UNIVERSITY OF SOUTHAMPTON
Faculty of Engineering, Science and Mathematics
School of Electronics and Computer Science

Bayesian learning for effective coordination in uncertain multi-agent systems

A progress report submitted for continuation towards a PhD

Supervisor: Professor Nicholas R. Jennings
Examiner: Professor Paul Lewis

by Mair Allen-Williams

June 8, 2007

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS
SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

A progress report submitted for continuation towards a PhD

by Mair Allen-Williams

Multi-agent systems draw together a number of significant trends in modern technology: ubiquity, decentralisation, openness, dynamism and uncertainty. As work in these fields develops, such systems face increasing challenges. Two particular challenges are decision making in uncertain environments, and coordination with other agents about whom they may have little or no knowledge. Although uncertainty and coordination have each been tackled as separate problems, formal models for an integrated approach typically make a number of simplifying assumptions, and often have few guarantees. One such formal model uses Bayesian methods, as these provide a means of building on available knowledge in order to act optimally in uncertain scenarios. In this report we explore the extension of a Bayesian decision-making model into the multi-agent domain. We demonstrate the effectiveness of the approach on a small coordination problem, and show how it could be applied to ambulance rescue problem inspired by the Robocup Rescue system.

Contents

List of Figures

Chapter 1

Introduction

This section begins with an overview of multi-agent systems, explaining their importance as part of modern developments in technology, and the way in which the need for coordinated decision making under uncertainty arises as a particular need of these systems. To provide a specific grounding for our research, we propose the disaster response domain, explaining some of the features which make it interesting (section 1.1). In sections 1.2 and 1.3 we go on to outline contemporary approaches to the problems of decision making under uncertainty and coordinated decision making, explaining where there are gaps in recent work and motivating the remainder of our work. In section 1.4 we detail our contributions.

Thus, we begin by introducing multi-agent systems. These may be described as systems of interacting intelligent actors, or *agents*, existing in some environment. This environment provides stimulation to the agents' senses, and reacts to the agents' actions. There is no global view, rather, each individual is able to sense part of the system. Such systems are becoming increasingly important as they draw together a number of significant trends in modern technology (Wooldridge, 2002):

Ubiquity: As computing chips become smaller and cheaper it is possible to add computational power and intelligence to many kinds of devices in almost any location. Multi-agent systems made of networks of these ubiquitous devices have much greater possibilities than individual devices. Such systems may also be mobile, adapting quickly to changing surroundings.

Decentralisation: With the advent of the world wide web and other computing networks, systems that distribute data and tasks among a network of machines are increasingly common.

Openness and dynamism: Open systems are those in which agents may enter or leave at any time, while dynamic systems have a constantly changing environment. Many real-world systems need to be both open and dynamic. Thus there has been a corresponding trend in computing away from systems in which one supplies a static problem and waits for a solution, towards providing interactive systems which are able to respond to a changing environment.

Uncertainty: Uncertainty plays a large part in systems which respond to environmental or sensor inputs. Moreover, a trend towards increasingly large and complex systems means that frequently systems are effectively uncertain, even if they are technically deterministic.

A combination of these features describes the kinds of decentralised data and information systems which are increasingly required by many commercial and industrial organisations. Moreover, multi-agent systems can be used to implement or to model all or part of such systems. Example application areas are as diverse as modelling eBay auctions (Rogers et al., 2006), modelling social structures (Sun and Naveh, 2004), or creating fight scenes in films (agent systems were used in *The Lord of the Rings*[1]). Consequently, multi-agent research is a lively and growing area facing many challenges.

In particular, agents acting in real environments frequently find themselves facing high degrees of uncertainty, about their current environment and about any other agents which might exist in the environment. Moreover, as these systems get larger, the full state of the environment may not be completely observable, adding to the agent's uncertainty. Thus agents in such environments will typically carry out some kind of discovery phase to determine the essential characteristics of the scenario, before they attempt to achieve their own goals. This discovery phase in a multi-agent system is tightly linked with the presence of other agents in the system. In particular, other agents with related goals may be prepared to share any information they have or discover. Even if such agents do not share the ultimate goals of our agents, they may share subgoals (such as determining the state of the environment) and thefore be prepared to cooperate. An agent exploring such an

[1] `http://www.massivesoftware.com/what_massive.html`

environment should therefore incorporate the behaviour of other agents into its plans. Beyond discovery, there will continue to be interaction between the agents in a multi-agent system. This may be explicitly cooperative or obstructive (so agents explicitly include the goals of others when making decisions). Conversely, there may be no explicit coordination, but because the actions of each agent will inevitably have the potential to interact with each other, the behaviour of other agents must be taken into account.

Now, this general problem of taking others' behaviour into account, *coordination*, is a key issue in a multi-agent system. In uncertain and open systems, fixed protocols for coordination must function against a background where agents are not fully aware of the situation; the resources available to them, or the presence or goals of the other agents. The negotiation of coordinated behaviour in such systems may be intertwined with the discovery phase, as agents discover the details of the other agents, and perhaps cooperate with other agents in determining properties of the situation.

Against this background, we aim to build upon existing techniques for decision making under uncertainty, explicitly modelling other agents, thus tackling the problem of providing coordinated behaviour in uncertain and partially-observable multi-agent systems. To provide a specific grounding for this research, we consider the domain of disaster response. As we outline in the next section, this has all of the characteristics relevant to the growing field of multi-agent research.

1.1 The disaster response domain

In disaster situations such as terrorist attacks, floods or earthquakes, many different teams from a number of organisations must cooperate to attempt to recover the situation. Their work may be interrupted by self-interested actors such as journalists, scavengers, or even terrorists. Moreover, some of the cooperating organisations may have conflicting subgoals—for example, suppose during an aeroplane crash an injured person is trapped in the wreckage very close to the "black box". The police will wish to keep the black box intact for the purposes of determining what caused the crash, while ambulance teams are concerned only with removing the injured person, perhaps necessitating the destruction of the black box unless they are very careful. The overall goal of both, of course, is something akin to maintaining the wellbeing of the people affected by the disaster or who might be affected by related disasters.

Scenarios of this nature provide rich grounds for the implementation of agent systems, such as the Robocup Rescue system[2], the DEFACTO system (Schurr et al., 2005) and others (e.g. (Burke, 2003), (Takeuchi et al., 2003)). In such applications, the extent of computer intervention lies on a scale between a fully automatic multi-agent system, and a human-managed system making use of agent systems. At one end of this scale, multi-agent systems can model every aspect of the disaster response, simulating the disaster, the affected humans, and the response agents. Robocup Rescue is an example of such a system. In the future, these systems could be taken further, deploying actual robots at the scene of the disaster. Indeed, there is already some work on human-robot teams (Schurr et al., 2005). At the other end of the scale, agent systems can be used alongside the human response teams, processing data and interactively suggesting courses of action (Dorais et al., 1998). In the middle of the scale can be found agents who defer to humans in scenarios they are uncertain about (Scerri et al., 2004).

The focus in this work is on the use of multi-agent systems for modelling aspects of a complete disaster response. This provides the broadest perspective on the problem. Complete solutions can be sought, and the resulting models used in more human-interactive applications. For example, applications in which the automatic system's function is to propose courses of action which may be explored by the human user. Another use for such models is in training human teams. For example, the Auckland urban search and rescue department are working together with Robocup Rescue developers to develop new strategic models for their own rescue services [3].

Taking this complete disaster response problem as the illustrative domain for exploring multi-agent systems motivates a number of important domain requirements. The following are all key properties of disaster response scenarios which should be taken into consideration by any coordination algorithm grounded in this domain:

Large: Disaster recovery scenarios may involve hundreds or thousands of distinct actors, organisations or teams, operating over a wide area.

