to always wait,



(a) Agent 1 (b) Agent 2

Figure 6.11: Actions chosen by both agents in the first setting of the tiger problem, depending on the approximation used for the static decision-making algorithm.

regardless of the local observation. The second agent's best response under this belief is the same as the policy found from the optimistic and uniform approximations.

6.6.2 Meeting Problem

Next the algorithms were applied to the meeting problem. Again each simulation consisted of 500 time-steps with missing communication.

Figure 6.12 shows the performance of the static algorithms compared to the decentralized, the guaranteed coordinated and the iterative approaches in this scenario. Here the performance of the individual approximations differs from the performance

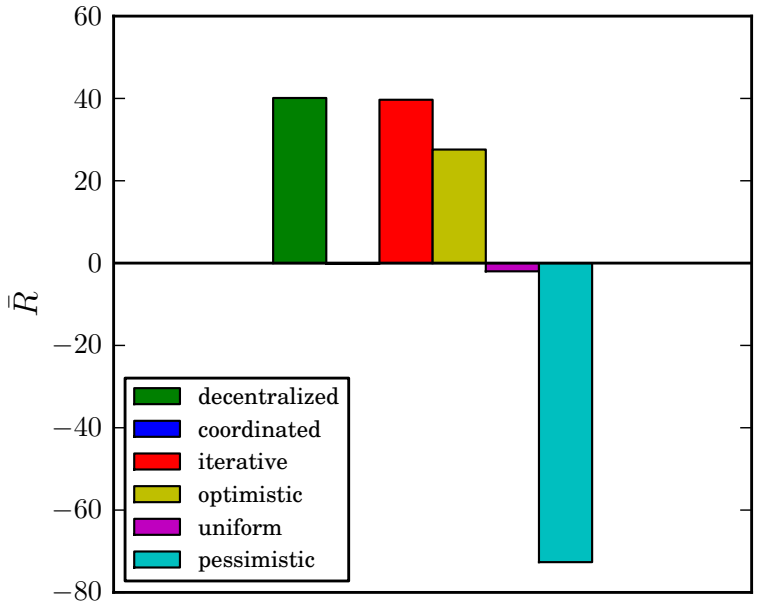
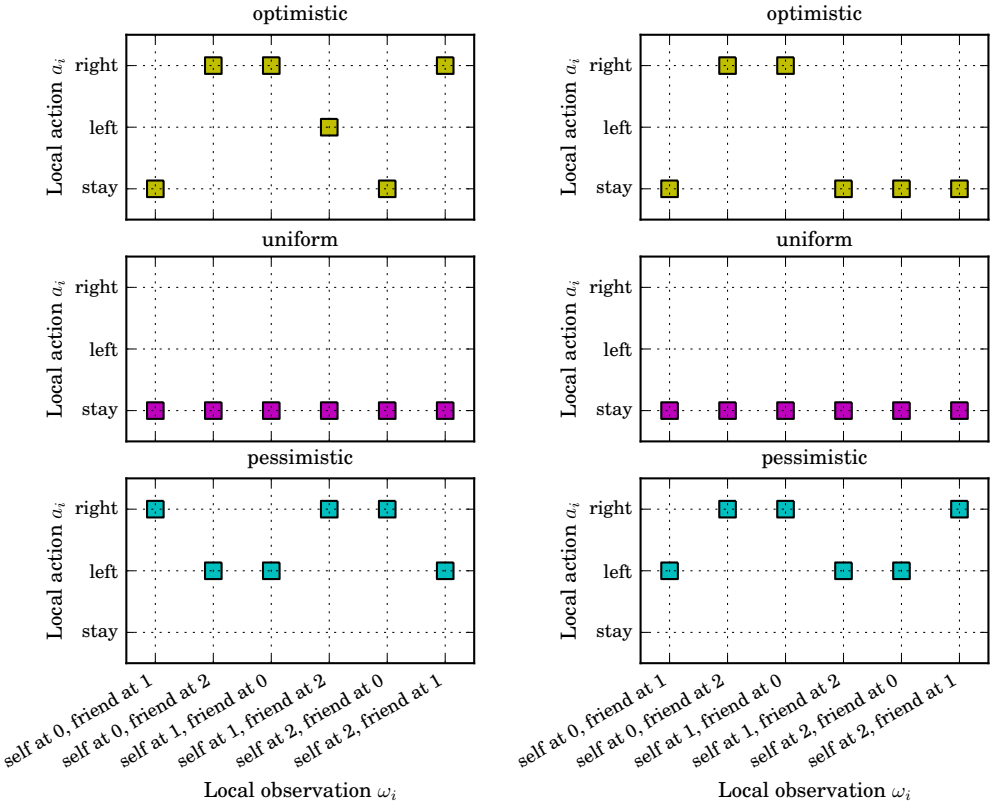


Figure 6.12: Average reward obtained under the decentralized, guaranteed coordinated, iterative and static algorithms in the meeting problem. Data points were obtained by averaging over 500 simulations consisting of 500 time-steps each. The comparison of the static algorithms to the decentralized, coordinated and iterative algorithms in a two-tailed Welch t-test yielded a p-value of 0.0 in each case.

in the tiger problem. While the pessimistic approximation again performs very poorly, the uniform distribution also achieves very low rewards. The optimistic approach on the other hand clearly outperforms the guaranteed coordinated approach. It does not match the performance of the distributed and iterative algorithms, however. This result suggests that in this setting there is a benefit from knowing (or assuming to know) the other agents' distribution over actions with little uncertainty.

Figure 6.13 shows the actions chosen by both agents for all three approximations used. It illustrates why the uniform approach performs poorly in this setting: both agents always remain at their position and thus obtain a penalty. The pessimistic approximation similarly leads to uncoordinated joint actions.



(a) Agent 1 (b) Agent 2

Figure 6.13: Actions chosen by both agents in the meeting problem

6.6.3 Monitoring Problem

Finally, the static algorithm with its different approximations was applied to the monitoring problem. As indicated in section 5.2.3, this setting allows for more general communication patterns than the simple full synchronisation / no communication considered so far. In addition to the synchronised setting two others have been implemented here: one in which agents have individual communication links between one another which can fail individually and one in which an agent always obtains information from one randomly chosen other agent.

Figure 6.14 shows the average rewards obtained under the decentralized, coordinated, iterative and static algorithms for a setting with fully synchronised communi-

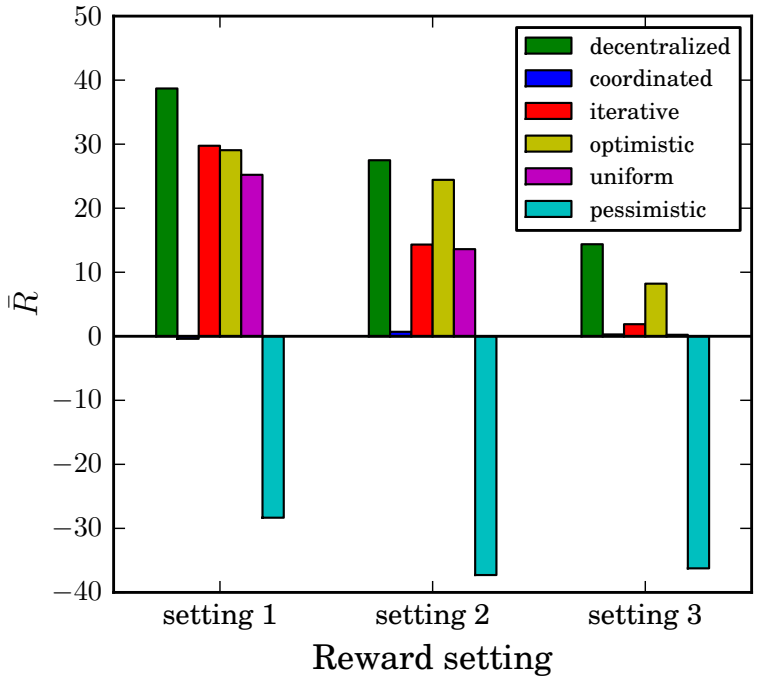


Figure 6.14: Average reward obtained under the different decision-making algorithms in the monitoring problem with zero communication. Data points were obtained by averaging over 500 simulations consisting of 50 time-steps each.

Algorithms		p-values		
		Setting 1	Setting 2	Setting 3
optimistic	coordinated	0.0	0.0	0.0
uniform	coordinated	0.0	0.0	0.94
optimistic	decentralized	0.0	1.96×10^{-8}	0.0
uniform	decentralized	0.0	0.0	0.0
optimistic	iterative	0.0	0.0	0.0
uniform	iterative	0.0	0.0	0.0
optimistic	uniform	2.22×10^{-15}	0.0	0.0

Table 6.4: p-values obtained in a two-tailed Welch t-test comparing different algorithms in the monitoring problem.

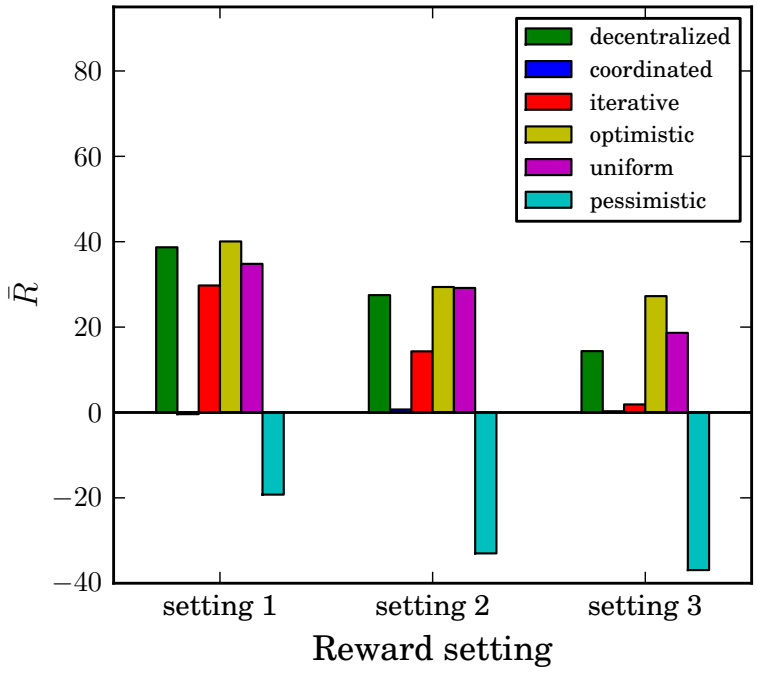
cation. The respective p-values obtained from a two-tailed Welch t-test are shown in table 6.4.

Again the pessimistic approximation performs very poorly. Close to no reward is

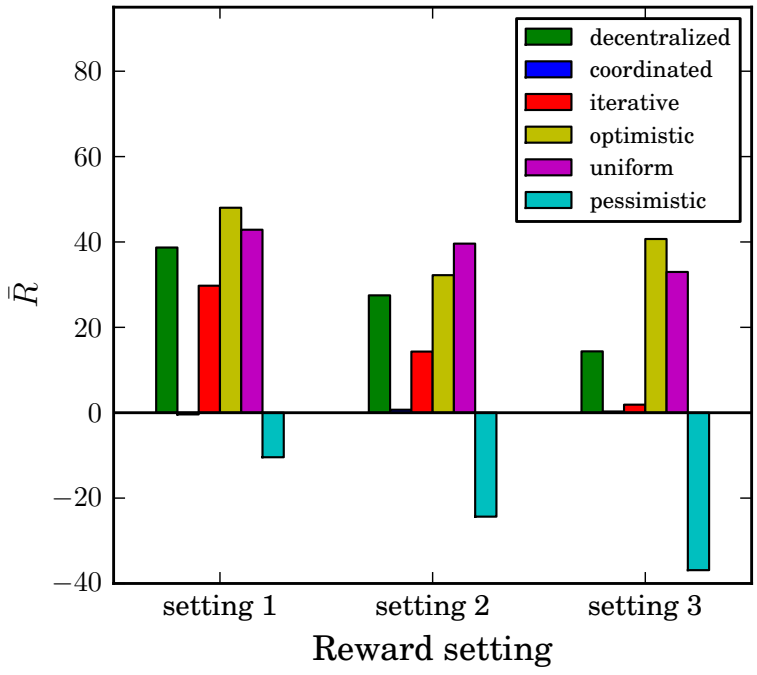
obtained under the guaranteed coordinated algorithm in all three settings, while the static algorithm with the optimistic approximation yields a good reward, in particular in the first two settings. The performance under the uniform approximation is mixed but always lower than the under optimistic one. The iterative algorithm performs slightly better than the uniform / optimistic ones in the first setting, but worse than the optimistic in the second and third setting. In all three settings the decentralized algorithm outperforms the approximate algorithms. To summarise, the optimistic algorithm shows the most consistently good performance among the approximate algorithms, while the performance of the others depends strongly on the reward setting.

Figure 6.15 shows the average reward obtained per time-step in the scenario with individual communication links for two different probabilities for successful communication. At a communication probability of $p = 0.3$ the optimistic / uniform approaches already yield rewards similar or slightly better than the decentralized solution. For a communication probability of $p = 0.6$ both approximate algorithms perform better than the decentralized one. In both settings the static algorithms outperform the iterative algorithm obtained under synchronised communication. As would be expected, there appears to be a communication probability above which the benefit of obtaining additional information outweighs the performance loss caused by approximate decision-making.

This is illustrated in figure 6.16, which shows the average reward obtained as a function of the communication probability for two settings of the monitoring problem with individual communication. Depending on the setting and the approximation used, the static decision-making algorithms can outperform the locally optimal decentralized algorithm even at low communication probabilities. Recall that the latter can be seen as a benchmark for how good the approximate algorithms perform: the better the approximation, the faster will the approximate algorithm outperform the decentralized

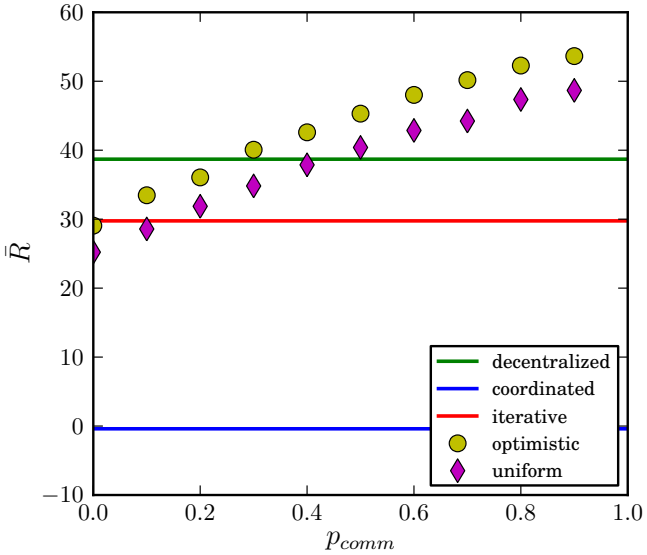


(a) $p = 0.3$

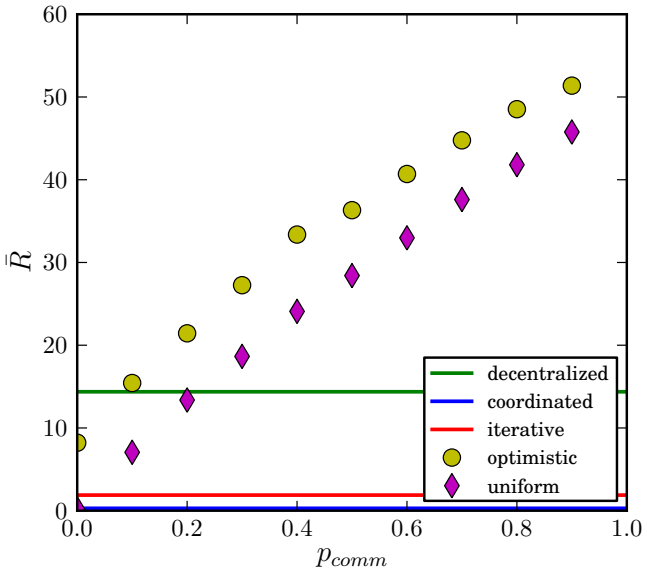


(b) $p = 0.6$

Figure 6.15: Average reward obtained per time-step in the monitoring problem with individual communication links between agents and two different probabilities for successful communication. The iterative algorithm is provided for comparison and was obtained under synchronised communication. Data points were obtained by averaging over 500 simulations consisting of 50 time-steps each.



(a) Setting 1. Comparison of uniform and optimistic algorithms in a two-tailed Welch t-test yielded a maximum p-value of 10^{-14}.



(b) Setting 3. Comparison of uniform and optimistic algorithms in a two-tailed Welch t-test yielded a p-value of 0.0 for all communication probabilities.

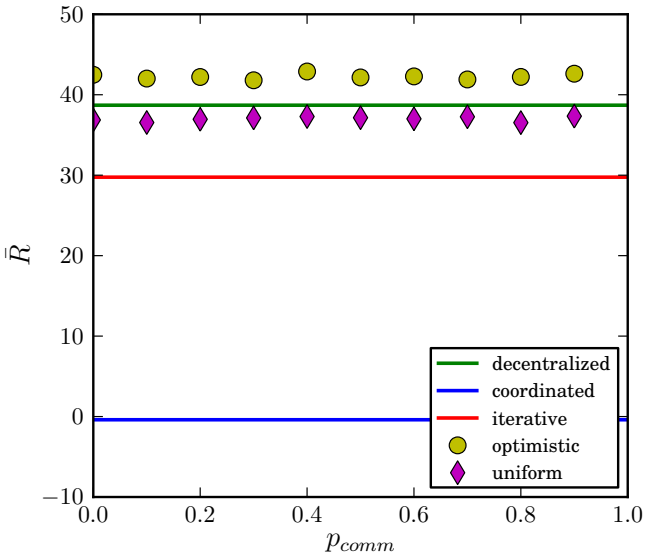
Figure 6.16: Average reward per time-step depending on the probability of successful communication from one agent to another in the first and third setting of the monitoring problem. The green, red and blue lines show the performance of the decentralized, iterative and coordinated algorithms respectively, which do not make use of communication. Data was averaged over 500 simulations with 50 time-steps each.

one when communication between some agents becomes available. For both settings studied here, the optimistic algorithm outperforms the one based on a uniform approach.

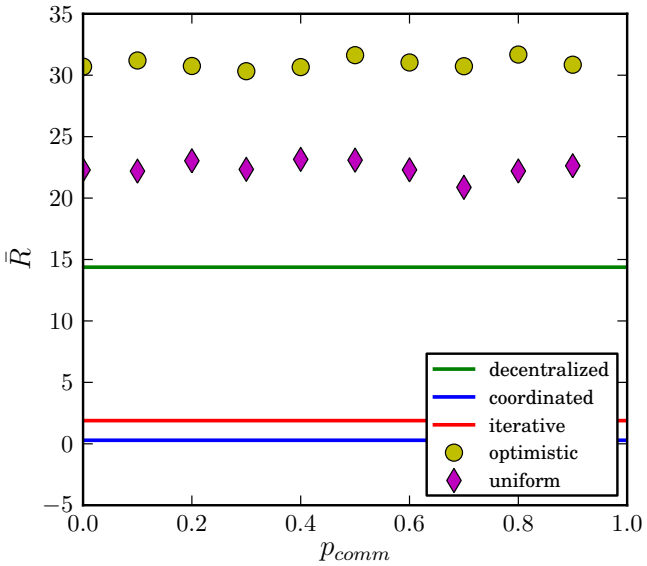
The average performance of the optimistic / uniform approximate algorithms in the final communication setting is shown in figure 6.17. Here each agent obtained information from one other agent at each time-step. This could for example be thought of as a communication from the nearest neighbour, which differs between time-steps. While the approximate algorithms clearly outperform the coordinated, iterative and decentralized ones in the second figure (Setting 3), the effect is less pronounced in the first (Setting 1). Again the optimistic algorithm performs better than the uniform one. For this communication pattern there is no dependence of the performance on the communication probability, as the number of agents from which observations are received is constant at each time-step.

6.6.4 Discussion

The algorithms discussed in this section are all based on simple heuristic approximations, which might not represent the decision-making of other agents very accurately. However, the application to the three benchmark scenarios has shown that they can produce good to near-optimal results. While the pessimistic approximation turned out to be a poor one, the optimistic and uniform approximations showcase similar performance in many cases, with the optimistic being overall more consistent. The algorithms discussed here have two distinct advantages over the iterative approach introduced in Section 6.3: the static nature of the algorithm allows policies to be computed on-line, in particular for scenarios with large observation spaces. Secondly, the algorithm can be used in scenarios with more general communication settings for which the possible set of combinations of observations quickly becomes too large to pre-compute every case iteratively. It can therefore make use of information gained from partial



(a) Setting 1. Comparison of the uniform vs. optimistic algorithms yielded p-value of 0, comparison of the uniform vs. decentralized algorithms a maximum p-value of 0.003, comparison of the optimistic vs. decentralized algorithms a maximum p-value of 10^{-8} in a two-tailed Welch t-test.



(b) Setting 3. All comparisons yielded a p-value of 0.0 in a two-tailed Welch t-test.

Figure 6.17: Average reward per time-step for a scenario in which agents obtain information from one other agent at each time-step in the first and third setting of the monitoring problem. The green, red and blue lines show the performance of the decentralized, iterative and coordinated algorithms respectively, which do not make use of communication.Data was averaged over 500 simulations with 50 time-steps each.

communications between agents.

6.7 Learning Algorithm

The analysis of the static approximations in the previous section showed that a heuristic approach can yield surprisingly good performance for various one-step scenarios. Still, the heuristics on which the algorithm is based are somewhat simplistic and it is not clear whether the approach will perform as well as it has in other settings, in particular when the actual distribution of agents' actions is very different from the one used to compute local action choices. The following section will therefore study an alternative approach based on agent learning.

Recall that so far the crucial step in developing a good approximate algorithm has been to find an approximation to the agents' beliefs over others' actions. In the first algorithm this was achieved iteratively while in the second a static distribution was postulated. In both cases agents reason explicitly over the actions chosen by others at the time of their own decision-making. But ultimately, the variable which determines an agent's decision-making is its expected reward under each local action. Recall that the best choice of action is given by

$$a_i^* = \arg\max_{a_i} V_i(a_i|b_i) \tag{6.20}$$

Based on Section 6.2 the algorithms developed in this chapter so far have arrived at an expression for $V_i(a_i|b_i)$ by modelling other agents' action choices. Another possibility lies in finding an approximation to $V_i(a_i|b_i)$ directly, without explicitly reasoning about others actions.

The approach presented below is based on agents locally learning a distribution $p(R|a_i, b_i)$ over possible rewards. This is obtained by starting out from a heuristic prior which is then updated according to Bayes' Theorem on the basis of observed

data. The expression $V_i(a_i|b_i)$ can then be obtained by calculating the expected reward under the learnt distribution:

$$V_i(a_i|b_i) = \sum_R R\, p(R|a_i, b_i) \tag{6.21}$$

While this approach avoids explicit reasoning about the interdependency of agents' action choices, it is nonetheless affected by the infinite regress. The local probability distribution $p(R|a_i, b_i)$ learnt by agent i implicitly depends on the actions chosen by others, which are in turn dependent on i's current choice. It is therefore not clear whether such an algorithm will converge to any stationary state: the true distribution over rewards for agent i might be so strongly altered by i's own action choice that the inferred distribution might never come sufficiently close to it. The following sections will explore both the convergence behaviour as well as the implementation practicalities of this algorithm. Convergence statements will rely heavily on the concept of merging of opinions in repeated games, which was introduced in Sections 3.6.3 to 3.6.5.

6.7.1 Convergence in Repeated Games

As explained in the previous section, the algorithm developed here is based on finding a local distribution over rewards for each agent, $p(R|a_i, b_i)$. Note that this expression is very similar to the subjective environment response function $\bar{e}_i$ which was described in Section 3.6.3 and defined to be

$$\bar{e}_i^{t+1} = p_i(c_i^{t+1}|\boldsymbol{h}_i^t, a_i^{t+1}) \tag{6.22}$$

If consequences are synonymous with rewards and the belief b_i can be thought of as containing all information about the previous history $\boldsymbol{h}_i^t$, the two expressions are the same. The repeated game framework and its corresponding convergence results will therefore be used in the development of the learning algorithm. As will become clear, the setting will require some theoretical extension to encompass the scenarios considered here.

To recap, let G be an infinitely repeated game in which each player has a countable set of actions A_i, a countable set of possible consequences (outcomes) C_i and a utility function $u_i : A_i \times C_i \rightarrow \mathbb{R}$ which assigns a payoff to an action-outcome pair. Let this game have subjective and objective environment response functions $\bar{e}_i$ and e_i, respectively, which describe the belief an agent holds over future play paths and the true probability of a given play path. Let the local strategies be denoted σ_i and let infinite local play paths be z_i. Then the game will converge to a subjective correlated ϵ-equilibrium if the probability distribution over play paths induced by $\bar{e}_i$, $p(z_i|\sigma_i, \bar{e}_i)$ is compatible with the true distribution $p(z_i|\sigma_i, e_i)$ [Kalai and Lehrer, 1993].

Several differences exist between repeated games and the settings used in this work: Firstly, agents are situated in a cooperative environment rather than a self-interested one. This does not pose a problem, as the cooperative scenario can be seen as a special case of the self-interested one in which the utilities of all agents are aligned. Secondly, the scenarios considered here are stochastic games, while the convergence results are formulated for repeated games. Recall from Section 3.6.2 that these are a special case of stochastic games. It therefore remains to be shown that the convergence results can be extended to more general cases of stochastic games. Finally, in the settings considered here, agents obtain local observations at each time-step before making their action choices. In the repeated game setting described above there is no local information available other than the observed consequences. It again needs to be shown that this does not invalidate the convergence results.

The following sections will show that both formally and practically the convergence results stated can be extended to hold for the one-step scenarios presented here.

6.7.2 Theoretical Extension

Consider first the problem of adapting the repeated game framework to encompass local observations made by agents, which need to be made explicit in the description of the setting. For simplicity, consider scenarios in which agents have either full or no communication for now.

These local observations can be thought of as messages in a correlation device. Together with the transition probabilities this allows formulating a probability over future play paths. As previously, let A_i be a set of agent actions and C_i be a set of individual agent consequences. In addition let O_i be an individual set of observations and let $O = O_1 \times \ldots \times O_n$ be the set of joint observations. Let the local observations be correlated by a joint observation function $p_O : S \times O \rightarrow [0, 1]$. For the time being, consider scenarios in which agents have either full synchronisation or no communication at all. Assume as before that the centralised solution to the system is known. The decision-making problem for the time-steps with full communication is therefore simply solved by using the centralised policy and can hence be ignored here. Instead, consider a play path which consists only of non-communication time-steps. For one-step scenarios with action-independent transition probabilities this is straightforward: as there is no sequentiality between successive time-steps, the history of time-steps can be obtained from a general play path by discarding all time-steps in which communication took place. Let $\boldsymbol{h}_i^t = (o_i^1, a_i^1, c_i^1, \ldots, o_i^t, a_i^t, c_i^t)$ be a local history of length t and let Z be the set of all infinite play paths. Further, let agent i's strategy, i.e. its probability of action choice, after a history of length t and observation o_i^{t+1} be denoted by

$$\sigma_i^{t+1} = p(a_i^{t+1} | \boldsymbol{h}_i^t, o_i^{t+1}) \tag{6.23}$$

The subjective and objective environment response functions will now depend on the

correlation messages o_i^{t+1} received and therefore become

$$e_i^{t+1} = p(c_i^{t+1}|\boldsymbol{h}_i^t, o_i^{t+1}, a_i^{t+1}) \tag{6.24}$$

and $\bar{e}_i$ respectively. In a stochastic game the probability of making a joint observation depends on the current state of the system. The probability of making a local observation o_i is then given by

$$\begin{aligned} p(o_i) &= \sum_{\boldsymbol{o}} p(\boldsymbol{o}) p(o_i|\boldsymbol{o}) && (6.25) \\ &= \sum_{\boldsymbol{o}} \sum_{s} p(s) p(\boldsymbol{o}|s) p(o_i|\boldsymbol{o}) && (6.26) \end{aligned}$$

With this the probability distribution over finite play paths as induced by e_i and $\bar{e}_i$ can be expressed as:

$$\begin{aligned} p(\boldsymbol{h}_i^{t+1}|\boldsymbol{\sigma}_i^{t+1}, e_i t+1) &= p(\boldsymbol{h}_i^t|\boldsymbol{\sigma}_i^t, e_i^{t+1})\, p(o_i^{t+1}|\boldsymbol{h}_i^t) \\ &\quad \cdot p(a_i^{t+1}|o_i^{t+1}, \boldsymbol{h}_i^t)\, p(c_i^{t+1}|\boldsymbol{h}_i^t, o_i^{t+1}, a_i^{t+1}) \\ &= p(\boldsymbol{h}_i^t|\boldsymbol{\sigma}_i^t, e_i^{t+1})\, p(o_i^{t+1})\, \sigma_i^{t+1}\, e_i^{t+1} && (6.27) \end{aligned}$$

This modification still satisfies the two requirements for information sequences, as given in section 6.7.1: The information sequence defined by $\boldsymbol{h}_i^t$ remains to be one in which the partitions of Z_i are refined at each time-step. Similarly, the $\sigma-$ algebra generated by the information sequence is the same as the $\sigma-$ algebra on Z_i. Subject to fulfilling the requirement of absolute continuity, the convergence results will therefore continue to hold.

6.7.3 Implementation Practicalities

The requirement for absolute continuity is determined by how the decision-making algorithm is implemented in practise.

The set of possible consequences, given a local observation and a local action, can be obtained by considering all global states that are consistent with the current

observation and all possible actions of others. From this the set of possible rewards can be constructed. In this case the joint rewards are the consequences and the individual utility is given directly by the rewards for joint actions, $R(s, \boldsymbol{a})$.

Convergence will only be guaranteed if the absolute continuity condition is met. The most important task is therefore to choose the right prior distribution over possible consequences. To select a suitable functional form for this distribution, consider the Multinomial distribution, which gives the probability that out of N events with K possible outcomes, the outcome k will be observed m_k times:

$$\text{Mult}(m_1, \ldots, m_K|\boldsymbol{\rho}, N) = \binom{N}{m_1\, m2 \ldots\, m_K} \prod_{k=1}^{K} \rho_k^{m_k} \tag{6.28}$$

where $\boldsymbol{\rho}$ gives the probability of each individual outcome. If choosing only one event out of K possible ones only one m_k will be nonzero and the multinomial can be simplified. Let $\boldsymbol{x} = (0, \ldots, 0, , 1, 0, \ldots, 0)$ denote the outcome where the k-th possibility is realised. Then

$$p(\boldsymbol{x}|\boldsymbol{\rho}) = \prod_{k=1}^{K} \rho_k^{x_k} \tag{6.29}$$

since the binomial coefficient simplifies to one. In the settings considered here, $\boldsymbol{x}$ is an indicator variable which denotes that agent i receives consequence c_k at the next step. Together with the distributions over states and observations, this distribution enables the construction of a distribution over possible play paths. Since the transition and observation functions are predetermined, the choice of the distribution over consequences determines whether the continuity condition is met. In particular the choice of probability vector ρ decides which consequences have nonzero probability. As the "right" value of ρ is unknown, the Bayesian approach will be to assign a prior distribution over its value. When actual consequences are observed during the simulation, this data can then be used to calculate the posterior distribution over probabilities.

The conjugate prior to a Multinomial is the Dirichlet distribution, which gives the probability that the probabilities of K possible events are ρ_i, given that each event has

been observed $\alpha_i - 1$ times. Subject to the restraints $0 \leq \rho_i \leq 1$ and $\sum_i \rho_i = 1$ it has the functional form

$$p(\boldsymbol{\rho}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{K} \rho_i^{\alpha_i - 1} \tag{6.30}$$

where

$$B(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)} \tag{6.31}$$

is the multinomial Beta function and

$$\Gamma(\alpha_i) = \int_0^\infty x^{\alpha_i - 1} e^{-x} \, dx \tag{6.32}$$

is the Gamma function. To begin with, no consequence counts will have been observed and a suitable choice of α_i's must be made. Fortunately the requirement that each α_i be at least one will ensure that no consequences will be assigned zero probability under the prior and subsequent posteriors. Only in the limit of infinitely many data points will the posterior probability go to zero for those consequences which are never observed. This means that the belief over consequences will always be absolutely continuous with respect to the true distribution over consequences.

A good choice of α's can be obtained when constructing the set of possible consequences given a local observation and action: for all possible actions of others and possible global states, count the number of times each consequence is obtained and use these as counts for the Dirichlet prior. This amounts to the assumption that each action-state combination is equally probable. After the first time-step the agent will start to obtain actual counts of consequences and will need to update its belief accordingly. Let $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_K)$ be a vector of observed consequence counts. Then the posterior distribution over parameters is given by

$$p(\boldsymbol{\rho}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \text{Dir}(\boldsymbol{\rho}|\boldsymbol{\alpha} + \boldsymbol{\beta}) \tag{6.33}$$

Let $\boldsymbol{\alpha}_{ioa}^t = (\alpha_{ioa}^{t1}, \ldots, \alpha_{ioa}^{tK})$ denote the vector of counts observed for each consequence up to time t for a given observation o_i and action a_i and let $\boldsymbol{\alpha}_{ioa}^0 = (\alpha_{ioa}^{01}, \ldots, \alpha_{ioa}^{0K})$ be

the vector of counts used for this observation and action in the Dirichlet prior. Then the subjective environment response function at time t which gives the probability of obtaining a certain consequence under action a_i is given by (see e.g. [Bishop, 2006])

$$e_i(c_i^{t+1}|\boldsymbol{h}_i^t, o_i^{t+1}, a_i^{t+1}) = \int_{\rho_i} p(c_i^{t+1}|\rho_i) p(\rho_i|\boldsymbol{\alpha}_{ioa}^0 + \boldsymbol{\alpha}_{ioa}^t) d\rho_i \tag{6.34}$$

$$= \int_{\rho_i} \rho_i \text{ Dir}(\rho_i|\boldsymbol{\alpha}_{ioa}^0 + \boldsymbol{\alpha}_{ioa}^t)\, d\rho_i \tag{6.35}$$

$$= \mathbb{E}[\rho_i] = \frac{\alpha_{ioa}^0 + \alpha_{ioa}^t}{\sum_{j=1}^K (\alpha_{ioa}^0 + \alpha_{ioa}^t)_j} \tag{6.36}$$

where the division of vectors is carried out element-wise. The agent will use its belief over consequences to solve the decision-theoretic problem of finding the best local action. To do this it must calculate the expected reward under the current environment response function:

$$\mathbb{E}[u(a_i^{t+1})] = \int_{c_i} e_i(c_i^{t+1}|\boldsymbol{h}_i^t, o_i^{t+1}, a_i^{t+1}) u(a_i^{t+1}, c_i^{t+1})\, dc_i \tag{6.37}$$

In the cases considered here, all sets are finite and integrals are replaced with sums.

For a summary of the learning algorithm so far see algorithm 6.3.

Algorithm 6.3 Learning decision making algorithm

```
for i = 1 to numagents do
  for o = 1 to numobs do
    for a = 1 to numact do
      α_ioa ← count_possible_consequences ( )
    end for
  end for
end for
for t = 1 to T do
  o ← make_local_ observation ( )
  a ← choose_action(α_io)
  c ← obtain_rewards(a)
  α_ioa ← update_counts(o, a, c)
end for
```

6.7.4 The Exploration-Exploitation Trade-off

In general the prior distribution used by an agent to calculate expected rewards under a given action may be arbitrarily bad. This means that an agent might believe that one action is "good" and another is "bad" when in truth the average reward obtained under the bad action is higher. Unless the agent chooses what it believes to be the bad action at some point, it will not be able to correct this belief: the prior is only updated for those actions which have been tried out and for which feedback has been received. As a result an agent that greedily chooses the action it believes to be best at each time-step will in general be choosing non-optimal actions. On the other hand, an agent which deliberately chooses those actions that have lower expected reward under the prior in order to reduce the uncertainty associated with them risks discovering that these actions are indeed worse. In that case its overall obtained reward will be lower than it could have been under the best action. In the reinforcement learning field this is known that the exploration-exploitation trade-off: the optimal behaviour lies somewhere between greedy exploitation of the believed best action and exploration of other actions for which the beliefs might be poor. Unfortunately, using a full Bayesian formulation proves intractable for all but the smallest settings. Many approximate approaches to solving this trade-off have been developed (see e.g. [Kaelbling et al., 1996]). One such option is a heuristic referred to as "optimism in face of uncertainty". This method uses greedy action selection paired with a modified utility, which is given by the expected reward plus a term which encourages actions that have been explored less often. The algorithm developed here is motivated by this heuristic. Together with the expected reward given by Equation (6.37), the action chosen is the one which maximises

$$V(a_i) = \sum_{c_i} e_i(c_i^{t+1}|h^t, o_i^{t+1}, a_i^{t+1})u(a_i^{t+1}, c_i^{t+1}) + \lambda \cdot \sigma_{\mathbb{E}} \tag{6.38}$$

where λ is a positive scaling factor and $\sigma_{\mathbb{E}}$ is the standard deviation of the expected reward, that is a measure for the uncertainty over the value of the expected payoff. This acts as an exploration contribution to the utility: if the uncertainty about an action is large enough, the agent will start choosing it over those actions that have slightly higher expected reward but lower uncertainty.

The uncertainty over the expected reward stems from the uncertainty over the distributions of other agents' actions: the probability of each outcome and thus the reward given for a local action varies with the probability of the others' action choices. But the functional form in which the distribution over expected rewards depends on the distribution over others' actions is not straightforward. One way of obtaining the standard deviation $\sigma_{\mathbb{E}}$ used in Equation (6.38) is to use a sampling method: n possible probability vectors p_i are drawn from the Dirichlet distribution. These samples are in turn used to calculate n instances of the expected reward $\{R_1, \ldots, R_n\}$. For this the standard deviation can be calculated according to the standard formula.

Using the value function (6.38) rather than just the expected reward does not affect the validity of the convergence result, which holds for any measure of optimality by which to choose local actions.

6.7.5 Multiple Equilibria

The convergence result given in Section 6.7.2 guarantees that a system in which agents maintain Bayesian beliefs over possible consequences and act according to some decision-making algorithm, such as maximising expected reward, will ultimately move to a point which is $\epsilon-$close to a correlated equilibrium. What is more, each agent will be playing a stationary policy at equilibrium. In a similar single-agent system this information would suffice to guarantee that the agent would converge to the optimal policy: by successively improving its distribution over possible rewards and by (ul-

timately) choosing the action which is best under the belief, it will eventually come arbitrarily close to the optimum. In a multi-agent setting this is in general not the case. This is due to the existence of multiple equilibria which form local maxima in the reward space. Thus, it is not universally guaranteed that an arbitrary decision-making and learning process will converge to the globally optimal equilibrium. This is due to the interdependence of the agents' action choices: the action played by one agent at a given time during the learning process might influence the other agents' subsequent choices. This can lead to the system arriving at a local maximum at which the received reward is substantially lower than at the global maximum.

In particular, simultaneous exploratory actions as introduced in the previous sections can lead to suboptimal policies: for a learning agent it is impossible to tell whether a low obtained reward was due to its own suboptimal choice of local action or another agent's action choice during exploration. Thus, a local action which is part of an agent's optimal decentralized policy might appear to be a bad choice of action due to the other agent's simultaneous exploration. This in turn will lead to the agent adopting a policy which is a best response to the current exploratory behaviour of the other agent but which could potentially be far from optimal in the long run.

Very few multi-agent learning algorithms can guarantee convergence to the global optimum. Those that do are not applicable to the problem settings considered here, as they make fairly tight assumptions, such as complete observability of the global state or full communication. But it is possible to employ strategies that increase the probability of reaching the global optimum. One simple approach which can be used in the setting described here, is to allow individual agents to explore their action space during predefined phases of the simulation, while making greedy action choices according to their current belief at other times. This opens the possibility of scheduling individual agent learning into non-overlapping slots. Within each slot one

agent will momentarily be playing a stationary policy while the other agent can explore various local actions without these exploratory choices having an unwanted feedback. In the scenario presented here, agents were allowed to learn during predefined time-steps in the scenario, for example agent 1 during the first 100 time-steps, agent 2 during time-steps $100 - 199$, agent 1 during $200 - 299$ etc. This approach is not guaranteed to converge to the global optimum, but has proven very effective in the simulation presented below. For the complete decision-making algorithm including the modifications described above see algorithm 6.4

Algorithm 6.4 Full learning decision making algorithm

```
for i = 1 to numagents do
  for o = 1 to numobs do
    for a = 1 to numact do
      α_ioa ← count_possible_consequences( )
    end for
  end for
end for
for t = 1 to T do
  for i = 1 to numagents do
    o ← make_local_observation( )
    a ← choose_action(α_io)
    cons ← obtain_rewards(a)
    if is_turn == True then
      α_ioa ← update_counts(o, a, cons)
    end if
  end for
end for
```

6.8 Results for Learning Approximation

The learning algorithm has been applied to the first two benchmark problems from Chapter 5. Here there is no stochasticity in nature: given the full state of the system the best action and its respective reward are known. But because agents only have partial information about the global state and the local actions chosen by the others,

Algorithms		p-values		
		Setting 1	Setting 2	Setting 3
learning	decentralized	0.0	0.0	0.0
learning	coordinated	0.0	1.55×10^{-15}	0.0
learning	iterative	3.53×10^{-11}	0.0	0.0
learning	optimistic	0.0	0.0	0.0
learning	uniform	0.0	0.0	0.0
learning	pessimistic	0.0	0.0	0.0

Table 6.5: p-values obtained in a two-tailed Welch t-test comparing different algorithms in the tiger problem.

they do not know which reward a local action will yield with certainty. In this setting, rewards and consequences can be used synonymously. To compute the expected value according to Equation (6.38), a scaling factor of $\lambda = 3$ was used.

6.8.1 Tiger Problem

The algorithm was first applied to the tiger problem. Each simulation run consisted of 500 time-steps.

Figure 6.18 shows the performance of the learning algorithm compared to the previously considered algorithms in the tiger problem. Table 6.5 shows the respective significance results obtained in a two-tailed Welch t-test.

In all three settings of the scenario the algorithm performs close to but not as well as the decentralized, iterative and static algorithms. This suggests that after a sufficiently long period of learning (in which the average reward obtained is lower due to the exploratory actions), the algorithm could be able to find a near-optimal policy.

Figure 6.19 shows the value function under different actions according to Equation (6.38) for one agent in one of the scenario settings. The value function is higher during exploratory time-steps due to the additional uncertainty term (see Equation (6.38)). Note how the uncertainty decreases over time for those actions which are

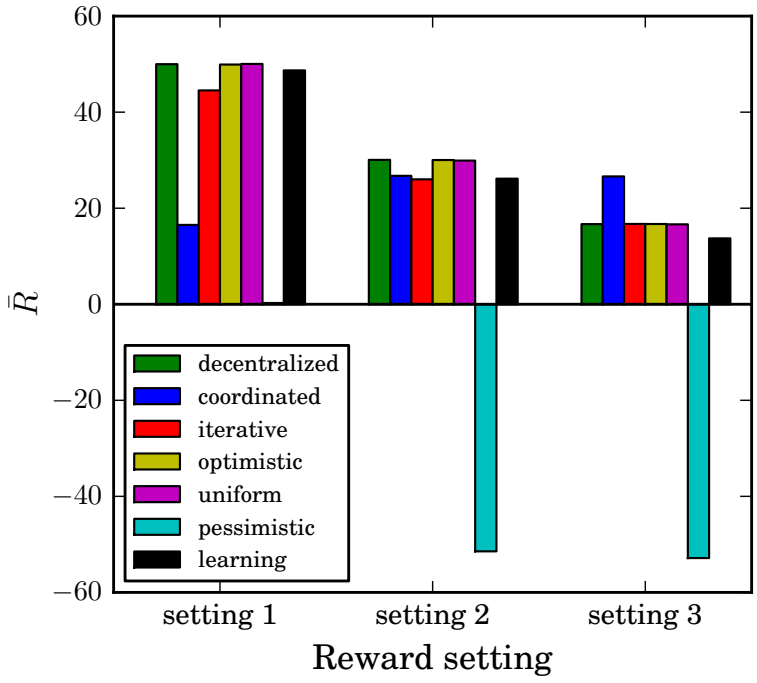


Figure 6.18: Average reward obtained under the decentralized, guaranteed coordinated, iterative, static and learning algorithms in the tiger problem. Data points were obtained by averaging over 500 simulations consisting of 500 time-steps each. Data for the learning algorithm was collected from the 100th time-step onwards allowing a learning phase at the beginning of each simulation.

chosen several times. The centre right figure (observation: reward centre) gives a good example for how the expected reward can change even after quite a few time-steps. During non-exploratory time-steps the agent chooses the action which has the current highest expected reward.