Dynamic: It is unreasonable to assume that a realistic system will be static. Environmental conditions are subject to constant change and agents must be able to adapt to these changes. In disaster recovery scenarios agents must

[2] `http://www.rescuesystem.org/robocuprescue/`

[3] `http://www.auckland.ac.nz/uoa/about/news/articles/2004/06/0011.cfm`

react to changing weather, unexpected events such as building collapse or fires and constantly moving traffic, among many other changing conditions.

Open: Systems of this nature will have agents moving in and out of the system constantly. In the worst case, in disaster scenarios agents are liable to die, hence vanishing suddenly. On the other hand, as volunteers and taskforces from elsewhere rush to contribute help, new agents will enter the response system.

Decentralized: Providing a central server is equivalent to reducing the system to a single-agent system—all the decision making can in principle be carried out by the central server; the central server has a complete view of the system, and so on (Panait and Luke, 2005). Furthermore, in large and dynamic systems of the kind we are investigating, providing a central controller is likely to be infeasible: there are unlikely to be the resources to allow communications between one central controller and every other node, one central controller is almost certainly not going to be able to obtain a complete view of the system, and the potentially rapid changes as agents enter and leave the system would be difficult to track.

Uncertain: As previously discussed, large, dynamic, open systems typically have inherent uncertainty. Even if the system is technically predictable, the complexity in the system is likely to make it effectively uncertain. For example, in disaster recovery scenarios taking place over broad areas, it is unlikely that any agent will have a complete view of the situation. Moreover, information which reaches the agent may be error-prone, increasing the uncertainty. At a different level of granularity, environmental conditions such as the expected weather or the height of a tide can be equally uncertain.

Heterogeneous: There are many different types of agents involved in a disaster response scenario, with a variety of capabilities and (potentially conflicting) goals. At a minimum there will be the rescue teams, each with distinct tasks: ambulances, police, helicopter teams, and there will be the people affected by the disaster. Involved may also be journalists, crime teams, environmental agencies, to name but a few.

Bandwidth-limited: One characteristic which is common in disaster scenarios is limited communication (Committee on using IT to Enhance Disaster Management, 2005). For example, mobile phone networks may become jammed, so that only a fraction of the messages initiated are able to reach their destination.

Competitive: As discussed, the actors at a disaster situation may have conflicting goals or subgoals, as in the example above. Furthermore, self-interested agents have no reason to attempt to resolve the conflicts cooperatively.

As can be seen, there are many challenges when working in such domains. However, the essential task of any agent operating in a multi-agent system is to process the inputs it receives, and to plan how to act, in the context of other agents. The central tasks for the agent are, therefore, information processing and coordinated decision-making (including decisions about information gathering).

The first of these two tasks, information processing, is, in its fullest sense, the task of forming a coherent world view as scattered, incomplete, potentially error-prone, even conflicting messages reach the agent at different times from heterogeneous sources. The extent to which the agent actually needs a complete world model will depend on its decision making policy. For example, if agents in a disaster situation are organised in such a way that each agent is allocated to a particular region (in the UK, this might be a county) and functions only in that region, it may choose to maintain a model only of that region and discard messages which concern other regions. In section 2.1 we briefly discuss the various information fusion techniques that are relevant to this activity.

Our primary interest, however, is how such agents gather and then make use of their information in a multi-agent setting. Acting optimally in such settings involves the integration of two established disciplines: decision making under uncertainty, and coordinated decision making. Each of these is dealt with in more detail in the following subsections.

1.2 Decision making under uncertainty

Let us first consider a single agent acting in its environment. The agent perceives the state of the world through some kind of sensory inputs, and makes a decision about how to act based on this state. Following the agent's action, the world transitions into a new state, and the agent may receive some *reward*. This model forms the basis of reinforcement learning theory (Sutton and Barto, 1998). Underlying reinforcement learning theory is the assumption that the immediate next state is dependent only on the previous state and choice of action—the Markov assumption. While this property does not hold for many realistic scenarios, it is a

sufficiently good approximation that the learning techniques which arise from this theory often get good results, as demonstrated by many practical examples such as (Hoar, 1996) (Smith, 2002a), (Abul et al., 2000).

With the Markov assumption, if the transition and reward models are completely known to the agent, the system can be solved, using the recursive Bellman equations (Sutton and Barto, 1998), to determine the expected optimal action from each world state. When there is uncertainty about these models, the agent must integrate the learning of the models (exploration) with acting to obtain rewards (exploitation). Reinforcement learning techniques such as Q-learning, TD(λ) and SARSA (Sutton and Barto, 1998) provide variations on ways of updating the estimated value of states and actions. The integration between exploitation and exploration is then achieved by selecting an action stochastically based on the current state and the learned state-action values. These techniques provide a good way for agents to make decisions (and hence act) in scenarios where there is uncertainty about the consequences of their actions and the environmental dynamics.

Within all such techniques, the aforementioned action selection is achieved using straightforward heuristic rules. For example, *ϵ-greedy* selection chooses the "greedy" action—the action currently believed to be the best—most of the time. With some small probability ϵ, a different action is chosen at random from the remaining actions. An improvement on ϵ-greedy selection is to base the probability of selecting *any* action proportionate to its estimated value—Boltzmann selection is an example of this approach, choosing an action a with probability proportional to $e^{Q(s,a)/\tau}$ (where s is the current state of the agent, $Q(s,a)$ is an estimate of the value to the agent of taking action a from state s—taking into account the consequences of the action on likely future states—and τ is some problem-specific constant). Both of these approaches have the disadvantage that action choices will never converge even as the model becomes more accurate. This can be mitigated by reducing ϵ and τ over time, so that action choices approach the deterministic choice. A more principled approach, however, is to make the variance in action choices directly dependent upon the certainty the agent has in a *model* of the system.

In more detail, learning methods may be described as either *model-based* or *model-free*. In model-free methods, the agent estimates the value of state-action pairs or of states, but does not maintain an explicit estimate of the transition or reward models. Such model-free methods typically involve simple updates at each step. By contrast, model-based methods update the transition and reward models at

each step. Model-based methods can be used to carry out many simulation steps alongside each real-time step, taking advantage of otherwise idle cpu cycles. Another advantage of model-based methods is the ability to bias the system towards a particular real model, or to re-use parts of the model on different problems. Here, we focus on model-based methods particularly because of the latter property: in disaster scenarios we expect to have initial beliefs about the situation based on domain knowledge or similar disasters, and would like to be able to incorporate these beliefs into our solutions.

Extending model-based approaches is a Bayesian approach to learning. Rather than maintaining a point estimate of a model, possibly with some associated certainty, an agent maintains a probability distribution over possible models (Dearden et al., 1999) (a *belief state*). Referring back to the problems with heuristic action choices, we see that in this method they are eliminated. This is because an agent can compute for every action, its expected value of its action given the belief state. This computation explicitly incorporates the uncertainty in the agent's beliefs, resulting in a principled approach to the problem of integrating exploration and exploitation. It is such a principled approach which will form the grounding for our own work, providing a theoretical base for decision making under certainty and exploring how this base can be practically extended into larger domains. However, fully-observable single agent problems form only a small subset of the kinds of problem which are motivated by the target domain outlined in section 1.1. Over the rest of this section we describe the extension of these techniques into more challenging problems as motivated by our example domain.

1.2.1 Partial observability

It is not always possible to observe the complete world state. More commonly in problems such as a disaster rescue, any particular agent will make a sequence of observations which allow it to make some inferences about the current state. In a disaster rescue these might be detailed local observations from an agent on the ground, combined with local observations communicated from other agents, or they might be (for example) more abstract observations from a helicopter passing over the disaster region. Given a set of local observations, many possible global states may be compatible with these local observations (a phenomenon described as *perceptual aliasing*). Commonly, the underlying process (moving from global state to global state) will still be believed to be Markov. Such problems are

described as *partially observable Markov decision processes*, or POMDPs, and there is a host of techniques for solving them.

First, if the underlying environmental model is known, a POMDP can be converted to a continuous Markov decision process by defining a *belief state* as a probability distribution over states. The resulting MDP can be solved using exact algorithms (Cassandra et al., 1997) or using approximations to make computation easier (Amato et al., 2006), (Kim et al., 2006). If the underlying model is not known, learning techniques must be used to refine a solution as the agent explores the system. Model-free approaches such as (Aberdeen and Baxter, 2002) have found some success in using learning techniques to solve POMDPs. However, we believe that model-based approaches may again have benefits. To this end, (Shani et al., 2005) demonstrates a model-based algorithm which handles perceptual aliasing using variable length suffix trees—these trees address the fact that even if state transitions are Markov, the observed process may not be. However, these approaches rely on a number of approximations and assumptions about the state space, hence are not entirely satisfactory. A principled alternative may be to extend the Bayesian model described previously into partially observable domains and this is the approach we describe in chapter 3.

Finally, these approximate solutions highlight a general difficulty in complex problems; that of successfully learning in a large state space. Reinforcement learning is a technique inspired by human behaviour, and we can look again to human behaviour to see how large state spaces may be managed. In the next section we give a brief overview of some common techniques.

1.2.2 State generalization and function approximation

Generally speaking, there are two main techniques that are used in decision problems with huge state spaces. The first, *abstraction* is to decide actions based on a high-level or abstract view of the state space, ignoring irrelevant details or coalescing several states together—for example, when cleaning a room, if one walks into any of the walls then an appropriate action is to reverse, regardless of which wall it is.

The second technique, *function approximation* is to use experience from states *similar* to the current one in deciding the immediate action. This may be achieved by partitioning or clustering the state space in some way (Sutton and Barto, 1998), or it may be achieved by mapping a high-dimensional space into a lower

dimensional one (Roy and Gordon, 2002). In simple reinforcement learning, agents learn tables of states and values. *Function approximation* replaces these lookup tables for state values with functions (such as neural networks or radial basis functions) which map from state variables to values. Online supervised learning techniques learn the parameters given the experience in (state, value-estimate) pairs. Function approximation may also be used to learn behavioural policies directly, taking features of the state as an input and outputting an action—this is generally done using classification techniques on the state, with the classes corresponding to actions.

In general we will need to use both abstraction and function approximation in order to act in the complex environment of disaster response. However, we do not focus on these problems within the current work, Rather, we make use of simple existing techniques—manually defining abstractions, and using neural networks for function abstraction where necessary. For our future work more sophisticated techniques will become necessary.

To sum up, in this section, we have outlined reinforcement learning and its extension to partially observable state spaces. However, in order to act in our target multi-agent domain, all agents must make decisions in the context of other agents. In the next sections we discuss current approaches to such *coordinated* decision making.

1.3 Coordinated decision making

Agents functioning in uncertain worlds among other agents may simply include their behaviour in the world model they develop. However, this may lead to inaccurate assumptions about the world, as other agents adjust their own behaviour. Maintaining models of the world and of other agents separately provides greater flexibility and may enable the agent to reuse a world model as agents come and go, or reuse agent models in fresh scenarios. Furthermore, although in this report we emphasize coordination as a part of a decision making process, if the other agents are modelled separately, a coordination mechanism can be studied distinctly or parts of the coordination mechanism (a communication protocol, for example), can be supplied separately. In some cases, this may be useful—for example, allowing agents to coordinate for resources, and then make individual decisions within the constraints of this coordination. Finally, if an agent maintains its models of the other agents separately from the environment, it can reuse these models if it

meets the same agents in a fresh environment, and it can adapt its overall model to an open system. It is therefore interesting to explore ways in which agents can coordinate when they have explicit models of the other agents in the system.

To this end, we outline each of the three potentially overlapping coordination mechanisms identifed be Boutilier (1996):

Conventions can be the simplest form of coordination. In a convention-based coordination system, there are a number of assumed "social rules" defining how agents interact when they are aware of other agents. There are many real-word precedents for coordination by convention. For example, traffic control is frequently based around conventions such as stopping at red lights, or travelling faster in the right-hand lane of a motorway than the left-hand lane. Such coordination has the advantages of being simple and requiring no setup time (Fitoussi and Tennenholtz, 2000). However, it is inflexible, and relies on all participants knowing the conventions and complying with them.

A potentially more flexible extension to coordination by conventions is a role-based organisation. Each role is associated with a set of conventions. Role-based structures have been successfully implemented for teams such as Robocup soccer teams (Tambe et al., 1999). However, this work is not applicable to very open domains as it considers teams which are fixed over the relevant time period. To handle dynamic open domains, self-organising structures can be used (Wang, 2002). Self-organisation can also be applied to uncertain domains: organisations can restructure as the agents obtain new information about the environment. However, we see this as a particular application of learning and so do not explore this technique in any further detail.

Communication is another common human coordination technique. Coordination through communication has a small setup time and some bandwidth costs. It requires a common language, and the flexibility of this language determines the flexibility of the resulting coordination. There is potential for probabilistic models of language (Fischer et al., 2005), permitting (for example) adaptation to changing environments. Alongside a language for coordination, agents must have some means of reasoning internally about the outcomes. The nature of the coordination will thus depend considerably on the agents' internal coordination models. In any large system, such as our focus domain, there must be some form of communication in order to share information between agents; it will be impossible for any one agent to

sense all the information it needs to function effectively in context (Dutta et al., 2004). However, we expect to make limited use of communication beyond this information-sharing, as the bandwidth restrictions will preclude it in most cases.

Learning techniques extend the single agent learning outlined previously into the multi-agent domain. The uncertainties of our target domain make learning techniques a natural approach to problems within this domain—agents must learn about the situation they are in. Such techniques offer potential for evolving coordinated polices within uncertain state spaces, either with a group of learners exploring the space and converging towards an equilibrium (as in (Claus and Boutilier, 1998) and (Littman, 1994)), or by one agent explicitly learning about the behaviour of others in order to adapt its own appropriately (Chalkiadakis and Boutilier, 2003). These learning methods may incorporate learning about what and when to communicate (Dutta et al., 2007), or learning about conventions (Kazakov and Bartlett, 2004). Conventions could also be used to provide strong prior assumptions about other agent behaviour within a Bayesian learning model such as (Chalkiadakis and Boutilier, 2003) (the multi-agent extension to the model of Dearden et al. (1999) previously discussed).

Furthermore, agents may apply learning methods to only parts of a coordination problem; for example, learning to act given a fixed communication protocol, learning to communicate given a fixed action model, or learning to choose between specific coordination protocols based on the tradeoffs between setup costs and effectiveness in the current environment (Excelente-Toledo and Jennings, 2005). Therefore, even given the complexity of applying a learning technique to a complete problem, it may be appropriate to consider learning for a part of the problem, especially in uncertain or dynamic conditions. This motivates our focus on learning techniques in this report.

As well as the three approaches to coordination identified above, another research domain which investigates coordination from a theoretical angle is **game theory** (Leslie, 2004). Agents which are playing some game try and derive, through exact evaluation or learning, a *best response* to the strategies of the other players in the game. If all the players iteratively keep playing best responses, and if strategies are *mixed* (stochastic) the play will converge to a (mixed) equilibrium. One of the challenges of game theory is to direct play so that convergence is not just to any

equilibrium but to an optimal one (Claus and Boutilier, 1998). In our context this may be expressed as, say, leaving a disaster scene with all lives saved, rather than all lives lost.

Within the domain of game theory, multi-agent learning in which the agents maintain explicit models of the other agents may be considered to be learning in stochastic games. An effective approach to extending single-agent reinforcement learning into this setting is the *win-or-learn-fast* (WoLF) approach: an agent's learning rate is adjusted according to its current performance. (Chalkiadakis and Boutilier, 2003), already mentioned, builds on (Dearden et al., 1999) to improves upon WoLF techniques using a Bayesian model: agents maintain beliefs about the behaviour of the other agents, as well as a probability distribution over world models. This eliminates the need for the use of heuristically determined learning rates, as well as allowing the use of prior information about agents.