Figure 6.20 shows the actions chosen by one agent for some of the possible local observations over a simulation with 500 time-steps. Note how for some observations there are more exploratory actions taken than for others.

Finally, Figure 6.21 shows one agent's belief over consequences, under a given action at the last time-step of a simulation, compared to the frequency of consequences observed under that action throughout the simulation. Due to the Dirichlet prior distribution, which assigns nonzero probability to all possible consequences, the agent

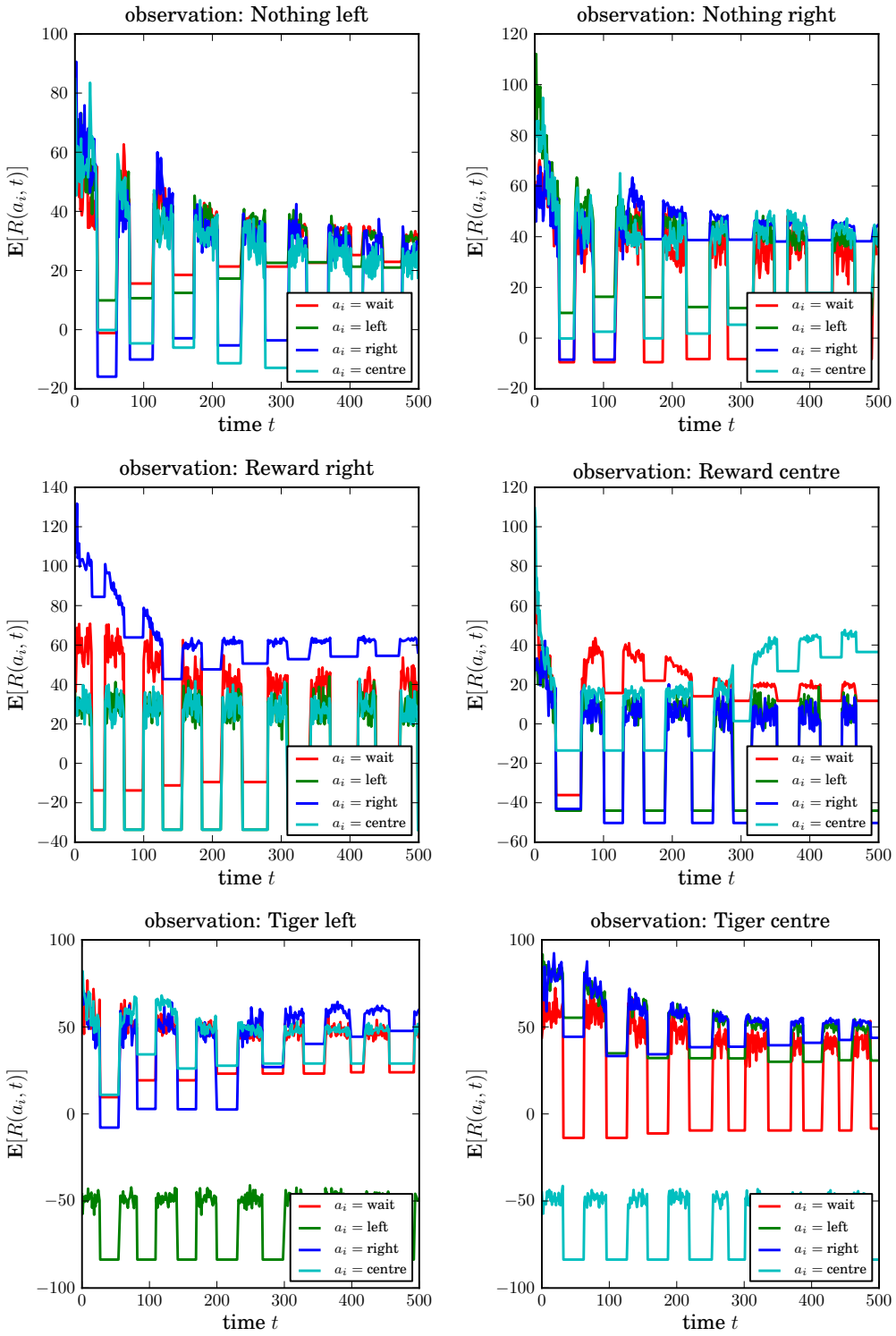


Figure 6.19: Development of one agent's value function under all available actions for some of the possible local observations in the first setting of the tiger problem. Data was obtained over 500 time-steps for each observation.

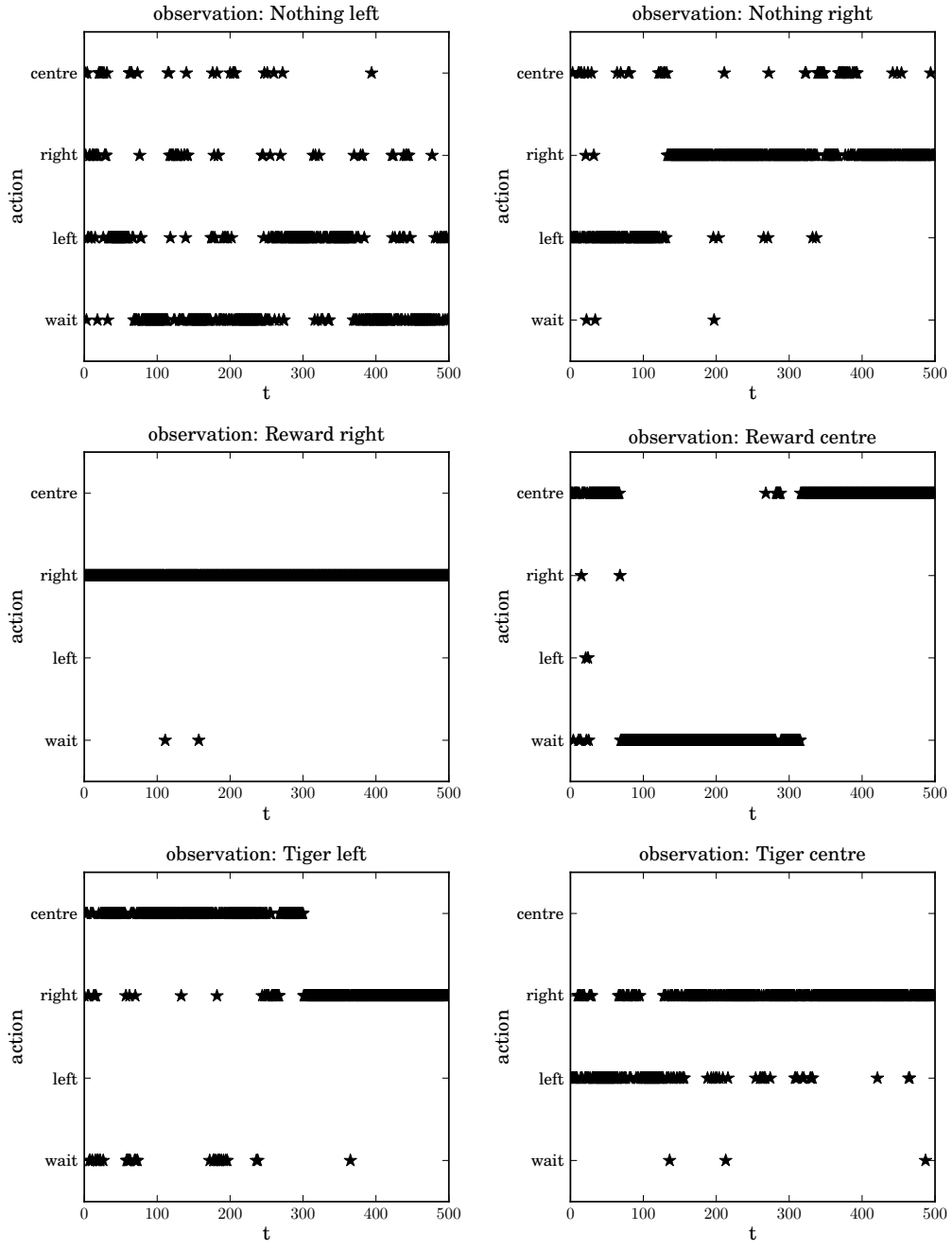


Figure 6.20: Actions chosen by the first agent in the first setting of the tiger problem throughout a simulation with 500 time-steps for some of the possible observation. Deviations from the main line show exploratory actions taken.

assigns a small posterior probability to all consequences even if they were not observed during the simulation. For most consequences the belief is very close to the

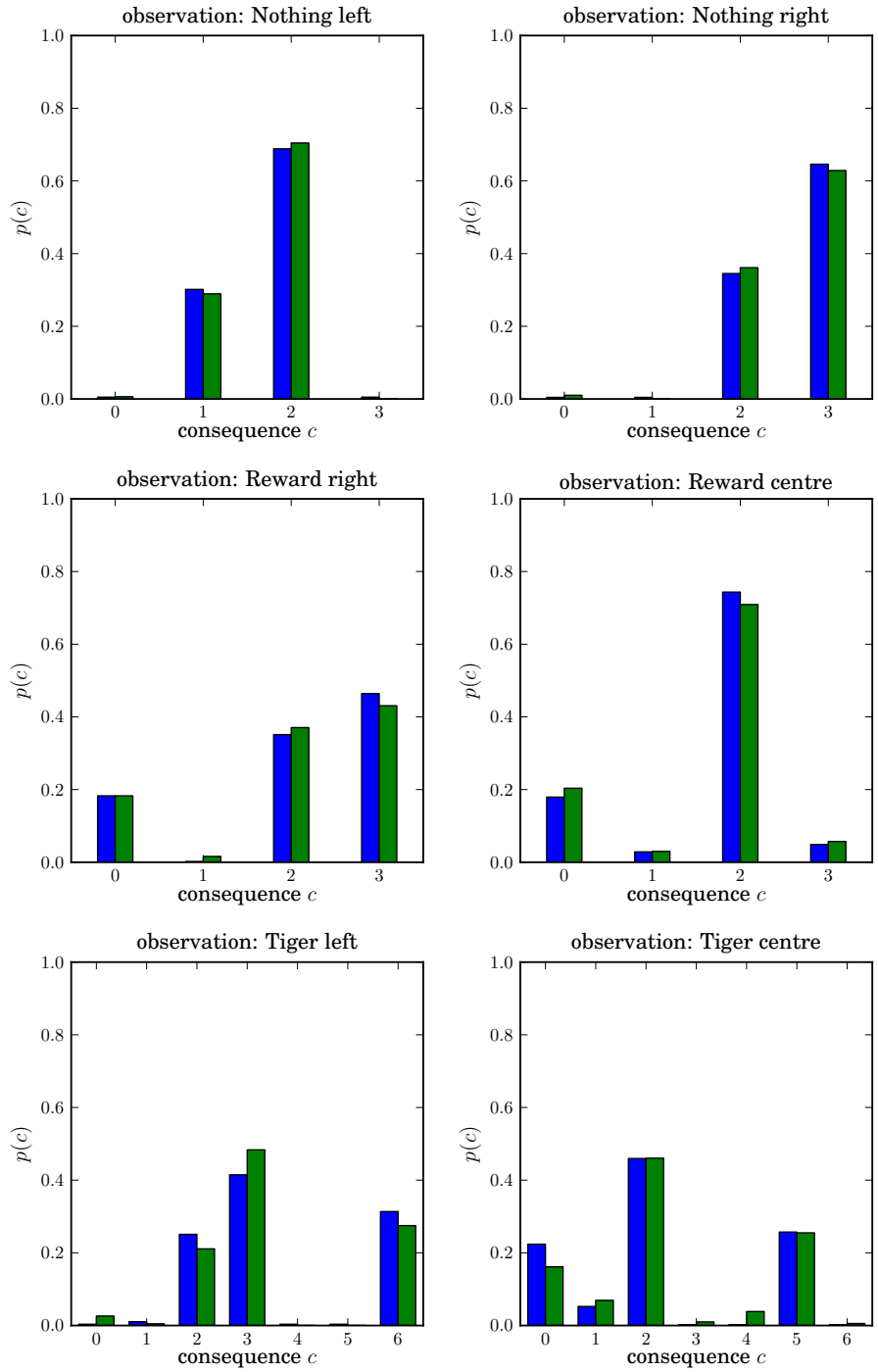


Figure 6.21: Comparison of one agent's belief over consequences (blue, left column) to the experimentally observed frequency with which a consequence occurs (green, right column) for some of the possible observations and the respective best responses in the tiger problem. Data points were taken after 1000 time-steps. The difference in numbers of consequences per observation is due to different action-state combinations (and thus Rewards) being possible depending on the observation and the corresponding best action.

observed frequency.

6.8.2 Meeting Problem

The learning algorithm was next applied to the meeting problem. Again a simulation run consisted of 500 time-steps.

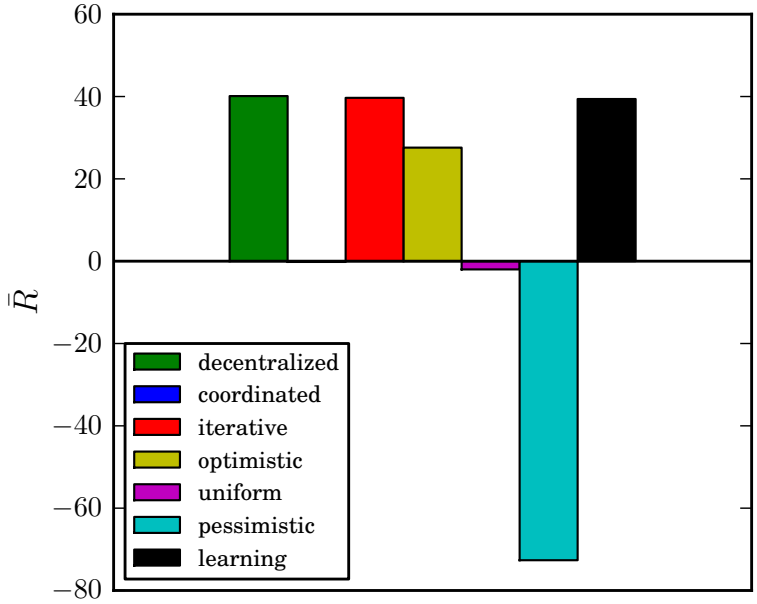


Figure 6.22: Average reward obtained per time-step under the decentralized, guaranteed coordinated, iterative, static and learning algorithms in the meeting problem. Data points were obtained by averaging over 500 simulations consisting of 500 time-steps each. Data for the learning algorithm was collected from the 100th time-step onwards with a prior learning phase.

Figure 6.22 shows the average reward obtained under the learning algorithm compared to the previously considered algorithms in the meeting problem. Table 6.6 gives the respective p-values obtained in a two-tailed Welch t-test.

In this setting the learning algorithm outperforms the coordinated and static algorithms. Again it performs nearly as well as the decentralized and iterative algorithms.

The development of one agent's value function for all possible observations in the scenario is shown in figure 6.23. Note that there is only a very short time-span at the

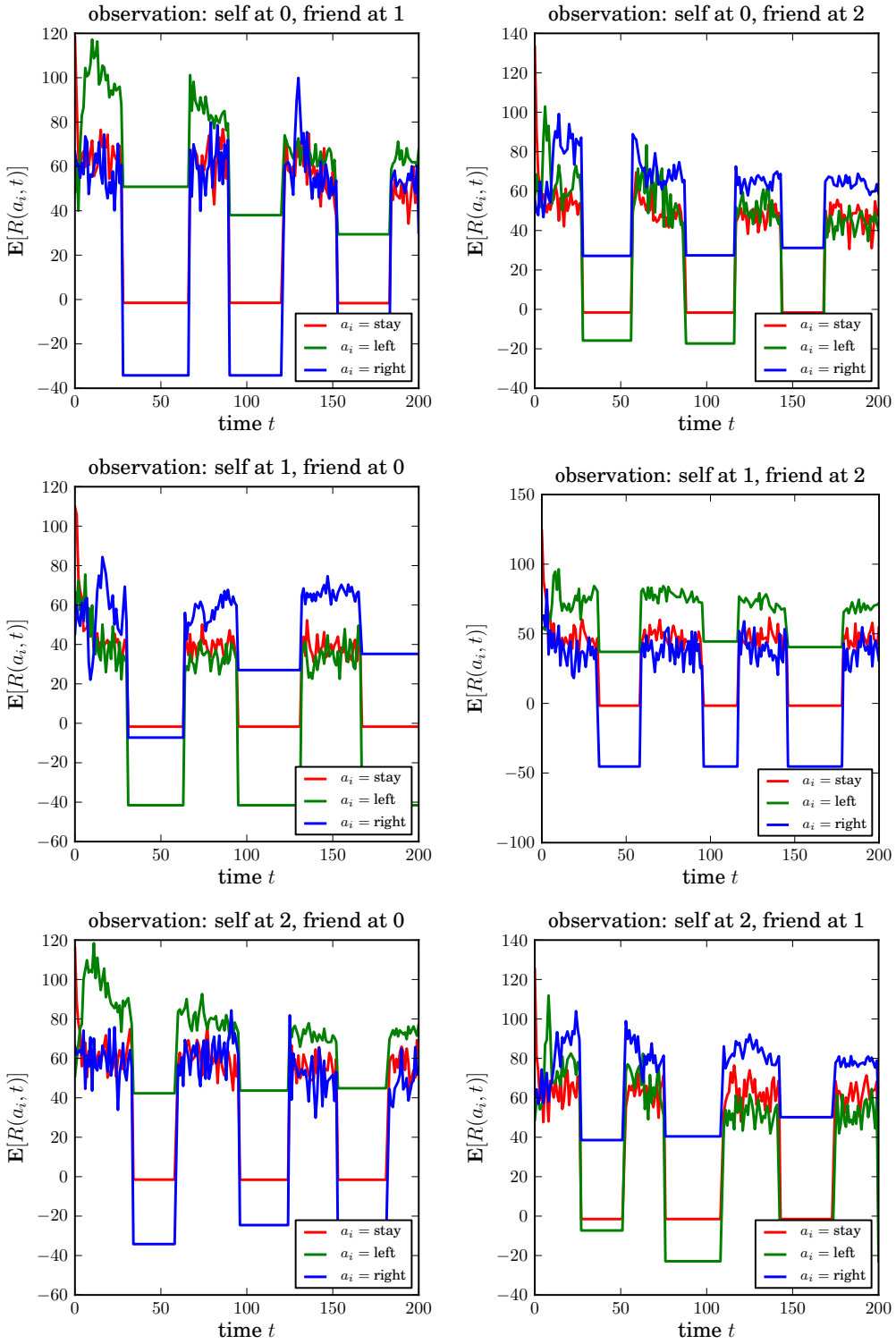


Figure 6.23: Development of one agent's value function under all available actions for all possible local observations in the meeting problem. Data was obtained over 200 time-steps for each observation.

Algorithms		p-values
learning	decentralized	7.09×10^{-3}
learning	coordinated	0.0
learning	iterative	0.30
learning	optimistic	0.0
learning	uniform	0.0
learning	pessimistic	0.0

Table 6.6: p-values obtained in a two-tailed Welch t-test comparing different algorithms in the meeting problem.

beginning during which the agent explores the action space to reduce its uncertainty and find the best action. This is due to the greater difference in expected reward under the individual actions in this setting and explains why the performance of the algorithm is so good, even after a relatively short learning period of 100 time-steps at the beginning.

Figure 6.24 shows the actions chosen by one agent throughout a simulation of the scenario. In comparison to the tiger problem fewer exploratory actions are taken throughout the run, again suggesting that the convergence is faster in this scenario. Figure 6.25 shows the agent's belief over consequences for a given action at the last time-step compared to the observed frequency of consequences under the same action. Again, the beliefs are very close to the empirical frequencies, showing that the algorithm is converging.

6.8.3 Monitoring Problem

Whilst the learning algorithm is in principle applicable to the monitoring problem, the large observation space (that comes with a scenario with general communication) limits its usefulness there: Each communication pattern defines a separate observation for which the agent must learn the distribution over consequences. This will require increasingly longer exploratory periods for each agent to ensure that enough data is

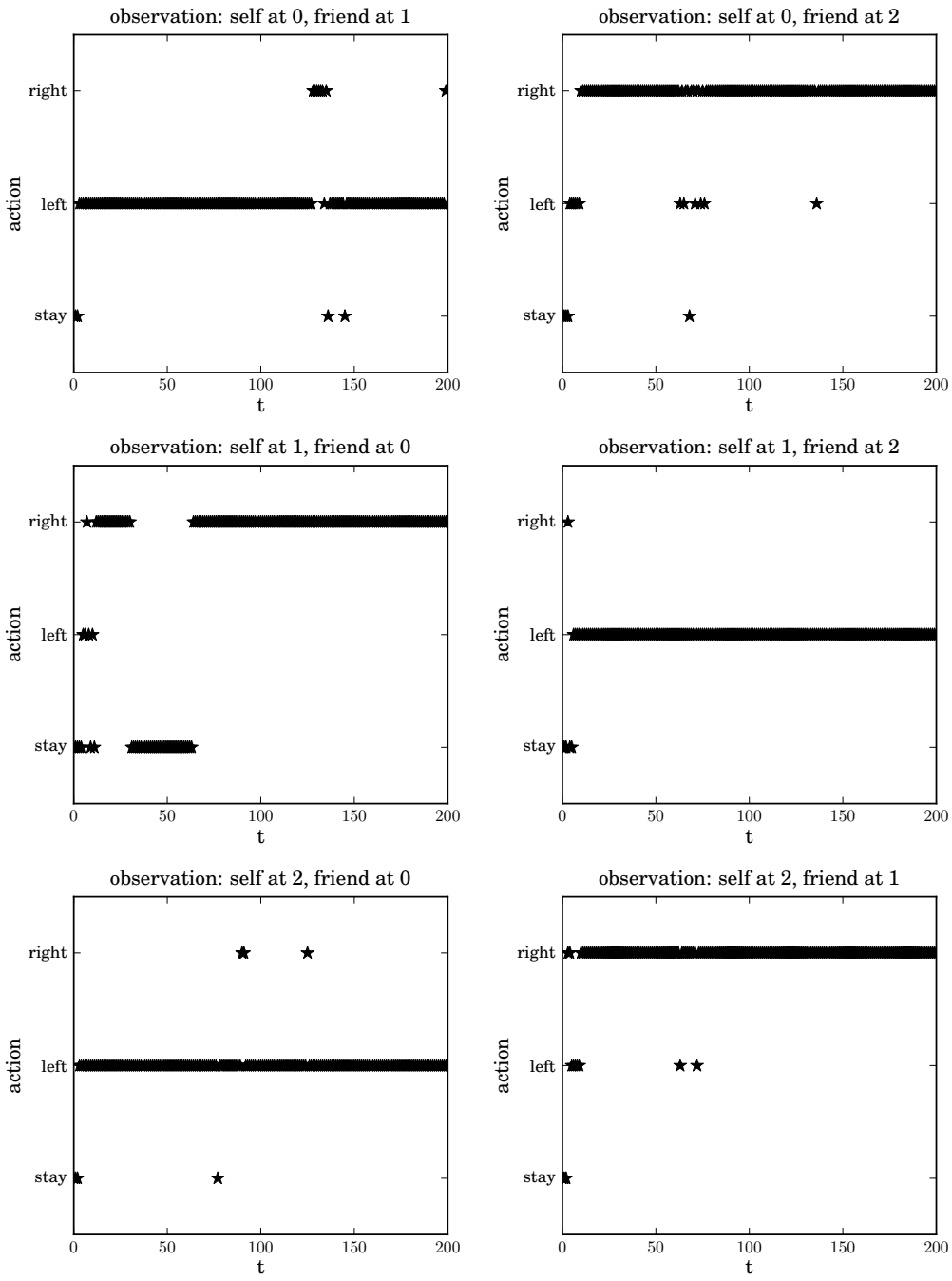


Figure 6.24: Actions chosen by the first agent in the meeting problem throughout a simulation with 200 time-steps for each observation. Deviations from the main line show exploratory actions taken.

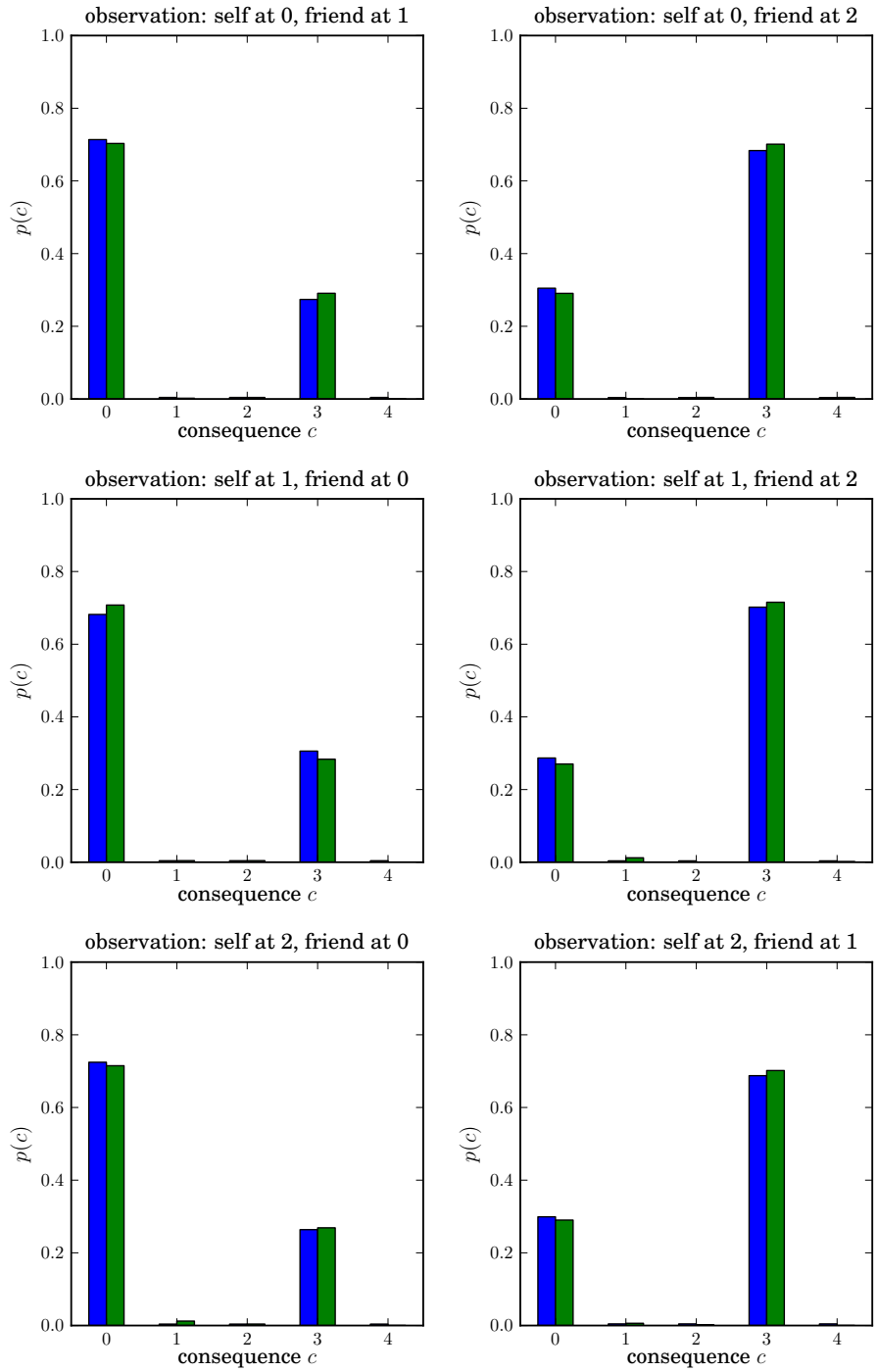


Figure 6.25: Comparison of one agent's belief over consequences (blue, left column) to the experimentally observed frequency with which a consequence occurs (green, right column) for all possible observations and the respective best responses in the meeting problem. Data points were taken after 500 time-steps. The difference in numbers of consequences per observation is due to different action-state combinations (and thus Rewards) being possible depending on the observation and the corresponding best action.

observed to update the distributions meaningfully. in practise this will often mean that policies will not converge within the time-frame given by the setting. The learning algorithm was therefore not applied to the monitoring problem here.

6.8.4 Discussion

The application of the learning algorithm to two of the benchmark problems showed good performance and convergence results. The average reward obtained was close to that of the best static algorithm in the tiger problem and near-optimal (and better than the static algorithms) in the meeting problem. Thus, the learning algorithm can indeed yield performance improvement over the more simple heuristic approach in some settings. However, its applicability is limited to scenarios with small observation spaces and/or simple communication patterns. Ultimately this means that it will not be feasible for many interesting scenarios.

6.9 Conclusions from One-step Scenarios

This chapter has presented three different approximate solutions to local decision-making and tested their performance on the benchmark scenarios. In the majority of settings of the benchmark problems which feature synchronised communication these approximate algorithms performed better than the benchmark algorithm which guarantees agent coordination. This analysis has therefore shown that using such an algorithm is not always (and possibly only rarely) the best solution to scenarios with sparse communication. For more flexible communication patterns (e.g. individual communication links between agents) the effect is even more pronounced: here, the heuristic approximate algorithm outperformed the decentralized algorithm for even small values of communication probabilities.

Among the approximate algorithms considered, the static approach based on a

simple heuristic is the one which produces the least computational overhead and for which all scenarios and communication settings remain tractable. Its simulation results required on the order of hours to compute. It also yields surprisingly good performance given its simplicity. Both the iterative and the learning algorithm required somewhat longer run times, on the order of days for some of the more complex settings. This makes them less well-suited for some scenarios, even though they can increase the performance for some of them.

Chapter 7

Application to Recording Task

7.1 Introduction

The previous chapter explored several possible approximate algorithms for local decision-making and compared how these perform on three different benchmark tasks. This analysis has shown that the static algorithm based on a heuristic approximation performs well in most of the test environments while incurring comparatively low computational cost. The benchmark settings used were designed to reflect different characteristics of multi-agent systems on which to test the performance of the approximate algorithms. Even so, the simulations were based entirely on artificial data. While this is a justifiable method for a first analysis, a demonstration of the method's performance on real data is desirable as a further test of its usability. This chapter will therefore present the results of applying the static algorithm to a non-synthetic data set.

7.2 Problem Setting

The data set used for this analysis consists of pedestrian tracks which were extracted from recordings of a foyer area collected by a single camera [Majecka, 2009]. Footage was collected at the Informatics Forum, the main building of the School of Informatics at the University of Edinburgh. This area is frequented by pedestrians and has a number

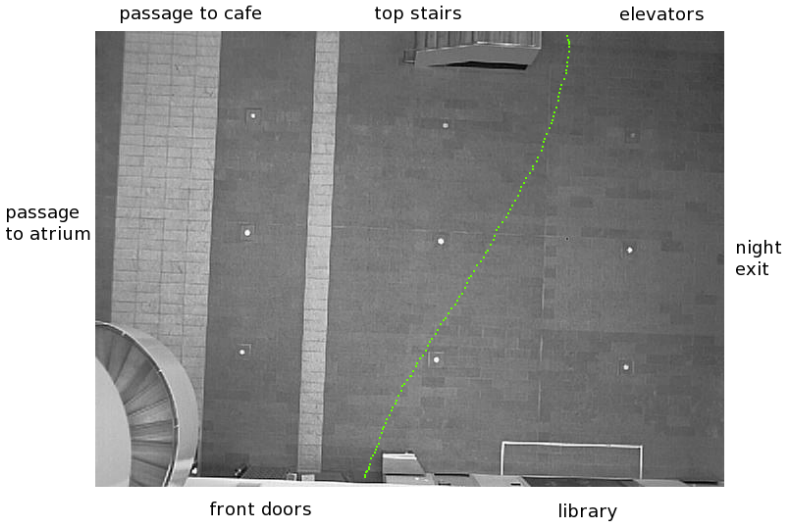


Figure 7.1: Sample image of the area used for recording pedestrian data showing a single pedestrian track. The main entrances and exits are clockwise: the front doors (bottom left), passage to the atrium (centre left), passage to the cafe (top left), top stairs (top centre), elevator(top right), night exit (centre right) and passage to the library (bottom right).

of entrance and exits points such as doors, stairs and lifts. Pedestrians passing through this area thus generate a pattern of characteristic tracks. For a sample image of the recorded area and details of the entry points see Figure 7.1. The area was recorded with an average frame rate of around 9 frames per second, depending on the local Ethernet and capture host machine loads. For each passing pedestrian the recording yields a sequence of images from which their individual track may be reconstructed. This is achieved through a series of detection algorithms which ultimately produce the coordinates of the targets' centre of mass for each frame. Extracting this data also requires separating real targets and tracks from objects mistakenly identified as targets. Data was collected over several months with an average of 1000 observed tracks per day. This has resulted in a total of over 92000 recorded real target tracks.

A subset of this data set was used to simulate the performance of a system of four agents monitoring the foyer area. Each of the agents receives track data obtained

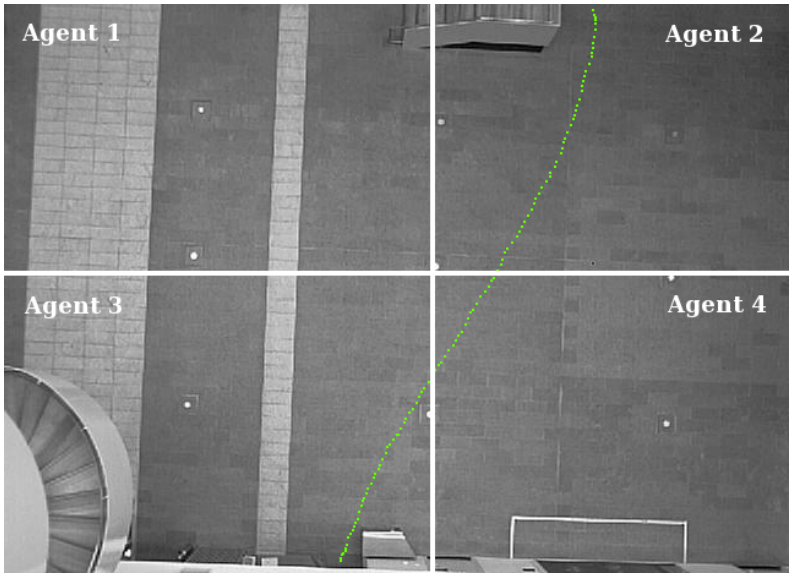


Figure 7.2: The overall surveillance area divided between four agents with non-overlapping fields of view. The green dots show a sample track generated by a pedestrian crossing the foyer. In general such a track will pass through more than one agent's field of view.

from a fixed region of the foyer. In practise this can be thought of as each agent being equipped with a camera whose field of view is smaller than the total area. The overall surveillance area is divided up between these agents with no overlap between individual fields of view and no areas unmonitored. For the simulation described here the fields of view were modelled as four rectangular local surveillance areas as shown in figure 7.2. Pedestrians moving across the foyer will in general pass through several agents' field of view. To record a complete track, footage from more than one camera is therefore needed.

The area considered here can easily be monitored by a single camera and dividing it into separate agent domains might seem artificial. However, in many surveillance settings the area of interest will not constitute a simple rectangle but rather a more complicated geometry which might include pillars, walls, corridors or other obstructions. In such settings a single camera will not suffice. The floor plan used here can

therefore be thought of as representative of such areas.

Assume that the aim of deploying these agents is to record track data. Each agent is equipped with a small amount of data storage, which will allow it to write individual frames to file. However, the capacity of this storage device is limited and as a result not all footage can be written. Instead, the agents are limited to saving only certain tracks which are considered of particular interest, for example those traversing between the library and the night exit. To record these tracks in full each agent must save those frames during which the pedestrian is located within its local surveillance area.

To ascertain whether a trajectory belongs to those that should be written, the pedestrian's entry and exit points are needed. As these will in general not lie within the same agent domain, an individual agent will not have sufficient information to make this decision independently. To ensure that complete footage of an interesting track is written, coordination between agents is therefore necessary.

The agent coordination problem outlined above has been mapped onto the theoretical problem settings considered in previous chapters in the following way:

The set of possible characteristic trajectories through the area is defined by the number of entry and exit points. This will be interpreted as the set of states: a pedestrian passing through the area may be taking any of the possible characteristic "paths" that lead through it, i.e. it may be in any of the possible states. On a local agent level only a segment of this track will be observed. Each agent through whose local surveillance area a pedestrian passes will thus observe entry and exit points somewhere along the boundary of its field of view. To discretise the set of local observations, the borders of all agents' fields of view were divided into virtual entry and exit regions. For a visualisation of the local regions of one of the agent see Figure 7.3. With this division an agent's local observations can be generated in the same way as the global states, as the possible trajectories between local entry and exit regions. Furthermore,

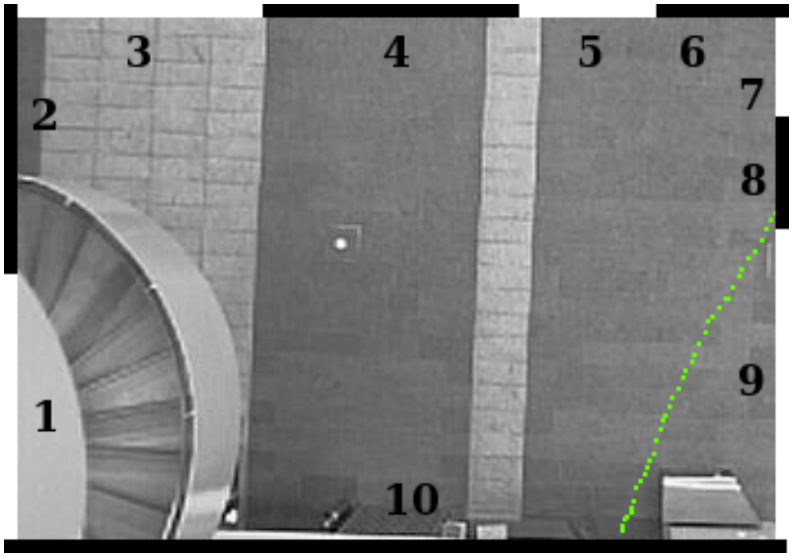


Figure 7.3: Local observation regions for one of the agents in the system: ten entry/exit regions divide the boundary of the surveillance area into discrete sections. Tracks which enter and exit through the same observation regions will generate the same local observation. In this example the local observation is given by the combination of observation regions 8 and 10.

a track which does not pass through an individual agent's field of view generates an empty observation. Based on the local observations the set of joint observations is then given by those combinations of local observations which correspond to actual observed tracks. To obtain these one starts from the Cartesian product of local observations and discards all elements which are physically impossible. These are the joint observations which do not satisfy continuous boundary conditions, i.e. in which a track would "jump" between observation regions when passing from one agents' field of view to the next. In addition, those tracks which are physically possible but very unlikely are also discarded. As an example consider a pedestrian entering through the front doors on the bottom left and passing through all agents' fields of view in a large circular movement before exiting into the library on the bottom right. Such a track is physically possible but highly improbable, as there is a much shorter path between the two points. Discarding such tracks therefore implicitly assumes that pedestrians do not deviate too

strongly from the shortest path between any two entry/exit points.

As in the settings studied in previous chapters, there exists no unique mapping from local observations to global states. Consider the track shown in Figures 7.1 through 7.3 with a pedestrian crossing from the front door to the lift. This track leads through the sample agent's observation regions 8 and 10. But a pedestrian passing between the front door and the night exit (at centre right) would produce a very similar track segment, which might result in the same local observation. From the local observation alone it is therefore not obvious which global track the pedestrian is taking. If, on the other hand, the full joint observation is known, the global state is uniquely defined by it. In the multi-agent decision process-framework this is expressed through local and global observation probabilities. To obtain values for these, recordings from several busy days were used to calculate the empirical frequency with which each observation occurred given the global track.

For simplicity agent tracks are assumed to be sparse (i.e. they can be considered individually) and independent of each other (i.e. situations in which one pedestrian follows another or a group moves together are rare). Indeed, roughly half of the recorded frames contain only one pedestrian, which suggests that sparseness is a reasonable assumption. As a result of the independence the transition probabilities between two states (i.e. between two pedestrians) do not depend on the particular tracks taken by agents. It is not obvious that this is the case in reality, as pedestrians could be following each other and the assumption could easily be relaxed by using a transition function which has a dependency between successive states.

For each local observation an agent has the choice between two actions: write the footage or discard. As in the benchmark problems considered in the previous chapters the challenge lies in choosing the best local actions given the available information. If agents can communicate between each other, this decision becomes trivial and the

full track footage will be written or discarded depending on whether the track is of interest or not. With only partial or no communication between agents, the action choice must once more be made under incomplete information. The optimal joint action for the centralised case is for all agents to write those segments to file that correspond to interesting tracks. The decentralized solution should therefore result in as many complete interesting tracks as possible without storing unnecessary segments from uninteresting tracks.

7.3 Results

The static local decision-making algorithm (see section 6.5) was applied to the recording task described in the previous section.

To employ the algorithm, each joint action needs to be associated with a reward. This can be generated from the problem setting in the following way: due to the limited storage space there is a cost associated with writing a track segment to local storage. A reward is obtained for each track segment of interest which is written completely and a penalty is incurred when an agent does not write a track segment which would have been of interest. Not writing a track which is not of interest yields a reward of zero. These items can be slightly re-written in terms of a joint reward for writing complete tracks, a joint reward for writing partial tracks, a penalty for writing unnecessary track segments and a penalty for not writing segments which are of interest. Note that this allows non-linear reward settings in which joint rewards are not just the sum of rewards for individual actions. For example correctly writing a track in full might yield the reward R, missing one segment might yield $0.8R$, missing two $0.4R$ and missing three or more $-R$. The (relative) magnitude of the actual rewards depends on the particular characteristics of the setting. For the simulations presented here three different reward matrices were used, see Figure 7.4.

Action	Reward
complete	100
partial	30
unnecessary	−10
skipped	−15

(a) setting 1

Action	Reward
complete	100
partial	20
unnecessary	−20
skipped	0

(b) setting 2

Action	Reward
complete	100
partial	0
unnecessary	−70
skipped	−20

(c) setting 3

Figure 7.4: Different reward settings used in the simulation of the recording task. "complete" refers to correctly written tracks, "partial" to interesting tracks which are partially written, "unnecessary" to unwanted track segments which are written and "skipped" to segments which are interesting but not written.

For this simulation four tracks were chosen as interesting: between the labs and the atrium, between the atrium and the night exit, between the labs and the lift as well as between the front door and the cafe.

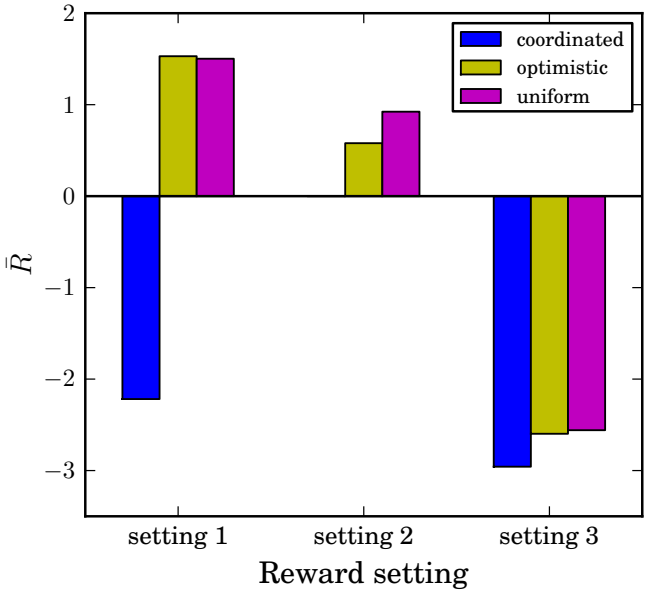


Figure 7.5: Average obtained reward per track for the three different reward matrices under the guaranteed coordinated algorithm as well as the static algorithm with uniform and optimistic distributions. In this setting no communication took place between agents. Data points were obtained from a total of 1280 tracks.