Considering the above coordination techniques in the light of our domain requirements; particularly uncertainty, partial observability, dynamism and openness, we conclude that "acting" and "coordinating" in uncertain systems should be completely integrated. That is, rather than an explicit coordination mechanism, agents should include their beliefs about other agents' behaviour in their action selection mechanism, adjusting their own behaviour according to their beliefs about the other agents. Moreover, we believe that such an integrated approach should be based on sound theoretical principles. Such a basis allows us to reason about properties of the approach, such as optimality or convergence (of agent behaviour).

In uncertain and dynamic domains, this motivates the use of multi-agent learning models, since these provide a basis for such coordinated action selection and are designed for uncertain domains. Furthermore, although in large domains it may be impractical to learn a complete solution in real time, we have explained that learning methods can be used on top of other coordination mechanisms to provide adaptability on top of known conventions or communication languages, to select between coordination mechanisms, or to use learning for some subproblem. We therefore believe that it will be advantageous to explore the application of multi-agent models to domains with the properties outlined in section 1.1.

In particular, as a point of departure we consider the model of (Chalkiadakis and Boutilier, 2003) in which agents maintain probability distributions over models. This has been proven to be effective on small test problems, including some problems not handled well by previous multi-agent learning mechanisms (see (Chalkiadakis and Boutilier, 2003) for more details). However, the model is only defined

for the fully observable case: agents can see the actions of all the other agents, and ascertain deterministically what the current state is. Furthermore, it has not yet been tested on large domains; the sample problems have two agents and half a dozen states. In our work, we address in detail the former shortcoming, and outline a possible approach to address the latter. In the following section we outline this research contribution in more detail.

1.4 Research contributions

In this report we shed some light on the issues associated with coordination in challenging domains as defined above, and go on, in particular, to describe an approach to coordinating in uncertain domains which fuses, in a principled way, the dual problems of learning about the domain and learning about the other agents.

Our approach is completely decentralized and is particularly applicable to uncertain, bandwidth-limited scenarios involving heterogeneous agents. Although our model assumes a static situation for guarantees to hold, in section 3 we discuss briefly the implications of applying the model to certain classes of dynamic scenario. In our work, the agents are completely cooperative—that is, they have identical goals. However, in section 5.2 we show how the approach could be extended to domains in which there are competitive agents. We also suggest a way of extending our model to open systems. Through this work, we make the following novel contributions to the state of the art:

- An extension of the model in (Chalkiadakis and Boutilier, 2003) to multi-agent systems in which the actions of other agents are partially observable. This covers two cases: First, agent actions are only sometimes observable, but inferences can be made about likely actions from observable state changes (for example, if a pile of rubbish goes flying past my window, it is likely that someone is flinging rubbish from one of the the windows above). Second, the effects of agent actions are consistently observable, and the agent has a model of the likely effects given an action. This could arise, say, in a disaster occurring at sea, where a rescue plane drops rescue packages. From the wind conditions and the progress of the packages, the intended targets for the packages can be modelled. This might be useful, for example, for another 'plane which is coordinating with the first over a period of time, to

estimate where packages will be dropped at the next disaster location and so plan its own actions.

- Previous work on coordination in uncertain and partially observable domains treats the other agents as a part of the environment. Using the above models, we demonstrate that explicitly modelling the other agents' behaviour can result in effective learning, thereby extending and improving upon the previous work.

- An extension of the model in (Chalkiadakis and Boutilier, 2003) to the case where states may be partially observable. As discussed, in many realistic domains it is unreasonable to assume that the agents will be able to fully observe their situation. However, the existing formal work on coordinating in uncertain domains with such partial observability is limited to non-learning environments (e.g. (Emery-Montemerlo et al., 2004), (Matsuno et al., 2001)). By extending the Bayesian learning model into such domains, we provide a basis for a principled approach to coordinated action in these domains.

  Since this model is sufficiently complex to be impractical for use, we outline the use of state abstraction and approximation techniques into order to apply our model to a practical problem.

The combination of these contributions is a model for coordinated decision making in rich and challenging domains, with high levels of uncertainty. This model is based on a well-founded approach, giving us confidence in its correctness and a set of guarantees about its behaviour in small systems. The model is extended into larger-scale systems using abstraction techniques, demonstrating its practical effectiveness. The system is intended to be very flexible in its applicability, guiding all or part of agent behaviour in both cooperative and competitive systems. In section 5.2 we discuss several interesting directions for development of this work.

1.5 Report structure

The rest of this report is structured as follows:

- In section 2, we begin by describing the salient features of the kinds of agents we will use. We then discuss the techniques which enable such agents to plan and act under uncertainty, and give an overview of how the state of the art

extends these models into partially-observable multi-agent domains. Within this section, we focus on a particular Bayesian learning system (Chalkiadakis and Boutilier, 2003) and discuss its suitability in our target domain. Since applying learning techniques in complex state spaces typically involves some approximations or abstractions of the state space, we outline some effective techniques for achieving this. We also describe other approaches to coordination and their relation to multi-agent learning.

- In section 3, we extend the previously described Bayesian learning system to partially observable systems and propose a state abstraction algorithm to make the model tractable for potentially large problems.

- In section 4, we instantiate the model of section 3 on a simple coordination problems (section 4), providing a preliminary demonstration of its effectiveness as a solution technique.

- Section 5 concludes the report. We outline the way in which our model could be extended to address more of the domain requirements in section 1.1, suggesting a number of directions for further study (section 5.2) and giving a timeline for the immediate work (section 5.3).

Chapter 2

Literature Review

In this chapter, we detail the current state of the art for agents acting in the context of uncertain multi-agent systems with the features discussed in section 1.1. We begin by outlining key properties of the intelligent agents which act in such systems (section 2.1). This provides a background for the rest of the chapter. In section 2.2 we describe learning models for decision making under uncertainty. Section 2.3 then develops these techniques into the multi-agent domain and discusses the coordination therein. In section 2.4 we discuss the general issue of how problems with large state spaces can use approximation techniques to render them tractable. Finally, section 2.5 summarises the chapter and motivates the work described over the rest of the report.

2.1 Autonomous agents

For the purposes of our work, we assume that an agent is an entity which is situated in some environment and reacts to that environment, in order to try and achieve some objective or goal (this definition is based on (Wooldridge, 2002), chapter 1). Such agents will be able to logically reason about their actions and their effects with respect to the current state of the world and the agent's goal. At times, inconsistencies and conflicts in agent goals and beliefs may crop up; the agent's reasoning mechanisms must have some means of resolving these. We believe that probabilistic methods provide a realistic way to do this for a number of reasons. First, such techniques are effective for reasoning in uncertain scenarios where an agent may want to reason over some degree of belief in a particular property.

Second, probabilistic representations are typically more compact than their logic-based counterparts for both the input data and the agent models (Stenning and van Lambalgen, 2005).

This reasoning mechanism will concern both the signal the agent receives from its environment, and the effects of its own actions. The agent may form an explicit world model or it may leave the model implicit as it reasons about plans. Explicit models have more potential for reasoning about states and behaviours as they store more information explicitly. However, maintaining explicit models may be computationally and memory intensive (Excelente-Toledo and Jennings, 2005). Despite this, we believe that the aforementioned benefits of explicit models justify their use where practical. In section 2.2, where we describe the use of learning for decision making under uncertainty, we revisit this issue in the context of specific learning models. If, as in many disaster arenas, the world is large and detailed, agents may only be able to model explicitly small parts of it accurately, due to space constraints. In such worlds, it is therefore appropriate either to use a model which can store information at different levels of detail, or to reason in a simplified abstract world. In section 2.4 we explain the possibilities for achieving either of these.

Finally, as well as reasoning about their environment, agents in a multi-agent system will interact with each other. This interaction can be modelled by defining a (hyper-)sphere of influence for each agent within the environment. Overlapping spheres of influence indicate interactions between agents (Wooldridge, 2002). A model of how different spheres interact will form a part of the agent's model of the system, as will models of the behaviour of the other agents. Once again, there will be tradeoffs between making these models explicit or implicit. Making decisions in the context of these other agents is the fundamental principle of coordination (Durfee, 1999). Clearly, this is a central part of a reasoning agent in a multi-agent system. In section 2.3 we show how to extend models for reasoning under uncertainty to incorporate explicit considerations of the other agents.

Thus, the in following sections we expand in detail on the dual aspects of making reasoned decisions under uncertainty, and making coordinated decisions in uncertain scenarios.

Example 2.1 Fire-damaged building

A fireman searches an old people's home after a fire. As he works his way up the building, he takes increasing care how he treads, as there is no knowing what kind of structural damage the fire may have caused to the building. At any point, placing his foot on a beam or floorboard may cause parts of the building to collapse. As part of his search, the fireman is removing valuables from the building. Different wings of the building provided different levels of comfort and were correspondingly priced. The rooms in the more expensive wings contain far more valuables than those in the cheaper areas, hence are typically more rewarding for the fireman to search. However, he does not initially know which wings are which, only slowly discovering the rewarding areas as he makes his way through the building.

2.2 Decision making under uncertainty

Example 2.1 demonstrates a scenario in which an agent may have to act without initially knowing the consequences of his actions. In the next subsection we introduce reinforcement learning as a means of incorporating experiences into a world model in order to make reasoned decisions under this type of uncertainty. We highlight a Bayesian (probabilistic) technique as a principled approach to reinforcement learning. We then go on in subsection 2.2.3 to discuss the extension of these techniques into partially observable domains.

2.2.1 Reinforcement learning

We consider, as in example 2.1, a single agent acting in an environment. The agent perceives its state through some kind of sensory inputs, and makes a decision about how to act based on the state. As a result of the agent's action, the world will transition into a new state. After each action, the agent may accrue some *reward*. A key feature of such problems is the notion of delayed reward—states which have no or negative reward, but which ultimately lead to higher rewards. This is a common feature of many real world problems (the fireman may travel through many rooms, potentially slipping and hurting himself, before he finds a wing where many valuables are salvageable).

Determining the reward achieved from a particular action may be straightforward—the fireman above may receive a point for each valuable he retrieves. However, in some kinds of problem deciding a reward function may be trickier. For example, following an earthquake, buildings may be burning while humans are buried and

trapped. A reward function for a team simultaneously rescuing humans and extinguishing buildings may try and put relative values on the buildings and the human lives, supplying some reward for unburnt buildings and some for live humans. We do not discuss this issue further, but simply suppose the existence of a known reward function (which in disaster response scenarios will typically be defined in terms of human lives).

Given this context, the goal for the agent is to maximise some function based on the reward obtained. This may be (Sutton and Barto, 1998):

- over a fixed time horizon
- during an *episode* in which the agent continues to act until some termination condition is reached
- the average reward over an indefinite time period, or
- the total reward over some time period

In the last case, more recent rewards may be valued more highly than earlier rewards—in particular, this encourages adaptation to nonstationary environments. In disaster scenarios, we may consider either the total reward accrued (perhaps in number of lives saved) when some termination condition is reached (the scene is cleared up), or we may consider how efficiently our agents can act to accrue reward over a fixed time period. For the purposes of this work, at the moment we focus on the former which we believe is a natural formulation for this kind of scenario—rescue agents will typically work towards a particular termination condition rather than for a fixed time period.

Specifically, in a reinforcement learning problem we define a finite set of states S, a finite set of actions A, and a finite set of rewards R. The environment dynamics are then defined by:

1. A transition probability function, $P(s'|s,a)$. This defines the probability of reaching state s' from state s given that the action performed was a.
2. A reward probability function $P(r|s,a)$. This defines the reward achieved by taking action a from state s.

The agent's decisions are made according to a policy π, where $\pi(s,a)$ defines the probability the agent will take action a from state s.

In this model, the probability of transitioning to a particular state, or achieving a particular reward, are conditioned only on the current state, and not on any previous states or actions. This is therefore a Markov decision process as described in section 1.2. Within a Markov decision process, if the transition and reward probability functions are known, then it is possible to derive the optimal policy using the recursive *Bellman equations:*

$$V_\pi(s) = \sum_a \pi(s,a) \sum_{s'} P(s'|s,a)\{E[r_{t+1}] + \gamma V_\pi(s')\} \tag{2.1}$$

$$V^*(s) = \max_\pi V_\pi(s) \text{ for all } s \in S \tag{2.2}$$

and

$$Q_\pi(s,a) = E\{r_{t+1} + \gamma V_\pi(s_{t+1})\} \tag{2.3}$$

where $V(s)$ denotes the value to the agent of being in state s, given both its immediate reward and the discounted future rewards it may achieve from that state. $Q(s,a)$ denotes the value to the agent of being in state s and taking action a, given the immediate reward and the expected value of the resulting state. γ is a problem-specific discount factor determining how "farsighted" or "myopic" the agent is. In each state s, the action a is chosen which maximises $Q^*(s,a)$. If the model of the environment is available, these equations form, in principle, a soluble system of simultaneous equations. Such systems can be solved iteratively, for example using *dynamic programming* techniques (Sutton and Barto, 1998).

This provides a theoretical grounding. However, in our example domain the dynamics of the environment are likely to be unknown to the agent, the inherent uncertainty of any large system possibly exacerbated by the effects of the disaster. Thus, it must learn the optimal policy through exploration of the state space.

As discussed in section 1.2, in this context learning algorithms may be categorised as model-based or model-free. Model-free algorithms are the more popular approach, requiring simple updates to $Q(s,a)$ or $V(s)$ estimates at each step. Examples include Q-learning, TD(λ) and SARSA (Sutton and Barto, 1998), all of which provide variations on how to adjust the estimate given the most recent experience. However, in section 2.1 we argued that model-based methods can be more powerful. Specifically, in many applications, agents may have spare cpu cycles during a timestep; for example while waiting for environmental input, while

carrying out a motor action, or if a timestep corresponds to some fixed unit of real time. In such applications, agents that maintain a model of the environment may use these spare cycles to simulate actions based on their model, and refine their policies accordingly. Providing the models are sufficiently accurate, this can result in much faster convergence to the optimal policy (Sutton and Barto, 1998).

Furthermore, model-based algorithms permit the use of a prior model to guide agent learning—although doing so can be a disadvantage if unwanted bias is introduced (Dearden et al., 1999) (Sutton and Barto, 1998). In a disaster scenario problem, such a prior model may be advantageous because agents can enter the scenario with some initial model based on previous knowledge of the area and previous disaster experiences, and then learn from the current experience to refine this model and hence their behaviour.

Finally, model-based algorithms may use the Bellman equations above to derive the optimal behaviour for the estimated model, or the iterative techniques derived from these equations (Sutton and Barto, 1998). However, deciding behaviour based on a point estimate of the model ignores a key variant: the agent's uncertainty about its estimate. The uncertainty in the estimate should affect both the caution with which the agent behaves, and the decisions it makes about trading exploratory actions (investigating unknown regions of the environment) with exploitative actions (those which it believes will accrue high reward). We refer back to example 2.1 as a demonstration of each of these points. Firstly, if the fireman is unsure about his estimated model of his current region, he must step forward cautiously so as to jump back if a board falls away underfoot. Secondly, if the fireman has found a wing filled with jewellery, but has left two wings unexplored, it may be that one of those wings contains far more expensive jewellery than the current one. In the following section we describe a Bayesian model-based technique which explicitly includes these uncertainties in the agent model.