Algorithms		p-values		
		Setting 1	Setting 2	Setting 3
optimistic	coordinated	0.0	2.88×10^{-5}	0.29
uniform	coordinated	0.0	2.34×10^{-13}	0.25
optimistic	uniform	0.95	0.06	0.91

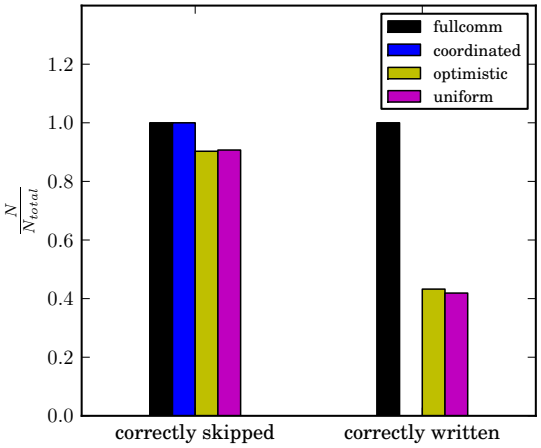
Table 7.1: p-values obtained in a two-tailed Welch t-test comparing the reward performance for different algorithms in the recording task

Skipped tracks				
Algorithms		p-values		
		Setting 1	Setting 2	Setting 3
coordinated	fullcomm	-	-	-
optimistic	fullcomm	0.0	1.81×10^{-10}	6.01×10^{-5}
uniform	fullcomm	0.0	6.01×10^{-5}	0.02
optimistic	coordinated	0.0	1.81×10^{-9}	6.01×10^{-5}
uniform	coordinated	0.0	6.01×10^{-5}	0.02
optimistic	uniform	0.73	1.16×10^{-3}	0.02
Written tracks				
Algorithms		p-values		
		Setting 1	Setting 2	Setting 3
coordinated	fullcomm	-	-	-
optimistic	fullcomm	4.66×10^{-15}	0.0	0.0
uniform	fullcomm	1.33×10^{-15}	0.0	0.0
optimistic	coordinated	1.17×10^{-10}	0.15	0.15
uniform	coordinated	2.83×10^{-10}	0.15	0.15
optimistic	uniform	0.87	1.0	1.0

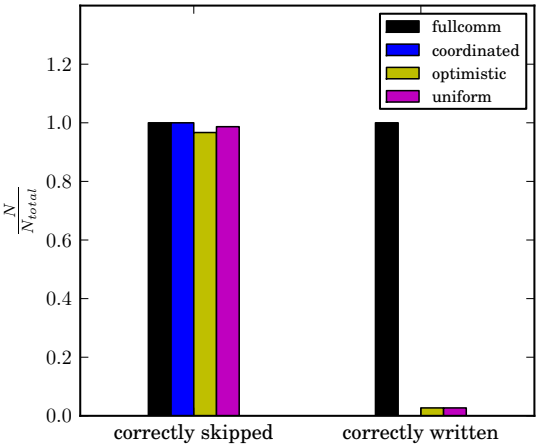
Table 7.2: p-values obtained in a two-tailed Welch t-test comparing the writing performance for different algorithms in the recording task

7.3.1 Zero Communication

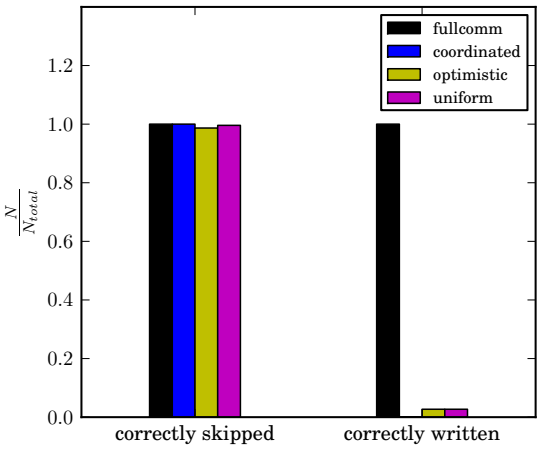
As in the previous chapter, the simulation was first carried out with no communication at all taking place between agents. For this communication setting the performance under the fully decentralized algorithm (see Section 5.3.1) defines the best possible



(a) setting 1



(b) setting 2



(c) setting 3

Figure 7.6: Fraction of tracks correctly written in total and correctly skipped in total under full communication, the guaranteed coordinated algorithm and the static algorithm with uniform and optimistic distributions. In this setting no communication took place between agents. Data points were obtained from a total of 1280 tracks.

performance. However, this setting is too complex for the decentralized JESP algorithm described there to remain tractable. It is therefore not available as a benchmark algorithm for this scenario. Figure 7.5 shows the average reward obtained per track under the static algorithm compared to the guaranteed coordinated algorithm. Table 7.1 shows the respective p-values obtained under a two-tailed Welch t-test comparing the average performance of the individual algorithm.

For all three reward settings both flavours of the static algorithm outperform the guaranteed coordinated one. There is a smaller difference between the performance of the uniform and optimistic variants of the algorithm with the optimistic outperforming in the first setting and the uniform performing better in the second and third. Note that while the reward obtained depends strongly on the reward setting, it is not the measure which is most expressive when judging the performance of the algorithm. Ultimately, the fraction of interesting tracks written and the amount of unnecessarily written ones determine the performance in a practical sense.

Figure 7.6 shows the average correctly written and correctly skipped tracks under the different decision-making algorithms for the three different reward matrices. Table 7.2 again gives the respective p-values obtained under a two-tailed Welch t-test.

For all three settings the guaranteed coordinated algorithm discards all tracks regardless of whether they are of interest or not. This is again because the algorithm does not allow for use of local information about a particular track. Instead all agents use the prior belief over whether a track will be of interest or not to make the decision. Both flavours of the static algorithm show similar behaviour in the second and third reward settings, with only a very small fraction of tracks correctly written to memory. For the first reward setting the fraction of correctly saved tracks is considerably higher under both flavours of the static algorithm. Note that this is achieved without the fraction of correctly discarded tracks being reduced by the same proportion.

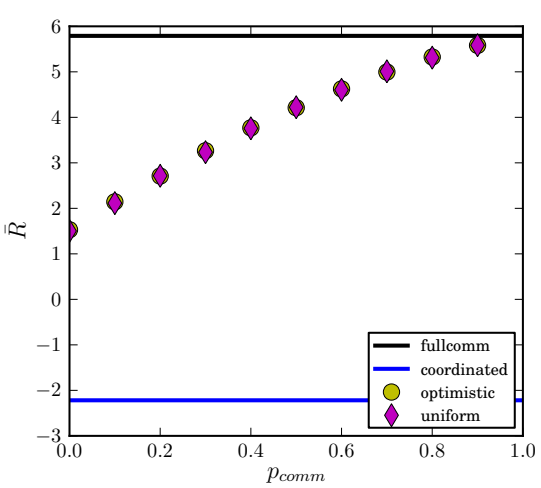
Figure 7.6 demonstrates how the actual rewards associated with actions play a more important role in settings with sparse information exchange between agents: while the performance of the centralised algorithm (full communication) is the same for all three reward settings, performance varies between reward settings for the approximate algorithms. This is because individual agents do not have full information about whether or not an observed track is of interest. The decision about whether to write

it will therefore depend on the cost associated with that action as well as an agent's belief about the nature of the track.

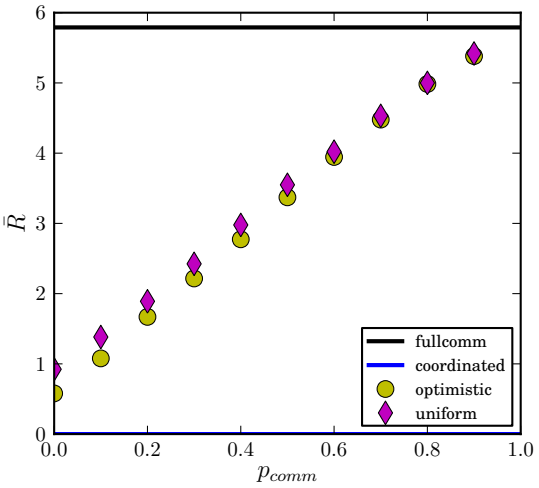
7.3.2 Variable Communication

As the scenario studied here comprises more than two agents, other communication patterns besides full and no communication are also possible. For this analysis, a setting in which agents have individual communication links was also simulated. This means that at the time of making a decision an agent may have obtained information from any number of other agents, ranging from none to all others. The probability of a successful communication was assumed to be the same for all agents and static throughout a simulation run. Figure 7.7 shows the average reward obtained per track under the different decision-making algorithms as a function of the communication probability between any two agents. As with the one-step monitoring scenario studied in section 6.6.3 the average reward increases steadily with the communication probability. In the limit of near-certain communication it approaches the reward obtained under the centralised algorithm. For this communication pattern there is a slight difference between the uniform and optimistic flavours of the static algorithm with the optimistic performing somewhat better.

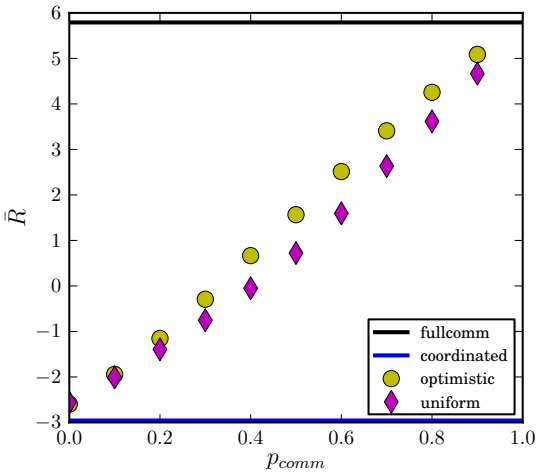
Again it is informative to take a look at the actual written tracks. See Figure 7.8 which shows the fraction of tracks correctly written and correctly skipped as a function of communication probability. As would be expected from Figure 7.6, the performance under the static algorithms is similar for the second and third reward settings with very few complete tracks written correctly for low communication probabilities. This fraction increases with the communication probability while the fraction of correctly skipped tracks is nearly constant. For the first reward setting the fraction of correctly written tracks under the static algorithms is higher for low communication probabilities



(a) Setting 1 with p-values from $0.66 - 0.96$.



(b) Setting 2 with p-values from $0.0 - 0.85$.



(c) Setting 3 with p-values from $0.0 - 0.42$.

Figure 7.7: Average obtained reward per track under the static approximation (markers) as a function of communication probability for agents with individual communication links between any two agents. The performance under full communication (black line) and for the guaranteed coordinated approach under zero communication (blue line) are given for comparison. Data were obtained for a total of 1280 tracks and 100 simulations. P-values were obtained in a two-tailed Welch test comparing the optimistic and uniform algorithms.

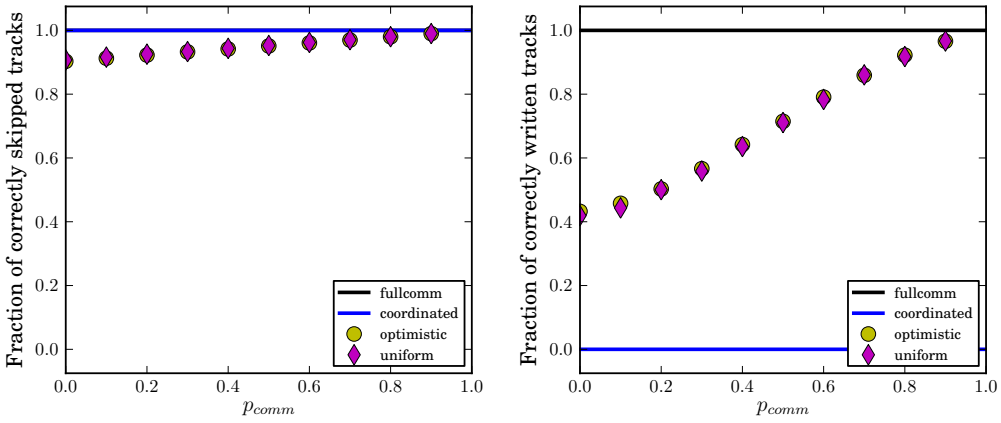
and increases with the communication probability. The fraction of correctly skipped tracks also increases with the communication probability, but from a much higher starting point. As with the zero-communication setting the difference between the

optimistic and uniform approximations is not very pronounced, suggesting that the action choices under both algorithms are very similar.

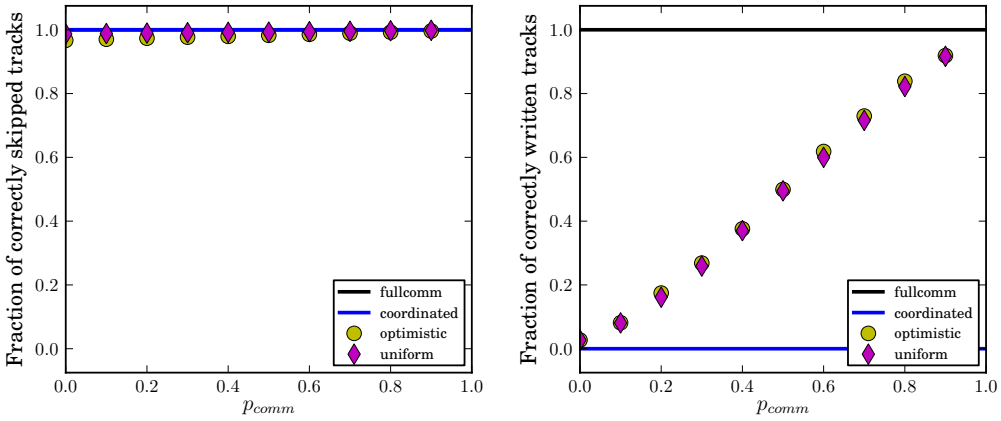
7.4 Discussion

The results presented in the previous section show that the static approximate algorithm for decision-making also performs well in this example setting based on non-synthetic data. This can be seen as further evidence for its usability and good performance behaviour. For all settings of the reward matrix studied here the algorithm performs as well as or better than the benchmark algorithm which guarantees coordination. The results presented here therefore serve as further anecdotal evidence that for many applications strict agent coordination might be less favourable than the use of local information for decision-making. Further, the simulation results for a setting with individual communication links between agents underline how the approximate algorithm makes use of all locally available information at any given time and how it approaches the performance of a centralised algorithm with increasing information exchange between agents.

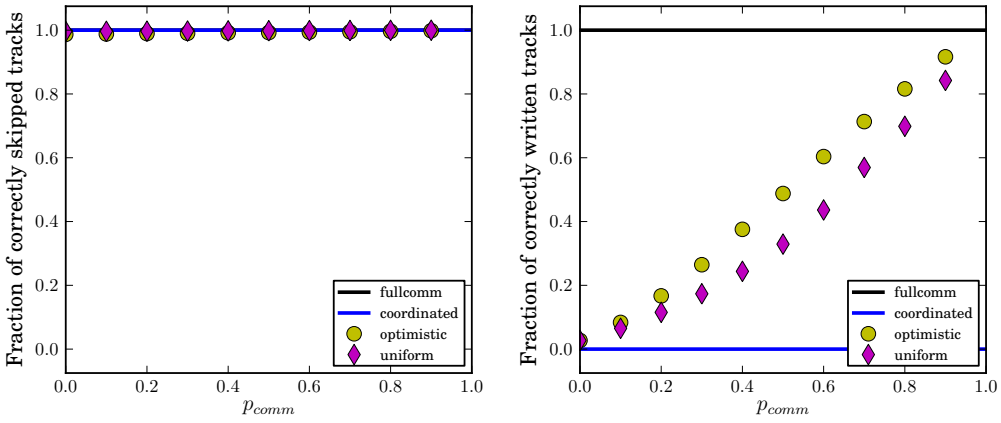
As this setting is more complex than those considered in the previous chapter, the computation times were also increased: producing the simulation results presented here required on the order of days.



(a) Setting 1. P-values of $0.0 - 0.87$ and $0.0 - 0.82$ for skipped and written tracks respectively.



(b) Setting 2. P-values of 0.0 and $0.003 - 1.0$ for skipped and written tracks respectively.



(c) Setting 3. P-values of 0.0 and $0.0 - 1.0$ for skipped and written tracks respectively.

Figure 7.8: Average fraction of complete tracks correctly skipped and correctly written under the static algorithm (markers) as a function of communication probability between any two agents. The performance under full communication (black line) and for the guaranteed coordinated approach under zero communication (blue line) are given for comparison. Data were obtained for a total of 1280 tracks and 100 simulations. P-values were obtained from a two-tailed Welch t-test.

Chapter 8

Extension to Sequential Scenarios

8.1 Introduction

So far the focus of the work presented here has been on one-step scenarios, that is settings in which the transition probability is independent of the current state and action. Recall that initially the approach towards local decision-making in Chapter 4 was more general and included sequential settings. Upon closer inspection the problem of approximate local decision-making turned out to be twofold: firstly, there is a combinatorial explosion in attempting to compute the expected future reward under general communication by accounting for all possible trajectories a system might take. Secondly, even for settings without sequentiality, there is an infinite regress problem caused by agents making interdependent decisions simultaneously. The previous chapters have studied several approaches to solving this infinite regress approximately. This chapter will now address the problem of reducing the complexity of sequential scenarios. As with the infinite regress problem the aim here is to develop a scalable, robust algorithm, which can be applied to settings with more than just a very small number of agents, states and actions.

The chapter will begin by returning to the formal treatment of sequential scenarios developed in section 4.6 and exploring how the complexity of these computations can be reduced. This will also require examining to which extent the arguments and

approximations made in the previous chapters continue to hold in this modified setting. The resulting decision-making algorithm will finally be applied to sequential variants of the one-step benchmark scenarios used in previous chapters. This will allow studying its performance similarly to that of the one-step algorithms.

8.2 Approaching Sequential Decision-Making

A sequential scenario is one in which the transition probabilities between states depend on the current state and/or action. They thus require taking the effect of current actions onto the future development of the system into account. Recall from equation (4.4) that the local value function under a local policy given an agents belief can be expressed as

$$V_{\pi_i}(b_i) = \sum_s p(s|b_i) \sum_{\boldsymbol{b}_{-i}} p(\boldsymbol{b}_{-i}|b_i) \sum_{\pi_{-i}} p(\pi_{-i}) \cdot \{R(s, \pi(\boldsymbol{b})) + \gamma \sum_{b_i'} p(b_i'|\boldsymbol{b}, s, \mathcal{I}) V_\pi(b_i')\} \tag{8.1}$$

where $\mathcal{I}$ denotes any prior information which is available about the information exchange between agents (e.g. a stochastic communication pattern). In the one-step treatment in Chapter 6 this was re-formulated in terms of probabilities over the current state and the local actions taken by other agents at the same time. The same approach can be taken here, yielding

$$V(a_i|b_i) = \sum_s \sum_{\boldsymbol{a}_{-i}} p(s, \boldsymbol{a}_{-i}|b_i) \{R(s, \boldsymbol{a}) + \gamma \sum_{b_i'} p(b_i'|b_i, s, \boldsymbol{a}) V^*(b_i)\} \tag{8.2}$$

where $V^*(b_i)$ denotes the value which would be obtained if all agents chose optimally from the following time-step onwards. This optimal value function is exactly where the complexity problem lies, as it is computationally infeasible to account for all possible combinations of future states and communication constellations. Even if some information is available about the nature of the communication pattern between

agents (e.g. if it is known that agents communicate individually with a certain success probability) equation (8.2) remains prohibitively complex. To obtain a feasible local decision-making algorithm it therefore needs to be approximated. In a more general sense it can be written as

$$V(a_i|b_i) = \sum_{s} \sum_{\boldsymbol{a}_{-i}} p(s, \boldsymbol{a}_{-i}|b_i)\{R(s, \boldsymbol{a}) + \gamma V_{future}(b_i, s, \boldsymbol{a})\} \tag{8.3}$$

where $V_{future}(b_i, s, \boldsymbol{a})$ is some future value function which summarises the system dynamics. A solution to Equation (8.2) must then enable an approximate computation of $V_{future}(b_i, s, \boldsymbol{a})$.

8.3 Approximations for Sequential Scenarios

One possible way of approximating $V_{future}(b_i, s, \boldsymbol{a})$ is to make simplifying assumptions about which communication patterns will occur at future time-steps, limiting them to patterns for which the future value can be calculated more easily. Here the centralised solution of the decision problem comes to mind, which is substantially easier to compute than the general communication case: finding the centralised value function is P-complete in the case of collectively observable systems and PSPACE-complete for collectively partially observable systems while the general communication case is NEXP-complete [Pynadath and Tambe, 2002]. Substituting the centralised value function for $V_{future}(b_i, s, a)$ is equivalent to assuming that agents will have full communication from the next time-step onwards. A somewhat similar approach has been applied to solving a (fully decentralized) partially observable stochastic game with good results [Emery-Montemerlo et al., 2004]. The future value is then given by

$$V_{future}(b_i, s, \boldsymbol{a}) = \sum_{s'} p(s'|s, \boldsymbol{a}) V^*(s') \tag{8.4}$$

where $V^*(s')$ is the value under the optimal centralised policy. With this the value at the current time-step is given by

$$V(a_i|b_i) = \sum_s \sum_{\boldsymbol{a}_{-i}} p(s, \boldsymbol{a}_{-i}|b_i)\{R(s, \boldsymbol{a}) + \gamma \sum_{s'} p(s'|s, \boldsymbol{a})V^*(s')\} \tag{8.5}$$

for a time-step with partial or no communication. If agents are able to communicate all observations, the uncertainty over the state and thus also the others' actions collapses and Equation (8.5) reduces to the centralised case

$$V(s, \boldsymbol{a}) = R(s, \boldsymbol{a}) + \gamma \sum_{s'} p(s'|s, \boldsymbol{a})V^*(s') \tag{8.6}$$

This approximation for $V_{future}(b_i, s, \boldsymbol{a})$ is of course overly optimistic. However, using value functions computed for the fully observable case to approximate those for the partially observable setting has previously been shown to be effective in some settings [Littman et al., 1995].