2.2.2 Bayesian model-based learning

Example 2.2 gives another example of the exploration-exploitation tradeoff. As the agent continues to explore the state-action space and so its model becomes more accurate, it should tend towards exploiting its knowledge and away from exploration. In point-based estimation models, various heuristics are used to select actions, selecting the action believed best (the "greedy" action) some fraction of the time (where the fraction depends on problem parameters) and exploratory

Example 2.2 Exploration-exploitation in New York

An earthquake occurs in New York, destroying many of the buildings on the east side of the Hudson. Ambulances from the west side rush to the rescue. Unfortunately, the earthquake has also destroyed several of the bridges across the river. An ambulance arriving at the riverside early after the disaster learns over the radio that there is a bridge still standing two miles downriver. However, there is no data about the bridges upriver. The ambulance driver knows that there is a bridge only half a mile away, if it is still standing, and another a mile and half away, but then no more bridges for five miles. The decision the ambulance driver must make about whether to travel in the uncertain direction, or head straight for the bridge which is known to be standing, is an example of an *exploration-exploitation* problem.

actions the rest of the time (Sutton and Barto, 1998). However, if the agent maintains, rather than a point model of the dynamics, a probability distribution over possible models, then the optimal action, whether exploitative or exploratory, can be determined without recourse to heuristics or problem-specific parameters. Specifically, the expected value of a particular action is given by:

$$E[Q(s,a)] = \int_M Q(s,a|M)P(M)$$

where M denotes a possible model, and $Q(s,a|M)$ is the Q-value given that particular model.

For an agent maintaining a probability distribution over models in this way, the Markov decision process is defined by the transition from probability distribution to probability distribution. Each state in this process is a probability distribution over states in the world; such states are described as *belief states*. The transitions between belief states are determined by Bayes' rule:

$$P(model|observations) \propto P(observations|model)P(model)$$

with the observations being the state and reward signals arising from each environmental transition. However, these MDPs do not have a finite state space, so cannot be solved using reinforcement learning techniques for finite state spaces. Instead, it is necessary to use some means of approximating a solution—for example in (Dearden et al., 1999), sampling techniques are used, using a finite number of candidate MDPs at each step when estimating the optimal action from the current state.

This probabilistic model is a well-founded approach to decision making within single-agent problems in which the state of the world is known at all times, but the environmental dynamics and in particular the effects of agent actions are uncertain. However, the scope of our illustrative domain is broader than this: in particular, we expect that frequently agents will not be able to observe the full state of the world but must make inferences based on partial observations. In the next section, we discuss approaches to decision making in systems where the state of the world is uncertain.

2.2.3 Partial observability

Example 2.3 London bombings

On 7 July 2005, a coordinated attack took place covering several locations in London. As events progressed, observers at each of the locations only gradually developed a picture of what was happening. At each incident, it was not initially clear what had occurred—a bomb, a fault? The connections between incidents came later and gradually. Similarly, natural disasters such as earthquakes lead to high degrees of uncertainty about the environment, both localised (the number of people trapped in a particular building) and widespread (the structure of roads may change completely).

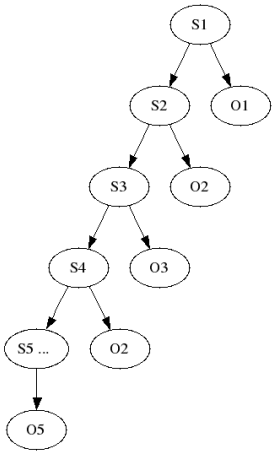


FIGURE 2.1: A simple POMDP

In the previous examples, the agent's learning concerned the effect of its actions on the environment and the rewards resulting from its actions. However, in many problems, as illustrated in example 2.3, it is not the effects of actions on the world state which are unknown, but the world state itself. In these partially observable problems (POMDPs), we assume the existence of a fixed, known model $P(s|o)$ where o is the current set of observations and s is a possible state. Although

the sequence of states is defined by an MDP, the sequence of observations is not. Consider the simple example in figure 2.1: if the current observation is $O2$, then the probability that the next observation will be $O5$ differs depending on whether the previous observation was $O3$ or $O1$.

The correct approach to finding the optimal policy in a POMDP is to maintain a continuous *belief state* containing the (Bayesian) probability of being in each state given the current observations (Kaelbling et al., 1998). Given the current belief state and an observation *obs*, we can obtain the new belief state by computing the probability of each possible state t:

$$\begin{aligned} P(t|obs, b) &\propto P(obs|t, b)P(t|b) \\ &= P(obs|t)\sum_s P(t|s)P(s|b) \end{aligned}$$

where s denotes a previous state and the belief state b defines $P(s|b)$. The belief state is the resulting pdf $P(t)$. Hence, a sequence of belief states does have the Markov property, thus forming a continuous MDP which can be solved exactly by exploiting its properties—the value functions are convex and piecewise linear (Kaelbling et al., 1998). Intuitively, there is a "piece" of linear value function for the policy tree arising from each possible state, and the value function for the state is the upper surface of all these segments. The witness algorithm (Kaelbling et al., 1998) is based around this notion, but does not scale to large problems. Incremental pruning (Cassandra et al., 1997) addresses some of the efficiency problems with the witness algorithm, but generally exact solution methods do not scale well and are thus not appropriate for real-world systems of the kind we are trying to address.

A more scalable approach to solving such continuous MDPs is to compute approximate value functions for belief states, exploiting the intuition that a large part of the belief space need never be visited. Techniques include point-based sampling (Izadi and Precup, 2006) and myopic evaluation (only looking ahead at the values of the next one or two states) (Chalkiadakis and Boutilier, 2003). A more recent method uses quadratically constrained linear programs to describe locally optimal policies (Amato et al., 2006), with promising results. Another novel and interesting technique is the use of principal components analysis (PCA)[1] to map the

[1]For an explanation of PCA, see for example (Bishop, 2004), chapter 8

belief space into a low dimensional space, carrying out the planning in this low dimensional space (Roy and Gordon, 2002).

In this work, we propose to use a combination of techniques, sampling and myopic evaluation, since high dimensional state spaces will result in high dimensional belief spaces, necessitating several approximation techniques to become tractable. In future work, we may investigate the combination of the PCA technique with other approximation techniques, as it appears to be an elegant way of reducing the state space while retaining the most important information.

However, none of these techniques are immediately applicable to our problem, since in the POMDPs described above, the transition probabilities $P(t|s)$ must be known to the agent. However, an agent entering a new, uncertain scenario will not necessarily know these probabilities and they may need to be learnt alongside the computation of the optimal actions.

As in the fully observable case, current solution techniques may be model-based or model-free. For example, Baxter and Bartlett (2000) propose a direct policy approach, using gradients to incorporate a performance measure into the learning. similar techniques are described in (Aberdeen and Baxter, 2002), in which the agent uses Monte-Carlo methods to learn through interaction with the environment.

By contrast to such model-free methods, model-based methods are developed around the insight that a POMDP is a form of hidden Markov model (HMM), in which a sequence of states gives rise to a sequence of observations. POMDPs have the added complication of including actions, but HMM solution techniques such as the recursive Baum-Welch equations (Roweis, 2003) can be adapted. However, this solution form uses a complete dataset (sequence of observations) and so is not appropriate to incremental learning. In realistic problems, such as we face here, an agent must develop and update its policy as it explores the environment, so some form of online learning is necessary. An alternative to the use of Baum-Welch updates is the use of short-term memory trees to provide model updates (Shani et al., 2005). Such trees contain variable-length sequences of observations, in order to handle the non-Markov properties of POMDPs. This approach can be integrated with an incrementally improving policy.

The above techniques rely on a number of approximations and assumptions about the state and hence are not entirely satisfactory. We propose, as an alternative, to extend the Bayesian model of the previous section (2.2.2) into this POMDP

domain. This formulation falls naturally into the belief-state space of POMDPs, and will provide the advantages of the Bayesian model-based methods (explicitly handling uncertainty, making use of prior knowledge) in this domain. In section 3.4.1 we give the details of this model.

However, in many examples of large and partially observable problems, the learning agent is not acting alone. We must therefore explore the generalisation of the above approaches into multi-agent systems. This is the focus of the remainder of this chapter.

2.3 Learning in multi-agent systems

Example 2.4 Traffic jams in New York

Consider again the ambulance driver arriving at the Hudson after an earthquake. If he is the only ambulance approaching the scene, he may choose not to take the risk of having to travel many miles up river, and head straight for the bridge which is known to be standing. However, if he knows that there is a fleet of ambulances following him, he may choose to head upriver so that he can send back data about the status of the bridges to later ambulances, enabling them to update their model without the travel costs. He might also consider that if all the ambulances were to head for the one bridge, a traffic jam would form there, perhaps wasting precious time.

Clearly, when several agents are functioning within a system, the interactions between their behaviour are relevant to the decisions they make. Example 2.4 illustrates this, extending the example of the previous section (2.2.2) into the multi-agent domain.

Generally speaking, there are two main approaches to the extension of single-agent learning into such multi-agent systems. The first approach, generally applied to cooperative systems, is to consider the problem as a whole, with the ultimate aim of finding optimal joint actions, although the implementation may be decentralized with each agent learning separately. This is the typical focus of work described as "multi-agent reinforcement learning" (MARL) (Panait and Luke, 2005). In our work, the focus is on the way in which an agent (or team of agents) operates in an environment in the context of other agents. However, as in many disaster scenarios, this context may be an assortment of agents, each of whose behaviour is determined by its own controlling algorithm. Furthermore, these agents may not all be cooperative. Therefore, solutions which rely on all agents behaving the same way and having the same goal are not appropriate to this kind of problem.

The second approach, more appropriate to our domain, is described as "learning in games" and arises from the addition of learning methods to game theory, extending single-agent learning with estimates of best responses to the other agents. In the case that all the agents in some game are learning best responses, they should converge to a Nash equilibrium. A further challenge then is to direct the "play" so that not just any equilibrium is achieved, but an optimal one.

There is a great deal of overlap in the two approaches. Indeed, recent work explicitly explores the relationship between them (Rezek et al., 2006), proposing new algorithms for game play inspired by machine learning techniques, and improving machine learning techniques with insights from game theory. Over the next sections, we describe extensions from the single agent into the multi-agent world and discuss how the techniques relate to the uncertain heterogeneous domains we introduced in section 1.1.

2.3.1 Learning in games

The first issue to consider as we extend our learning agent models into explicitly multi-agent domains, is what we hope our agents will achieve (Shoham et al., 2003). A typical target is the Nash equilibrium of the underlying stochastic game. However, in large and complex systems, a NE may not be a useful target—finding an equilibrium may be too costly, for example. Furthermore, finding a NE is only possible if all the players are adjusting their strategies towards this goal. In heterogenous settings, such as the disaster response domain, we cannot ignore the possibility that some agents may have, say, naïve fixed controllers and all we can do is make our behaviour a best response to these.

More realistically, therefore, we are likely to seek learning algorithms capable of finding a *satisfactory* solution which have desirable properties over an indefinite period of play (Lesser, 1999). Such properties may include: convergence (of rewards, of actions, of strategies), rationality, and no-regret—this property means that a learning algorithm should not allow itself to be exploited by malicious opponents (Bowling, 2005). Which of these are more relevant is problem-specific. For example, the no-regret property can be ignored if all the agents are known to be cooperative, while convergence is less important to a constantly changing scenario.

The WoLF approach, exhibited by (Bowling and Veloso, 2001) and its extension GIGA-WoLF (Bowling, 2005), is an effective way of adapting single-agent learning

in a multi-agent domain. "Win or learn fast" agents do not explicitly consider the other players, but adjust their learning rate according to how good the current policy seems. This results in a particular agent settling into a particular policy if it seems to be working well, but quickly changing it when other agents adapt to it. In a cooperative domain where all the agents use the same system, this algorithm converges to the equilibrium. The algorithm may also be useful useful in open or dynamic domains as it provides a rule to adapt to changing circumstances without explicitly modelling the circumstances.

However, the WoLF approach must decide an appropriate learning rate given the apparent performance of the algorithm, where the learning rate parameters are problem specific heuristics. Better performance may be achieved by maintaining a probability distribution over the agent strategies, and taking this distribution into account in computing a best response (Chalkiadakis and Boutilier, 2003). By maintaining a distribution rather than a point estimate, the agent can account for the uncertainty in its beliefs. This is a model-based approach, therefore has the advantages that have previously been discussed for model-based approaches, including incorporation of prior knowledge.

In more detail, in the multi-agent system of Chalkiadakis and Boutilier (2003), explicit models of the other agents are also maintained. The agent's belief state, b, now contains: a probability distribution over the environment model, a probability distribution over strategies, the most recent state and action choices, and a history trail which contains as much state as is necessary to accurately model the other agents' strategies[2]. At each timestep, the agent (supposing it to be agent i) selects the action a_i which leads to the greatest expected value, given the above beliefs. In this model, the expected value is computed by the following Bellman equation over the belief state (Chalkiadakis and Boutilier, 2003):

$$Q(a_i, b) = \sum_{\mathbf{a_{-i}}} P(\mathbf{a_{-i}}|b) \sum_{s'} P(s'|a_i \circ \mathbf{a_{-i}}, b) \sum_{r} P(r|s', a_i \circ \mathbf{a_{-i}}, b)[r + \gamma V(b < s, \mathbf{a}, r, s' >)]$$

where

[2]Determining the relevant history assumes some knowledge about the other agent strategies. If the agent maintains insufficient history, then its model of opponent strategies will never be accurate and thus the results will be suboptimal. In the case where the underlying process is Markov and where all the agents are learners trying to converge to an equilibrium, we may safely assume that the previous state contains sufficient history to obtain an accurate model.

$$V(b) = \max_{a_i} Q(a_i, b)$$

and $b < s, \mathbf{a}, r, s' >$ is the updated belief state which arises from being in the belief state b, in which the current state is s, and carrying out joint action $\mathbf{a}$, resulting in new state s' and reward r. The new history trail and state in this fresh belief state are trivial. The environmental and strategy models are updated using Bayes' rule.

One key assumption made by this model, which we have already disputed for the single-agent case, and which becomes increasingly improbable as the problems get larger, is full observability. In multi-agent problems agents will frequently have some local observations which are not available to any other agents. Therefore, over the next two sections we discuss the extension of POMDPs into the multi-agent domain (2.3.2), and reinforcement learning algorithms therein (2.3.3).

2.3.2 Multi-agent POMDPs

Example 2.5 Ambulance rescue in New York

Consider once again the ambulance driver of examples 2.2 and 2.4. In example 2.2, the driver received information about the status of each bridge over the radio as it was discovered. However, suppose that there are several fleets of ambulances operating on this scene, each using a different radio frequency. Information between the fleets is no longer completely shared, resulting in each fleet having a slightly different picture of the scene.

Example 2.5 describes a scenario in which agents may make local observations which are not known to the other agents. Each agent in such a scenario may have a distinct estimate of the state, or probability distribution over states, even if all had the same prior knowledge. As before, we focus on POMDPs as a useful model with an associated body of theoretical knowledge.

In section 2.2.3 we have already shown that there is considerable complexity in solving POMDPs. Adapting solutions to the multi-agent environment is therefore a challenging problem. However, one of the key differences between multi-agent and single-agent POMDP solutions is that if the environmental dynamics are known, single-agent solutions can be computed offline. However, in a multi-agent environment where agents must respond to the behaviour of others, the policy *must* be computed online. The advantage of computing policies online is that policy searches can be directed only into the regions of the state space which are actually visited (Paquet et al., 2005).

Just as fully observable single-agent learning can be extended to the multi-agent case by treating other agents as part of the state (section 2.2.2), so multi-agent POMDPs can be solved using any online POMDP algorithm, treating knowledge about the other agents as part of the state. The branch-and-bound algorithm in (Paquet et al., 2005) is an example of such an algorithm. By incorporating problem-specific knowledge into the search (for example, treating some variables as static in the short term), this algorithm is demonstrated to work on larger scale problems. Other methods can be used to improve efficiency such as local factorisation (Kim et al., 2006) and Bayesian network representations (Sallans, 2002).