Assuming full communication in future effectively reduces the problem to a one-step setting once more. This not only greatly reduces the computational complexity but also allows referring back to the algorithms developed in Chapter 6. In principle all algorithms studied there can be applied to the solution of Equation (8.5). As the static approximation (see section 6.5) has shown to yield good performance while requiring low computational resources it will be used in this chapter.

Note that while the approximate solution to equation (8.3) described here assumes that full communication will take place at the next time-step this need not be the case in practise. Depending on the communication characteristics of the setting at hand several consecutive time-steps with no or partial communication might occur. Such consecutive time-steps without full communication did not pose a problem in the one-step settings considered so far. As the transition probabilities in those scenarios were independent of the state and action, current states were not affected by previous ones. In the case of sequential scenarios this independence between time-steps is lost since

the current state of the environment depends on the previous state and/or joint action. As a result the local belief held by an agent at a particular time will also depend on the belief it held previously. In particular it will depend on whether or not communication took place at the preceding time-step.

In the one-step settings considered so far an agents' prior belief over the global state of the system was given by the global prior probability of that state. For example, in a setting in which all states were equally likely the prior probability of the system being in a particular state s^t at time t was given by

$$p(s^t) = \frac{1}{|S|} \tag{8.7}$$

and thus independent of the time. This prior was commonly known to all agents and was then updated locally depending on the individual agents' observation as well as the observations communicated by others:

$$p(s^t|\omega_i^t) = \frac{p(\omega_i|s^t)p(s^t)}{p(\omega_i)} \tag{8.8}$$

See Sections 6.2.2 and 6.2.1 for details about how this inference can be carried out efficiently using a graphical model representation of the multi-agent system.

By contrast, the prior belief held by an agent in a sequential setting will directly depend on the previous state and/or joint action through the state transition probability. Consider first the case in which agents were synchronised at the previous time-step. At the current time-step they will all know what the last global state, joint action and received reward were and will therefore hold the same prior belief about the current state, given by

$$p(s^t) = p(s^t|s^{t-1}, \boldsymbol{a}^{t-1}, r^{t-1}) = p(s^t|s^{t-1}, \boldsymbol{a}^{t-1}) \tag{8.9}$$

which is just the probability of transitioning from s^{t-1} into any of the possible states after having taken action $\boldsymbol{a}^{t-1}$. The posterior distribution over the state as seen from

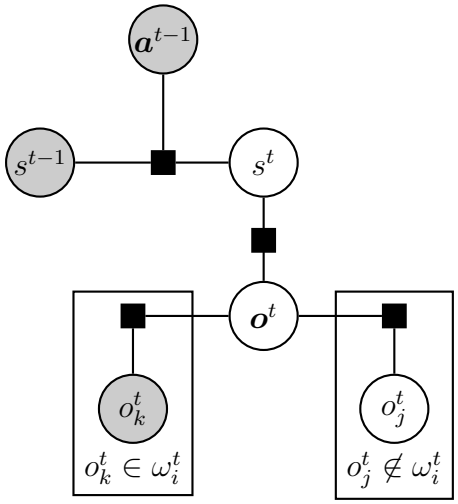


Figure 8.1: Factor graph at first time-step without full communication as seen from agent i

agent i, $p(s^t|\omega_i^t)$, can be obtained from the prior in the same way as in equation (8.8). For a corresponding factor graph visualisation of this case see Figure 8.1.

If there was no full communication at the previous time-step, the agents will hold differing priors about the current state, depending on the information that was available at earlier points in time. Figure 8.2 shows the factor graph representation of this scenario as seen locally by agent i for the case of two consecutive time-steps without full communication. From this it becomes apparent that the message passed from the factor node connecting s^t to s^{t+1} depends on the variables observed by agent i at time t and with that on the belief over states held at that time . In particular, the message sent from s^t to the factor node connecting s^t and s^{t+1} can be written as

$$\mu_s = p(s^t|\omega_i^t)\ \mu_r\ \mu_{\boldsymbol{a}} \tag{8.10}$$

where $p(s^t|\omega_i^t)$ is the posterior distribution over s^t calculated at the time t and μ_r and $\mu_{\boldsymbol{a}}$ are the messages obtained from the factor nodes connecting to r^t and $\boldsymbol{a}^t$ respectively. Calculating the message coming from the direction of r^t is simple, as the binary probability distribution over rewards, $p(r^t|s^t, \boldsymbol{a}^t)$, can be extracted from the

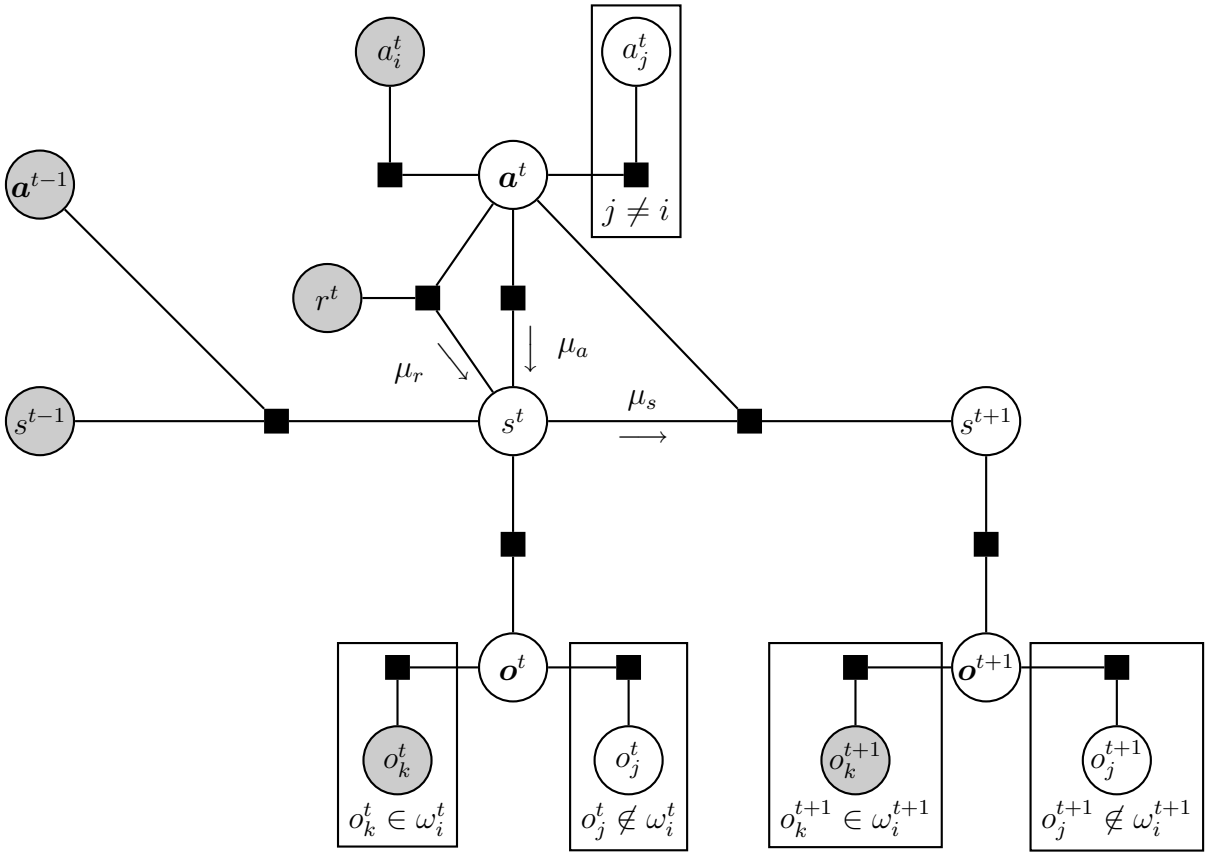


Figure 8.2: Factor graph representation of an agent system with one and two consecutive time-steps of partial or no communication. The graph is constructed from the viewpoint of agent i

reward matrix. Obtaining the message coming from the factor connecting to $\boldsymbol{a}^t$ is less straightforward. Computing this message requires knowledge of the probability distribution which describes the choice of joint action at time t. Had the agents been synchronised at that time-step the joint action would have been given by the centralised policy. However, this is not the case here as the agents did not synchronise and chose actions locally instead. It is therefore not immediately clear what probability distribution describes the generation of the joint action given the global state. In particular, the previous joint action is a result of individual local action decisions made by agents using the very approximate algorithm being constructed. These local action choices were based on the individual locally available information $\boldsymbol{\omega}$. Thus, the

graph structure shown here, which implies that $\boldsymbol{a}^t$ was directly generated by s^t does not actually represent the true mechanism by which $\boldsymbol{a}^t$ was chosen.

On the other hand, it is not straightforward to write down an alternative graphical model which accurately reflects how the local actions were chosen at time t. This would require knowing which observations the other agents made and visualising the interdependence of their local action choices. At the same time, a functional expression for the probability over joint actions at the previous time-step is needed to calculate the agent's current belief over the state s^{t+1}. In order to still be able to use a graphical model representation and calculate a posterior distribution over s^{t+1} from it, an approximate expression for the probability over joint actions $\boldsymbol{a}^t$ given the state s^t will be used. Similar to the choice of distribution for the static approximation in section 6.5, a uniform distribution over joint actions was chosen here. Thus, all actions which are compatible with the known local action that agent i took at the previous time-step will have the same probability of having occurred.

Note that the graph in Figure 8.2 is not actually a tree. In order to apply the sum-product algorithm to this inference problem it therefore needs to be turned into one. This can be done by turning the initial directed graph into a junction tree (see Section A.3). In this particular case this is just a matter of drawing the state and joint action together into one augmented variable represented by a single node. The resulting factor graph is shown in figure 8.3. On this graph the sum-product algorithm can be used to obtain the posterior probability $p(s^{t+1}|\omega_i^t, \omega_i^{t+1}, r^t, \boldsymbol{a}^{t-1}, s^{t-1})$. In the general case the posterior at time t would be denoted by $p(s^0|\bar{\omega}_i^t, \bar{r}^t, \boldsymbol{a}^0, s^0)$ where $t = 0$ marks the time at which the agents were last synchronised and $\bar{\omega}_i^t$ and $\bar{r}^t$ are the observation and reward histories from the last synchronisation up to the current time-step respectively.

Once the expression for the posterior over the state has been obtained, agents can calculate the approximated expected value according to equation (8.5), using the static

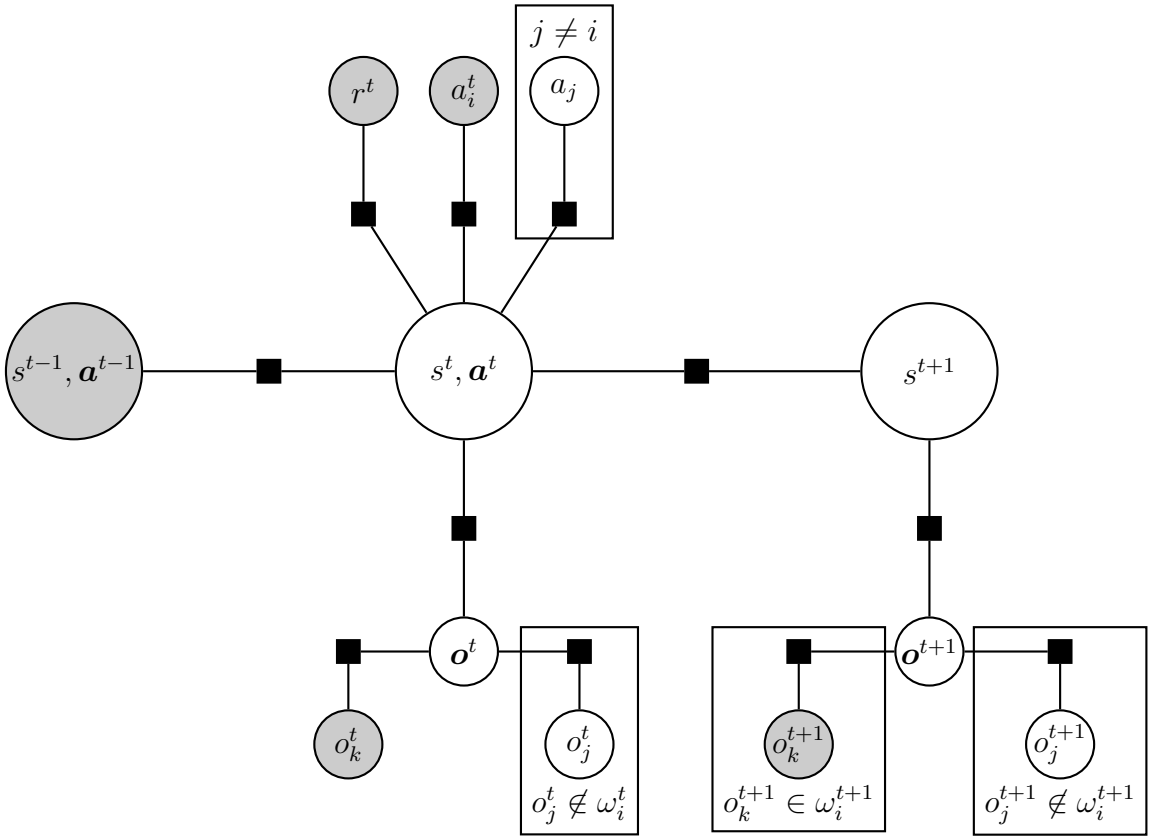


Figure 8.3: Tree-structured factor graph including the augmented state-action node

algorithm for approximating others' action choices.

Constructing a guaranteed coordinated decision-making algorithm which uses the same approximation to the systems' sequentiality can be done in a similar way. Here the computation of the posterior distribution over states is less complicated as the previous choice of joint action is always known and local observations are not taken into account. See Figure 8.4 for the corresponding graphical model.

8.4 Results

The decision-making algorithm described above was applied to sequential versions of the benchmark algorithms introduced in chapter 5. To recap, the characteristics of these scenarios have been summarised in table 8.1. For detailed descriptions of the

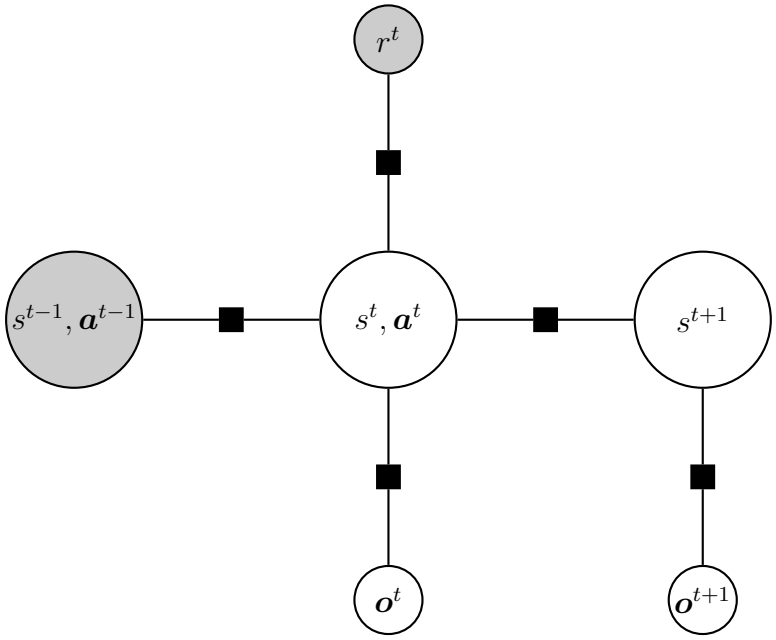


Figure 8.4: Factor graph for the guaranteed coordinated algorithm for the case of two consecutive time-steps without synchronisation

individual settings and the respective reward matrices used see Section 5.2. Turning these one-step settings into sequential ones requires changing the transition function such that transitions are no longer independent of the current state and joint action. There are many ways in which the transition of states might depend on the current state and action. For the simulations carried out here the functions were chosen such that a system in a given state was more likely to transition into the same state at the next time-step than into others. In principle any other sequential transition function could have been implemented instead. This particular choice was made because it yields a genuinely sequential scenario but is nonetheless based on a transition function with a simple structure.

	Tiger Problem	Meeting Problem	Monitoring Problem
# Agents	2	2	4
States	Defined by content behind doors	Defined by location of agents and predator	Defined by type of passing object
Local observations	Content behind one door for each agent	Own location, uncertain observation of other agent's location	Observation of one characteristic of the passing object per agent
Local actions	Choose one door or pass	Move to another location or stay at current	Attack, standby or send friendly signal
Coordination aim	Choose the same door	Meet at one location, avoid predator	Act coordinatedly
Jointly observable	Yes	Yes	No
Symmetric optimal action	Yes	No	No
Communication pattern	Synchronised	Synchronised	General

Table 8.1: Summary of the characteristics of the one-step benchmark problems introduced in Section 5.2.

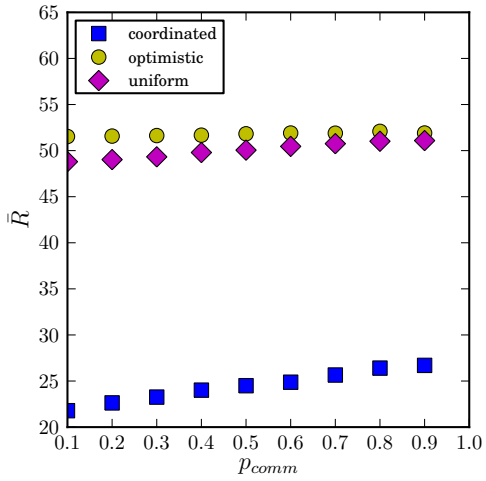
8.4.1 Tiger Problem

For the tiger problem the transition matrix was designed such that the system had a proportionately higher probability of transitioning to the same state as the current when a "good" joint action was chosen. In this case a good action was defined as one which yields a high reward. The probability of transitioning into any of the other states was kept uniform. Agents were therefore doubly rewarded for choosing the optimal joint action: not only does it yield the highest reward but also decreases the uncertainty over the subsequent state. The reward settings were unchanged from those of the one-step scenario.

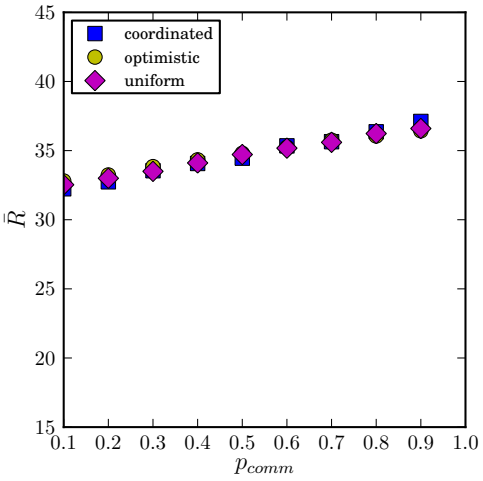
Figure 8.5 shows the average reward obtained per time-step without communication in this setting. Similar to the one-step case, the overall performance of the individual algorithms in this scenario depends on the particular settings of the reward matrix. However, in all cases the optimistic algorithm performs as well or better than the uniform one. Note that due to the sequentiality of the scenario the reward obtained under the approximate algorithms is no longer independent of the communication probability. This is because agents' action choices rely on their inference over the global state which becomes less informed with an increased number of time-steps without communication. This effect is directly visible in Figure 8.6 which shows the average reward obtained depending on the number of time-steps since the last communication. For most settings the performance of an algorithm decreases for increased time without communication.

8.4.2 Meeting Problem

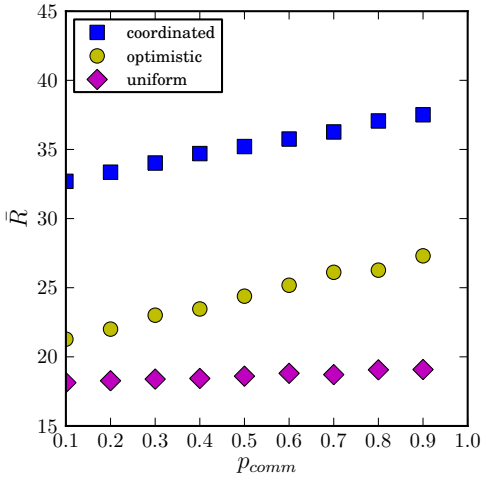
The transition matrix for the meeting scenario was chosen similar to that used for the tiger problem with better action choices leading to higher probabilities of transitioning into the same state.



(a) Setting 1, p-values $0.0 - 6.9 \times 10^{-6}$



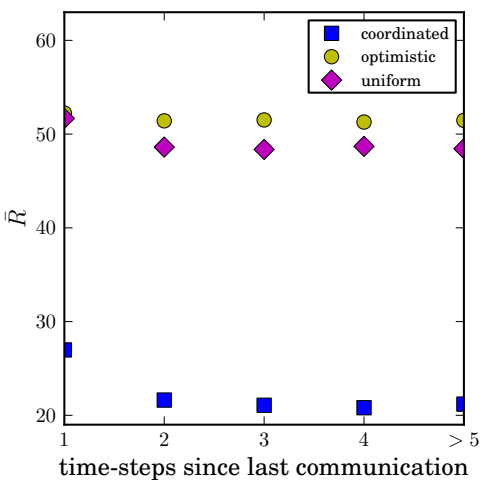
(b) Setting 2, p-values $2.7 \times 10^{-5} - 0.57$



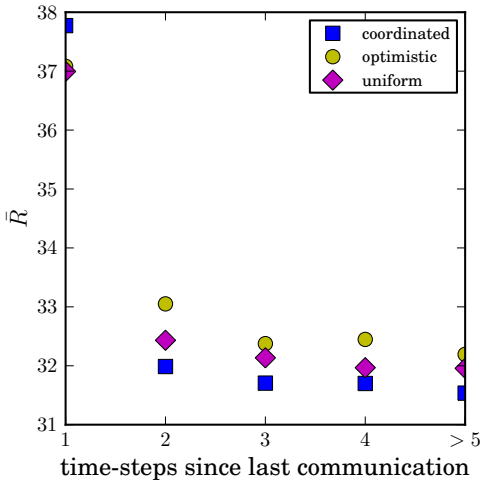
(c) Setting 3, p-values 0.0

Figure 8.5: Average reward per time-step without communication for the coordinated, uniform and optimistic algorithms in the three settings of the tiger problem. Data was averaged over 2000 simulations with 500 time-steps each. P-values were obtained in a two-tailed Welch t-test between the optimistic and uniform performance.

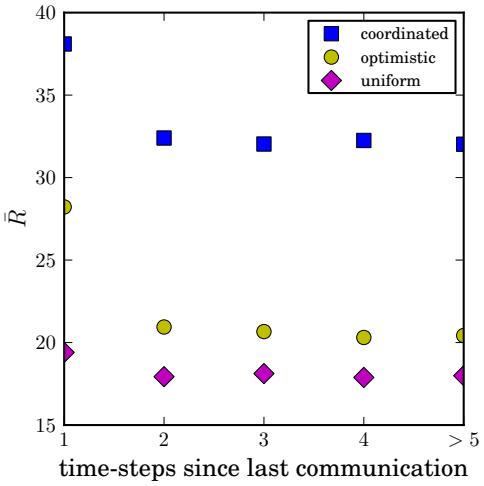
Figure 8.7 shows the average reward per time-step without communication. Here the performance of both static algorithms is unchanged with increasing communication probabilities while the performance of the guaranteed coordinated approach improves with increased communication. This is confirmed by Figure 8.8 which shows the average reward depending on the number of time-steps since the last communication.



(a) Setting 1



(b) Setting 2



(c) Setting 3

Figure 8.6: Average reward per time-step without communication as a function of time since the last synchronisation in the tiger problem. Data was averaged over 2000 simulations with 500 time-steps each.

Similar to the one-step case the optimistic approximation performs significantly better than the uniform one as well as the coordinated algorithm in this scenario.

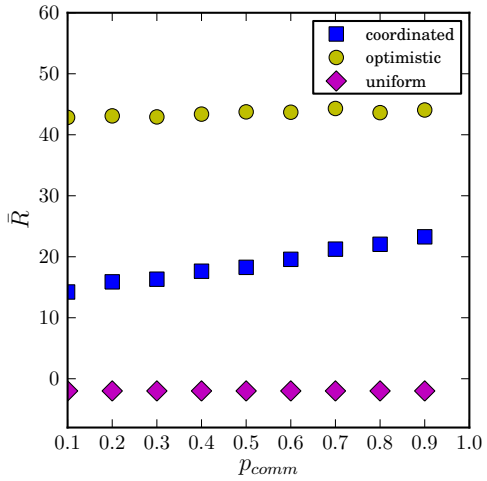


Figure 8.7: Average reward per time-step without communication for the coordinated, uniform and optimistic algorithms in the meeting problem. Data was averaged over 500 simulations with 500 time-steps each. All comparisons of means in a two-tailed Welch t-test yielded a p-value of 0.0.

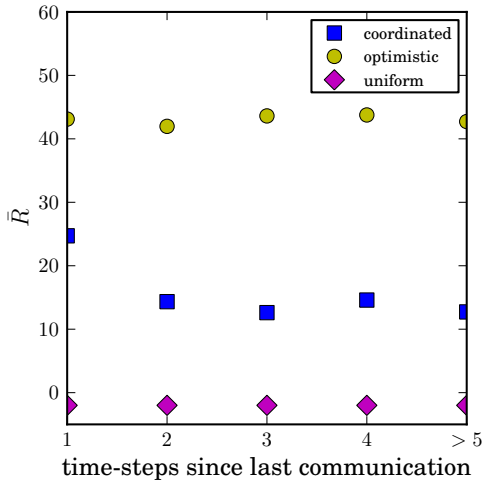
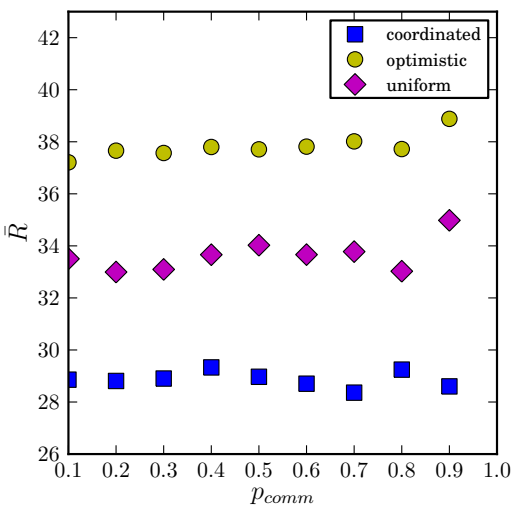


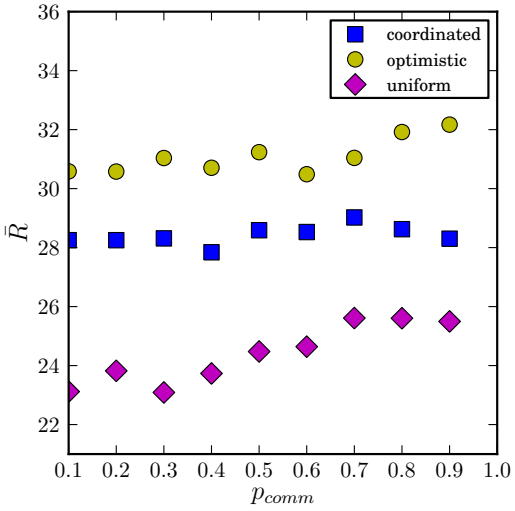
Figure 8.8: Average reward per time-step without communication as a function of time since last communication in the meeting problem. Data was averaged over 500 simulations with 500 time-steps each.

8.4.3 Monitoring Problem

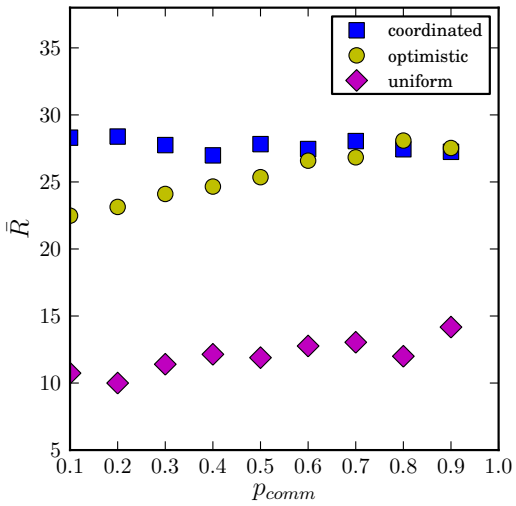
The monitoring scenario was motivated by a group surveillance task in which agents choose actions depending on the identities of passing objects. In this setting the actions



(a) Setting 1

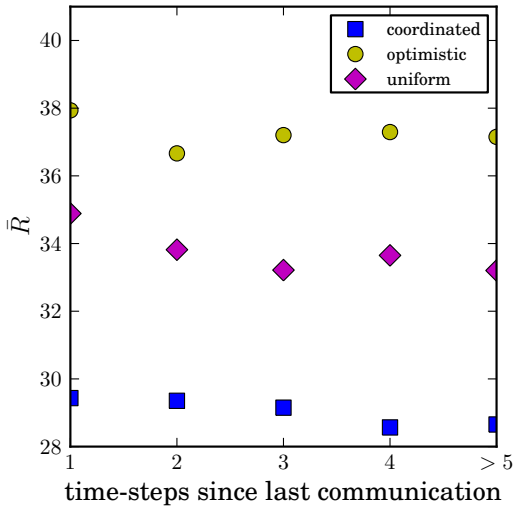


(b) Setting 2

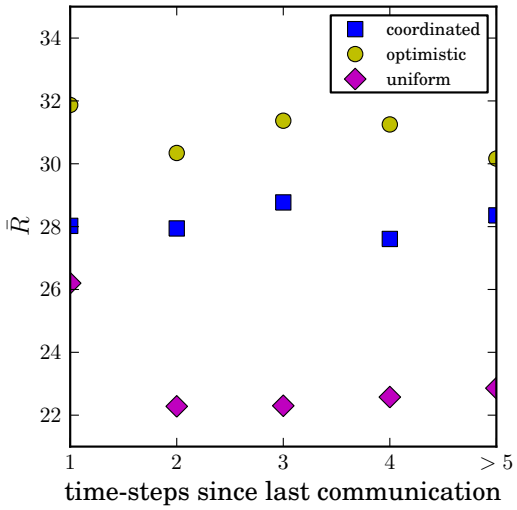


(c) Setting 3

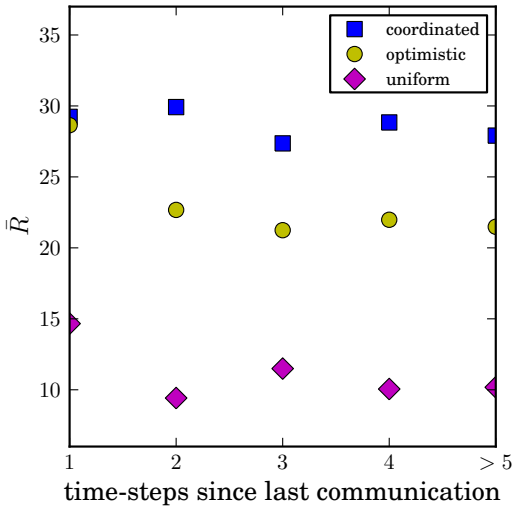
Figure 8.9: Average reward per time-step with zero communication for the coordinated, uniform and optimistic algorithms in the three settings of the monitoring problem with synchronised communication. Data was averaged over 2000 simulations with 50 time-steps each. Algorithms were compared in a two-tailed Welch t-test. All p-values for the comparison between the uniform and optimistic algorithms were below 10^{-5}. The maximum p-value for the comparison between the uniform and coordinated algorithms was 8.44×10^{-3} (Setting 2). The maximum p-value for the comparison between the optimistic and coordinated algorithms was 0.79 (Setting 3).



(a) Setting 1



(b) Setting 2



(c) Setting 3

Figure 8.10: Average reward per time-step without communication as a function of time since the last synchronisation in the monitoring problem. Data was averaged over 2000 simulations with 50 time-steps each.

chosen should not influence the transition of states (i.e. the occurrence of objects). The transition matrix was therefore chosen to be independent of the joint action taken but transitions into the same state as the current one were again given higher probabilities. This can be thought of as a setting where objects do not necessarily occur individually

and might pass in groups.

Centralised Communication

Due to the monitoring scenario comprising more than two agents, different communication patterns are possible in this setting. The algorithm was first applied to a scenario with synchronised communication in which agents communicate with all others whenever communication is possible. Figure 8.9 shows the reward obtained per time-step without communication for this setting. For the first two reward settings the performance differs between algorithms while the dependency of each algorithm's performance on the communication probability appears to be weak. In the third setting, for which the coordinated approach initially performs best, the optimistic approach reaches a similar performance for high communication probabilities.

Figure 8.10 shows the average reward obtained as a function of time since the last communication. In the third setting the difference between the first and second time-step since communication is comparatively large for the optimistic algorithm. This helps to explain the performance of the optimistic algorithm observed in Figure 8.9c.

Individual Communication

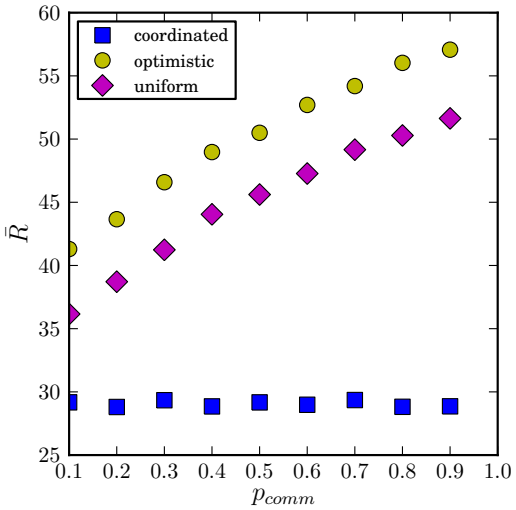
The algorithm was also applied to a setting with independent communication links between individual agents. Figure 8.11 shows the average reward obtained per time-step in this setting. Here, the static algorithms outperform the coordinated one for most reward settings and communication probabilities. Note in particular the third reward setting in which the uniform algorithm outperforms the coordinated algorithm for communication probabilities above $p_c \approx 0.3$ even though the coordinated algorithm clearly performs better for the zero-communication case. Again, the optimistic algorithm performs significantly better than the uniform one. As with the performance in the one-step monitoring problem observed for this setting (see Figure 6.16), the

benefit of communication quickly outweighs the certainty obtained from guaranteed coordination.

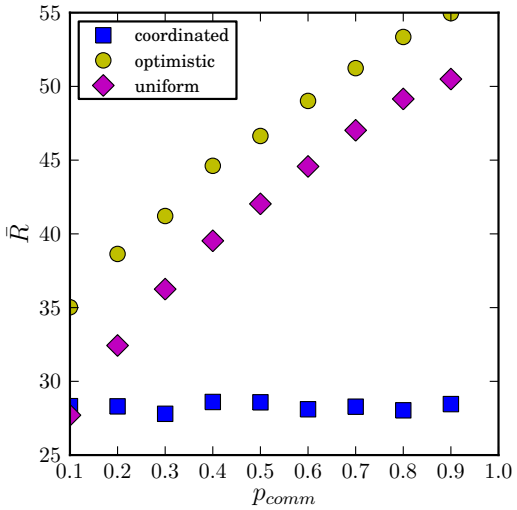
In the case of individual communication links there might be no time-steps at which agents are fully synchronised (depending on the communication probability). Plots of the kind such as in figure 8.10 are therefore not meaningful for this setting.

8.5 Discussion

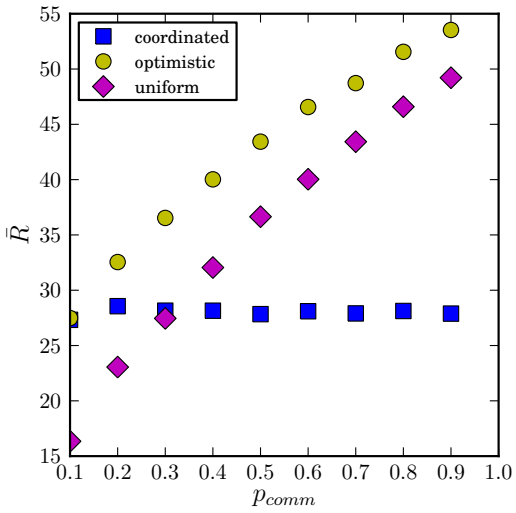
The results presented above have shown that the sequential local decision-making algorithm proposed in this chapter yields good performance on the benchmark problems. Although the assumption of full communication at future time-steps is very simplistic, it provides an effective approximation of the sequential decision-problem for these scenarios. The simulations also show that in settings with sequentiality, the approximate algorithms can perform even better when some communication is available, as this improves the quality of the inference carried out by agents. What is more, the approach provides an algorithm which is computationally very lightweight and will therefore scale to much larger systems: the only computations needed at each time-step are the inference over states and the marginalisation over states and other agents' actions.



(a) Setting 1



(b) Setting 2



(c) Setting 3

Figure 8.11: Average reward per time-step with partial communication for the coordinated, uniform and optimistic algorithms in the three settings of the monitoring problem with individual communication links. Data was averaged over 2000 simulations with 50 time-steps each. Algorithms were compared in a two-tailed Welch t-test. All p-values for the comparison between the optimistic and uniform algorithms were below 10^{-5}.

Chapter 9

Conclusions and Future Work

This chapter will conclude this thesis by reviewing the work presented here and evaluating its findings. To recap, it will provide a summary of the analysis carried out in the preceding chapters and discuss its merits and weaknesses. Finally it will suggest possible areas of future work that could improve and expand the work carried out here.

9.1 Summary

This thesis has studied the problem of decision-making in agent systems with varying amounts of information exchange between agents. Analysing such settings was motivated by a need for algorithms which allow scalable and flexible decision-making while at the same time ensuring good overall performance in scenarios with sparse communication. Within this work, a localised approach was taken to decision-making: individual actions were chosen at agent level based on locally available information rather than by a centralised algorithm. A centralised solution was relied on only in those cases where full synchronisation was available (see Chapter 4).

In addition, this work was motivated by the hypothesis that the existing focus on guaranteed coordination of agent actions is not always optimal. The conjecture was that it might lead to worse performance than an algorithm which makes use of all available information, even if it is not shared between all agents (Section 4.4).

In analysing these settings it became apparent that there are two main challenges to finding suitable solutions: that of approximating the infinite time-horizon of the decision-problem and that of approximating the infinite regress over agent reasoning. To simplify the treatment, the infinite regress problem was first studied independently within the setting of one-step scenarios (Chapters 5 to 7) in which there is no sequentiality of global states. Chapter 6 presented three approximate algorithms for local decision-making in such settings: one based on an iterative solution to the infinite regress, one static heuristic and one learning algorithm. These algorithms were all tested on benchmark tasks and compared to the performance of a completely decentralized and a guaranteed coordinated algorithm. Chapter 7 then applied the heuristic algorithm to a real-world data set and showed how it can be employed to solve the coordination problem between a set of recording agents. Finally, Chapter 8 extended the previous work to sequential settings in which there is an interdependence between the sequence of global states and joint actions. This was achieved by simplifying the settings in a way that allowed the application of the one-step algorithms once more. The resulting algorithm was tested on sequential versions of the benchmark settings used previously.

9.2 Conclusions

When judged by the objectives that motivated this work, the algorithms developed here have a number of strengths.

From an information-theoretic viewpoint the work presented has confirmed the hypothesis formulated in Section 4.4: guaranteeing agent coordination at the cost of local information will lead to lesser overall performance in many settings. Designing algorithms which guarantee coordination without explicitly considering the trade-off between coordination and local information is therefore not necessarily the best ap-

proach to solving decision-making in the scenarios considered here. This finding has implications for many current decision-making algorithms which are firmly based on guaranteed coordination. It is possible that a number of these could improve their performance if this prerequisite were dropped.

From a more practical viewpoint, the algorithms presented here fulfil many of the requirements outlined at the beginning. They provide a general method for decision-making in multi-agent systems with sparse, irregular and/or non-deterministic communication patterns. In particular, they can be employed in settings with faulty communication to increase robustness against communication failures. In addition, the heuristic algorithm allows the implementation of local decision-making at very low computational cost, enabling its application to larger, more complex systems than previously possible. The performance of this algorithm observed in some of the benchmark settings also suggests that it can be used to compute approximate solutions to a completely decentralized policy for dec-(PO)MDPs which would otherwise be too complex. In the tiger problem, for example, the heuristic algorithms performed just as well as the decentralized one (see Section 6.6.1).

On the other hand, the work presented here also has some shortcomings. This study has taken a first look at how local decision-making can be implemented for general communication settings. As such the results and methods are still somewhat basic.

No performance bounds were given for any oft the algorithms developed. While the performance on the benchmark tasks provides good evidence for their applicability, more reliable statements will ultimately be needed. Similarly, no quantitative measure has been found to determine which characteristics of a system make it better suited towards the guaranteed coordinated or the local approach. To determine which of the two approaches should be favoured in a particular setting, such an indicator is

necessary.

The approximations used to approach the two main challenges encountered in this work (the infinite regress and sequentiality) are both rudimentary. Out of the algorithms developed for one-step scenarios, only the static one is universally applicable thanks to its low computational complexity. While its performance is surprisingly good, it is based on quite a simple heuristic. Similarly, the approximation used for the sequential setting is basic and does not reflect the complexity of the decision-problem to a great degree. Although performing well, neither of these approaches are particularly principled.

9.3 Future Work

Most of the immediate suggestions for future work on this topic can easily be derived from the shortcomings outlined above.

More quantifiable work is needed on the influencing factors that determine the trade-off between guaranteed coordination and using local information. In particular, it would be desirable to develop some form of characteristic measure which condenses the system attributes in a way that immediately shows which of the two approaches is more effective. Any such measure would have to reflect, among other things, the reward structure of the setting and the interdependence of agents' action choices.

Similarly, more reliable statements are needed about the performance of the individual decision-making algorithms. Here, the static heuristic is particularly interesting, as it has proven to be the most applicable in the benchmark settings tested within this work. To ascertain whether such performance guarantees can be given, it might provide useful to study existing approaches to dec-(PO)MDPs, some of which do provide performance guarantees at much higher computational complexity, in more detail.

On an algorithmic level, the approaches for solving both the infinite reasoning

regress and the complexity of sequential settings should be refined. For the former, the aim should be to find a distribution over other agents' actions which reflects their actual choices more accurately. In particular when partial communication is available, the uniform distribution over others' actions is quite cautious. Here, the amount of information exchange should influence the distribution used.

Similarly, for the challenge of approximating the system's sequentiality a more sophisticated representation of the future value is needed. This should reflect the particular communication pattern to some extent. One possibility might be to begin with a crude approximation, such as full future communication, and adjust this estimate as more data about actual rewards becomes available. As an extension to this approach a more Bayesian stance could be taken by keeping a belief over future values, which would be updated successively.

Finally, the algorithms developed should be applied to more real-world tasks. In particular for the case of sequential scenarios this will provide the ultimate test for how functional they are both in terms of performance and computational overhead.

Appendix A

Inference in Graphical Models

A.1 Introduction

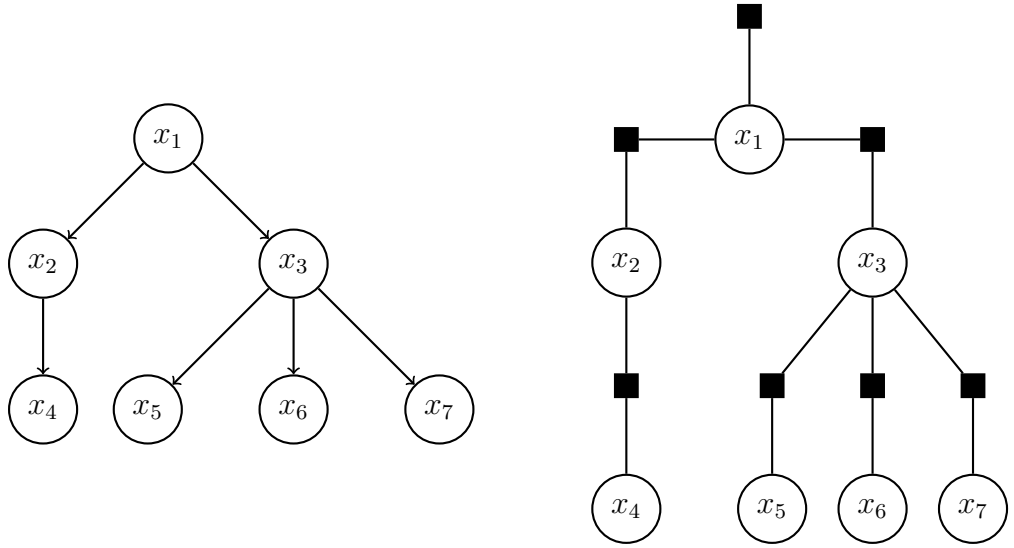
This appendix will outline an efficient inference algorithm on tree-structured graphs. It enables the calculation of the marginal distribution over one or more variables in the graph.

A.2 Tree-Structured Graphs

The following algorithm, the *sum-product algorithm* [Pearl, 1988,Frey, 1998], is applicable to directed and undirected graphs which have a tree structure. In the undirected case a tree is defined as a graph in which there is one and only one path between any two nodes. A directed graph is a tree if it has one node which has no parents and all other nodes have one parent. In a tree-structured graph the node which has no parents is called the *root* while those nodes at the "end" of the tree which are childless are referred to as *leaf nodes*. Note that in the case of an undirected tree-structured graph any of the nodes can be chosen as the root.

A.2.1 Factor Graphs

The sum product algorithm can be formulated in a convenient way in the context of *factor graphs*. These provide an alternative representation to the (un-)directed graphs



(a) Tree-structured graph (b) Possible factor graph representation

Figure A.1: Tree structured directed graphical model and possible factor graph representation of it.

described so far, which emphasises the factorial nature of the joint distribution over variables.

Recall from section 2.3.1 that the joint distribution over all variables in a graph can be written as a product over individual functions which are defined on subsets X_g of the set of variables:

$$p(x_1, \ldots, x_n) = \prod_g f_g(X_g) \tag{A.1}$$

In a factor graph this is made explicit by adding nodes to the graph for each of the factors in the joint distribution. Thus, a factor graph consists of variable nodes, factor nodes and undirected edges that connect the two types of nodes. Each directed (or undirected) graph can be converted to a factor graph and in general there will be multiple factor graphs which represent the same directed graph. For an example of a tree structured graph and a corresponding factor graph representation see figure A.1.

A.2.2 Belief Propagation

Belief propagation (also referred to as the sum-product algorithm) is based on the idea of passing individual messages between two nodes in a graph to perform inference calculations. The formulation below uses discrete variables for which marginalisation correspond to sums. The algorithm is more general, however, and can also be applied to continuous variables.

In a tree-structured graphical model which has been written as a factor graph the marginal distribution over a variable node x_i can be written as

$$p(x_i) = \prod_{k \in ne(x_i)} \mu_{f_k \to x_i}(x_i) \tag{A.2}$$

where $ne(x_i)$ denotes the neighbours of node x_i and $\mu_{f_k \to x_i}(x_i)$ is a message passed from the neigh boring factor node f_k to node x_i. This message takes the form

$$\mu_{f_k \to x_i}(x_i) = \mathcal{F}_k(x_i, X_k) \tag{A.3}$$

where X_k denotes the set of variables that lie in the sub-tree below f_k, see figure A.2. Let $X_{f_k \setminus i}$ denote the subset of variables other than x_i on which f_k depends. By recursively using the graph's factorisation property, the message from f_k to x_i can be expressed as

$$\mu_{f_k \to x_i}(x_i) = \sum_{x_m \in X_{f_k \setminus i}} f_k(x_i, X_{f_k \setminus i}) \prod_{x_l \in X_{f_k \setminus i}} \mu_{x_l \to f_k}(x_l) \tag{A.4}$$

where $\mu_{x_l \to f_k}(x_l)$ denotes the messages f_k receives from its neigh boring variable nodes in $X_{f_k \setminus i}$. Using graph factorisation once more, the message sent from x_l to f_k is found to be

$$\mu_{x_l \to f_k}(x_l) = \prod_{m \in ne(x_l) \setminus f_k} \mu_{f_m \to x_l} \tag{A.5}$$

Equations A.4 and A.5 provide recursive means for calculating the messages sent to node x_i through the graph. To compute the marginal distribution over x_i it is chosen

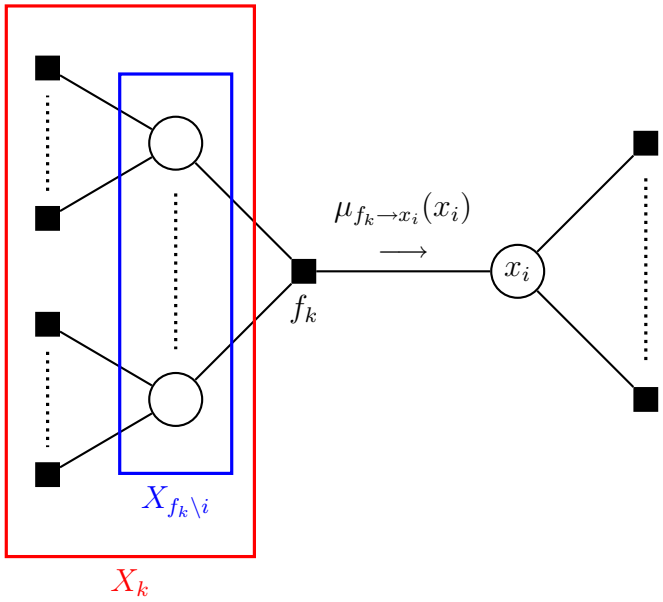


Figure A.2: Part of a factor graph with sub-graph relevant to message passed from factor node f_k to variable node x_i

as the root nodes with all resulting leaf nodes initialised as

$$\mu_{x_a \to f_b}(x_a) = 1 \tag{A.6}$$

if they are variable nodes and

$$\mu_{f_c \to x_d}(x_d) = f(x_d) \tag{A.7}$$

if they are factor nodes. Messages are then propagated through to x_i from the leaves according to equations (A.4) and (A.5) respectively. For a particularly efficient use of this algorithm, messages are first sent through from all leaves to an arbitrary root node. From there the messages associated with the opposite direction are sent back into the direction of the leaves. At this point all nodes have exchanged messages with all their neighbours, which allows any marginal to be calculated immediately.

If some of the variables in the graph have been observed, the message passing scheme remains the same. The calculation of individual messages is merely adapted such that summations over observed variables in messages of the type given by equation

A.4 are collapsed to a single term. This will result in non-normalised message being passed around the graph, which can easily be corrected for by normalising the final result in equation A.2.

A.3 General Graphs

The sum-product algorithm is only applicable to trees and polytrees (graphs in which there are nodes that have more than one parent but in which there is still only one path between any two nodes). To perform inference on arbitrary acyclic graphs, a more general message-passing algorithm is needed. This can be achieved by using graph manipulations to turn the graph into a *junction tree* [Lauritzen and Spiegelhalter, 1988], in which groups of nodes which do not fulfil the tree structure requirement are pulled together into clique nodes such that the resulting graph is again a tree. Belief propagation can then be used to perform inference on the clique nodes.

Appendix B

Rewards used in Monitoring Problem

The following pages list the reward settings used in the monitoring scenario (see section 5.2.3).

Actions	Rewards		
	Object: Colleague	Object: Enemy	Object: Civilian
All standby	0	0	-70
1 standby, 3 send friendly	100	0	-100
1 standby, 3 attack	-100	-80	100
1 standby, 2 send friendly, 3 attack	-20	-20	30
1 standby, 1 sends friendly, 2 attack	-50	-50	50
2 standby, 2 friendly	90	0	-100
2 standby, 2 attack	-80	-80	60
2 standby, 1 sends friendly, 1 attacks	-30	-30	20
3 standby, 1 shows friendly	70	0	-100
3 standby, 1 attacks	-40	-30	30
All send friendly	90	0	-100
1 sends friendly, 3 attack	-100	-100	90
2 send friendly, 2 attack	-50	-50	50
3 send friendly, 1 attacks	-30	-30	30
All attack	-100	-100	90

Table B.1: Reward settings used in the first setting of the monitoring problem

Actions	Rewards		
	Object: Colleague	Object: Enemy	Object: Civilian
All standby	0	0	-70
1 standby, 3 send friendly	100	0	-100
1 standby, 3 attack	-100	-90	100
1 standby, 2 send friendly, 3 attack	-100	-70	30
1 standby, 1 sends friendly, 2 attack	-100	-80	50
2 standby, 2 friendly	70	0	-100
2 standby, 2 attack	-100	-100	50
2 standby, 1 sends friendly, 1 attacks	-80	-80	20
3 standby, 1 shows friendly	70	0	-100
3 standby, 1 attacks	-100	-100	30
All send friendly	90	0	-100
1 sends friendly, 3 attack	-100	-100	90
2 send friendly, 2 attack	-80	-80	50
3 send friendly, 1 attacks	-60	-60	30
All attack	-100	-100	90

Table B.2: Reward settings used in the second setting of the monitoring problem

Actions	Rewards		
	Object: Colleague	Object: Enemy	Object: Civilian
All standby	-100	0	-100
1 standby, 3 send friendly	100	0	-100
1 standby, 3 attack	-100	-100	100
1 standby, 2 send friendly, 3 attack	-100	-100	-30
1 standby, 1 sends friendly, 2 attack	-100	-100	50
2 standby, 2 friendly	70	0	-100
2 standby, 2 attack	-100	-100	50
2 standby, 1 sends friendly, 1 attacks	-100	-100	-30
3 standby, 1 shows friendly	50	0	-100
3 standby, 1 attacks	-100	-100	-30
All send friendly	90	0	-100
1 sends friendly, 3 attack	-100	-100	90
2 send friendly, 2 attack	-100	-100	50
3 send friendly, 1 attacks	-100	-100	30
All attack	-100	-100	90

Table B.3: Reward settings used in the third setting of the monitoring problem

Bibliography

[Amato et al., 2006] Amato, C., Bernstein, D. S., and Zilberstein, S. (2006). Optimal fixed-size controllers for decentralized POMDPs. In *Proceedings of the Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM) at AAMAS*.

[Amato et al., 2007] Amato, C., Carlin, A., and Zilberstein, S. (2007). Bounded dynamic programming for decentralized POMDPs. In *AAMAS 2007 Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains*.

[Aumann, 1974] Aumann, R. (1974). Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96.

[Becker et al., 2005] Becker, R., Lesser, V., and Zilberstein, S. (2005). Analyzing myopic approaches for multi-agent communication. In *Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, pages 550–557.

[Bellman, 1957a] Bellman, R. (1957a). A Markovian Decision Process. *Journal of Mathematics and Mechanics 6*.

[Bellman, 1957b] Bellman, R. (1957b). *Dynamic Programming*. Princeton University Press.

[Bernstein, 2005] Bernstein, D. S. (2005). Bounded Policy Iteration for Decentralized POMDPs. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1287–1292.

[Bernstein et al., 2002] Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. (2002). The Complexity of Decentralized Control of Markov Decision Processes. *Math. Oper. Res.*, 27(4):819–840.

[Bishop, 2006] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer New York.

[Blackwell and Dubins, 1962] Blackwell, D. and Dubins, L. (1962). Merging of opinions with increasing information. *Annals of Mathematical Statistics*, 33(3):882–886.

[Boularias and Chaib-draa, 2008] Boularias, A. and Chaib-draa, B. (2008). Exact dynamic programming for decentralized POMDPs with lossless policy compression. In *Proc. of the Eighteenth Int. Conf. on Automated Planning and Scheduling*.

[Boutilier, 1999] Boutilier, C. (1999). Sequential Optimality and Coordination in Multiagent Systems. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 478–485, San Francisco, CA, USA. Morgan Kaufmann.

[Cammarata et al., 1988] Cammarata, S., McArthur, D., and Steeb, R. (1988). Strategies of cooperation in distributed problem solving. In *Distributed Artificial Intelligence*, pages 102–105, San Francisco, CA, USA. Morgan Kaufmann.

[Cassandra et al., 1997] Cassandra, A., Littman, M., Zhang, N., et al. (1997). Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pages 54–61.

[Chechetka and Sycara, 2007] Chechetka, A. and Sycara, K. (2007). Subjective approximate solutions for decentralized POMDPs. In *AAMAS '07: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1–3, New York, NY, USA. ACM.

[Durfee, 1988] Durfee, E. H. (1988). *Coordination of Distributed Problem Solvers*. Kluwer Academic, Norwell, MA, USA.

[Emery-Montemerlo et al., 2004] Emery-Montemerlo, R., Gordon, G., Schneider, J., and Thrun, S. (2004). Approximate Solutions for Partially Observable Stochastic Games with Common Payoffs. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 136–143, Washington, DC, USA. IEEE Computer Society.

[Frey, 1998] Frey, B. (1998). *Graphical Models for Machine Learning and Digital Communication*. MIT press.

[Fudenberg and Tirole, 1991] Fudenberg, D. and Tirole, J. (1991). *Game Theory*. MIT Press.

[Genesereth et al., 1986] Genesereth, M., Ginsberg, M., and Rosenschein, J. (1986). Cooperation without communication. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 51–57.

[Goldman and Zilberstein, 2008] Goldman, C. V. and Zilberstein, S. (2008). Communication-Based Decomposition Mechanisms for Decentralized MDPs. *Artificial Intelligence Research*, 32:169–202.

[Hansen, 1998a] Hansen, E. (1998a). An improved policy iteration algorithm for partially observable MDPs. *Advances in Neural Information Processing Systems*, pages 1015–1021.

[Hansen, 1998b] Hansen, E. (1998b). Solving POMDPs by searching in policy space. In *Proceedings of the fourteenth conference on uncertainty in artificial intelligence*, pages 211–219.

[Hansen, Eric A. and Bernstein, Daniel S. and Zilberstein, Shlomo, 2004] Hansen, Eric A. and Bernstein, Daniel S. and Zilberstein, Shlomo (2004). Dynamic programming for partially observable stochastic games. In *AAAI'04: Proceedings of the 19th National Conference on Artifical Intelligence*, pages 709–715. AAAI Press / The MIT Press.

[Howard, 1960] Howard, R. (1960). *Dynamic programming and Markov Processes*. MIT press.

[Huber and Durfee, 1993] Huber, M. and Durfee, E. (1993). Observational Uncertainty in Plan Recognition Among Interacting Robots. In *Proceedings of the IJCAI Workshop on Dynamically Interacting Robots*, pages 68–75, Chambéry, France.

[Huber and Durfee, 1995] Huber, M. J. and Durfee, E. H. (1995). Deciding when to commit to action during observation-based coordination. In Lesser, V., editor, *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95)*, pages 163–170, San Francisco, CA, USA. The MIT Press: Cambridge, MA, USA.

[Jaynes, 2003] Jaynes, E. (2003). *Probability Theory: The Logic of Science*. Cambridge University Press.

[Jennings, 1996] Jennings, N. R. (1996). Coordination techniques for distributed artificial intelligence. *Foundations of distributed artificial intelligence*, pages 187–210.

[Jennings et al., 1998] Jennings, N. R., Sycara, K., and Wooldridge, M. (1998). A Roadmap of Agent Research and Development. *Autonomous Agents and Multi-Agent Systems*, 1(1):7–38.

[Kaelbling et al., 1996] Kaelbling, L., Littman, M., and Moore, A. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, 4:237–285.

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

[Kalai and Lehrer, 1993] Kalai, E. and Lehrer, E. (1993). Rational learning leads to Nash equilibrium. *Econometrica*, 61(5):1019–1045.

[Kalai and Lehrer, 1994] Kalai, E. and Lehrer, E. (1994). Weak and strong merging of opinions. *Journal of Mathematical Economics*, 23(1):73–86.

[Kalai and Lehrer, 1995] Kalai, E. and Lehrer, E. (1995). Subjective games and equilibria. *Games and Economic Behavior*, 8(1):123–163.

[Lauritzen and Spiegelhalter, 1988] Lauritzen, S. and Spiegelhalter, D. (1988). Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B*, 50(2):157–224.

[Littman, 1994] Littman, M. (1994). Memoryless policies: Theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 238–247.

[Littman et al., 1995] Littman, M., Cassandra, A., and Kaelbling, L. (1995). Learning policies for partially observable environments: Scaling up. In *Proceedings of the Twelfth International Conference on Machine Learning*, page 362. Morgan Kaufmann.

[MacKay, 2003] MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

[Majecka, 2009] Majecka, B. (2009). Statistical Models of Pedestrian Behaviour in the Forum. Master's thesis, University of Edinburgh.

[Meuleau et al., 1999] Meuleau, N., Peshkin, L., Kim, K., and Kaelbling, L. (1999). Learning finite-state controllers for partially observable environments. In *Proceedings of the Fifteenth International Conference on Uncertainty in Artificial Intelligence*, pages 427–436.

[Nair et al., 2003] Nair, R., Nair, R., Tambe, M., Tambe, M., Marsella, S., Yokoo, M., Pynadath, D., and Marsella, S. (2003). Taming Decentralized POMDPs: Towards Efficient Policy Computation for Multiagent Settings. In *IJCAI*, pages 705–711.

[Nair et al., 2004] Nair, R., Roth, M., and Yohoo, M. (2004). Communication for Improving Policy Computation in Distributed POMDPs. In *AAMAS '04: Proceedings*

of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, pages 1098–1105, Washington, DC, USA. IEEE Computer Society.

[Oliehoek et al., 2009] Oliehoek, F., Whiteson, S., and Spaan, M. (2009). Lossless clustering of histories in decentralized POMDPs. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 577–584.

[Oliehoek et al., 2008] Oliehoek, F. A., Spaan, M. T. J., Whiteson, S., and Vlassis, N. (2008). Exploiting locality of interaction in factored Dec-POMDPs. In *AAMAS '08: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 517–524, Richland, SC.

[Oliehoek and Vlassis, 2007a] Oliehoek, F. A. and Vlassis, N. (2007a). Q-value functions for decentralized POMDPs. In *AAMAS '07: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1–8, New York, NY, USA. ACM.

[Oliehoek and Vlassis, 2007b] Oliehoek, F. A. and Vlassis, N. (2007b). 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.

[Papadimitriou and Tsitsiklis, 1987] Papadimitriou, C. and Tsitsiklis, J. N. (1987). The complexity of Markov decision processes. *Math. Oper. Res.*, 12(3):441–450.

[Pearl, 1988] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, USA.

[Peshkin et al., 2001] Peshkin, L., Meuleau, N., and Kaelbling, L. (2001). Learning policies with external memory. *Arxiv preprint cs / 01 03003[CS.LG]*.

[Pineau et al., 2003] Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 1025–1032.

[Pynadath and Tambe, 2002] Pynadath, D. and Tambe, M. (2002). The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, 16(1):389–423.

[Rosenschein and Genesereth, 1985] Rosenschein, J. and Genesereth, M. (1985). Deals among rational agents.

[Roth et al., 2005a] Roth, M., Simmons, R., and Veloso, M. (2005a). Decentralized Communication Strategies for Coordinated Multi-Agent Policies. In *Multi-Robot Systems: From Swarms to Intelligent Automata, volume IV*. Kluwer Avademic.

[Roth et al., 2005b] Roth, M., Simmons, R., and Veloso, M. (2005b). Reasoning about joint beliefs for execution-time communication decisions. In *AAMAS '05: Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 786–793, New York, NY, USA. ACM.

[Roth et al., 2007] Roth, M., Simmons, R., and Veloso, M. (2007). Exploiting factored representations for decentralized execution in multiagent teams. In *AAMAS '07: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1–7, New York, NY, USA. ACM.

[Russell and Norvig, 1995] Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Seuken, 2007] Seuken, S. (2007). Memory-bounded dynamic programming for DEC-POMDPs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence IJCAI*, pages 2009–2015.

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

[Shoham and Leyton-Brown, 2009] Shoham, Y. and Leyton-Brown, K. (2009). *Multi-agent Systems: Algorithmic, Game Theoretic and Logical Foundations*. Cambridge University Press, New York, NY, USA.

[Smallwood and Sondik, 1973] Smallwood, R. and Sondik, E. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088.

[Sondik, 1971] Sondik, E. (1971). *The optimal control of partially observable Markov processes*. PhD thesis, Stanford University.

[Sondik, 1978] Sondik, E. (1978). The optimal control of partially observable Markov processes over the infinite horizon: Discounted costs. *Operations Research*, 26(2):282–304.

[Spaan and Vlassis, 2005] Spaan, M. and Vlassis, N. (2005). Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24(1):195–220.

[Spaan et al., 2008] Spaan, M. T. J., Oliehoek, F. A., and Vlassis, N. (2008). Multiagent Planning under Uncertainty with Stochastic Communication Delays. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 338–345.

[Sycara, 1998] Sycara, K. (1998). Multiagent Systems. *AI Magazine*, 10(2):79–93.

[Szer and Charpillet, 2006] Szer, D. and Charpillet, F. (2006). Point-based dynamic programming for DEC-POMDPs. In *AAAI'06: Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1233–1238. AAAI Press.

[Weiss, 1999a] Weiss, G., editor (1999a). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, USA.

[Weiss, 1999b] Weiss, G., editor (1999b). *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 27–77. MIT Press, Cambridge, MA, USA.

[Welch, 1947] Welch, B. (1947). The generalization of "Student's" problem when several different population variances are involved. *Biometrika*, 34(1-2):28.

[Williams and Baird, 1994] Williams, R. and Baird, L. (1994). Tight performance bounds on greedy policies based on imperfect value functions. In *Proceedings of the 10th Yale Workshop on Adaptive and Learning Systems*.

[Wooldridge and Jennings, 1995] Wooldridge, M. and Jennings, N. R. (1995). Intelligent Agents: Theory and Practice. *Knowledge Engineering Review*, 10(2):115–152.

[Wooldridge, 2001] Wooldridge, M. J. (2001). *Introduction to Multiagent Systems*. John Wiley & Sons, Inc., New York, NY, USA.

[Xuan et al., 2001] Xuan, P., Lesser, V., and Zilberstein, S. (2001). Communication decisions in multi-agent cooperation: model and experiments. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 616–623, New York, NY, USA. ACM.

[Young, 1950] Young, D. (1950). *Iterative methods for solving partial difference equations of elliptic types*. PhD thesis, Harvard University.

[Zhang et al., 2004] Zhang, H., Croft, W., Levine, B., and Lesser, V. (2004). A multiagent approach for peer-to-peer based information retrieval system. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 456–463. IEEE Computer Society.