In the ambulance example above, an ambulance driver does not need to visit a region which has been visited by other ambulances. If he has an explicit model of the behaviour of the other ambulances, he can combine it with his own observations of the scene and any nearby vehicles in order to optimise his own behaviour. This would be an instantiation of a model-based approach. As the problems become larger (more states, more unknowns, more agents), the difficulties of the model-based approaches (computational and memory intensity) begin to render them less useful. Nonetheless, there is some existing work in this area.

In more detail, a formal model-based approach to solving multi-agent POMDPs is to extend game-theoretic models into *partially observable stochastic games* (POSGs). A POSG consists of: a finite set of agents I, a finite set of states S, a finite set of actions A, a finite set of observations O, an initial state distribtion and a set of Markovian transition probabilities $P(s', \mathbf{o}|s, \mathbf{a})$, and a set of reward functions, $R_i : S \times A \rightarrow R$. If the dynamics of the system are known, then in principle the POSG can be solved to determine the optimal action for a particular agent given a set of beliefs about the other agents. Whereas a POMDP is solved by conversion to a belief-state MDP and then solving the resulting continuous MDP, a POSG is complicated by the need to include beliefs over the other agents' belief states. Hansen et al. (2004) provide a solution method for POSGs which finds an equilibrium by iterating through all the agents, repeatedly removing any dominated strategies from the agent's strategy space. More computationally tractable approximations have also been proposed, such as (Emery-Montemerlo et al., 2004), in which the full POSG is approximated by a series of one-step POSGs.

The above POSG solutions, and similar, are offline approaches, and appropriate only to finding equilibria in cooperative games. Thus they are not immediately

applicable to an agent learning to act in an unfamiliar situation, even where the dynamics are known. However, the approximation techniques of (Emery-Montemerlo et al., 2004) may be useful for problems of the kind we are addressing.

The descriptions above focus on problems in which the *state* is partially observable. In many kinds of partially observable multi-agent POMDPs, including the ambulance example (2.5), it may also be impossible to consistently observe the actions of the other agents. We can classify such scenarios into two types. In the first type, as in our ambulance example, we simply cannot see the actions of every agent, for example because the agents have a limited field of vision. In the second type, we can observe the effects of actions, but not the intended action itself. For example, in many robotics problems, actions may not be completely deterministic; an agent aiming to travel in a particular direction will have some probability of successfully doing so, and some probability of travelling in another direction. Example 2.6 suggests a rescue scenario with this property.

Example 2.6 Partially observable actions at sea

A shipwreck occurs on a stormy sea as a ship travels around the Western Cape of South Africa. A 'plane drops rescue packages and inflatable liferafts. Unfortunatey, the wind blows the packages about as they fall towards the water and they often do not reach their intended targets.

Now, in a single-agent problem, the agent would model this second case within the transition function, the non-determinism of actions swept into the probability $P(s'|s,a)$. However, in a multi-agent problem, it may be to our advantage to model the underlying strategy of the opponents. Then, if the agents are upgraded (or in the shipwreck example, if the weather conditions change) so that $P(effect|action)$ is changed, but their strategies remain unchanged, the learning agent can adapt its behaviour as soon as the new $P(effect|action)$ model is known. Of course, the other agents may well modify their behaviour based on the new model; still, we hope that having this explicit model may give us a head start in keeping up with them. Likewise the first case is typically handled by treating the problem as a single agent problem, and the behaviour of the other agents as a part of the environment. However, we propose that explicitly modelling the behaviour may be advantageous, reaping the benefits, already discussed, of explicit models.

Finally, these solution techniques are appropriate only if the environment and strategy models are known to our agent. Our focus, however, is on uncertain systems. Furthermore, in a scenario in which the other agents are adapting their behaviour, it will be necessary for us to constantly update our models of the other

agents' strategies. Although understanding the case where the models are known will provide us with useful building blocks, we must explore how agents can learn strategies in unfamiliar situations. This is the subject of the next section.

2.3.3 Learning in multi-agent POMDPs

Example 2.7 Fire-damaged building: multi-agent extension

Recall the fireman of example 2.1, exploring a fire-damaged old people's home. Suppose that instead of acting alone, the fireman is only one of several men carrying out such a search. Furthermore, despite all advice to the contrary and their lack of safety training, two of the old folk have joined in. Alongside the firemen, two policemen carefully traverse the building looking for indications of arson. In this example, we have a multi-agent problem in which several types of agent must interact and take each other into account—for example, permitting each other's passage through doorways as they meet, or avoiding putting excess weight on damaged floorboards. Some of the agents aim to cooperate—the firemen spread out, each searching a different region of the building. Others—the policemen—have different goals. Not all of the agents behave predictably or rationally; one of the old folks gets confused at times. Finally, neither the complete state of the building, nor the location or even number of other people in the building is known to each agent at any one time.

Finally, we consider agents acting in our target domain: namely a multi-agent domain where there is uncertainty and partial observability. In example 2.7 we extend our original example into this domain. Acting in such challenging domains, we draw on the work in learning and multi-agent learning, in POMDPs and in POSGs. We may take either multi-agent learning as a starting point, and extend it to the partially observable domain, or we can consider ways of integrating learning with our POSG solutions.

In fact, very little of the previous work has been extended to this difficult problem. However, card games have formed a testing ground for applying learning techniques to partially observable competitive games. Although in card games such as hearts (Matsuno et al., 2001) or poker (Shi and Littman, 2002), the environmental model is known to the agent, the behaviour of the other players is not necessarily known. Over many games, agents can estimate the behaviour of the other players, and use these estimates alongside POMDP solution techniques to learn to play effectively. However, these card game techniques have predictable transition functions, thus are not directly applicable to our domains with uncertain transition functions.

Furthermore, as we have developed the theory of learning in MDPs through increasing layers of complexity, correct and complete solutions to the problem become ever more intractable. Algorithms use tricks such as factorisations (Sallans, 1999), assume independences (Kim et al., 2006), and repeatedly approximate (Roy and Gordon, 2002) (Chalkiadakis and Boutilier, 2003), leaving the original principled approaches some way behind. If these tricks are executed carefully—if the factorisations are correct, the assumptions not too far from reality, and the approximations directed by the problem structure, then the solutions found may not fall too far behind the optimum. In the next section we discuss ways of reducing the state space in learning problems, in order to render such problems more tractable.

2.4 Managing large state spaces

Example 2.8 Earthquake

Following a large earthquake, a fireman observes a number of fires from the watchtower at his station. Although he can guess that there will be a number of other fires across the city, he can also see immediately that large areas of the city are impassable where roads have collapsed or buildings have collapsed onto the roads. He therefore concentrates his rescue efforts only on the reachable parts of town. Once he has decided (perhaps in collaboration with other firefighters) which fire he will tackle first, and travelled there, his focus grows even narrower, taking in the details of this fire in as much detail as possible, but requiring no information about other fires in the area.

At the same time, a helicopter observes the scene from above. The pilot's task is to report on the state of the city. Unlike the firefighter on the scene, the pilot is uninterested in details of the fires, recording only an overview. In the helicopter, all fires are categorised into a small number of size classes (such as *small, medium, large*) and approximate coordinates recorded.

In section 2.2.3 we briefly mentioned approximate methods such as sampling for computing value functions in continuous MDPs. Example 2.8 illustrates that frequently agents will either be interested only in a high-level view of the state space (as the helicopter pilot in the example), or in some particular region of the state space (as the allocated firefighter). It is therefore reasonable to reduce the state space in either of these ways, selecting a reduction approach appropriate to the particular problem.

A related issue in managing large or continuous state spaces is that in similar states similar behaviours are often appropriate. This permits an agent to estimate an

appropriate action for many unvisited states. In this section, we describe current techniques for both parts of managing large and/or continuous state spaces.

2.4.1 State space abstractions

In section 1.2.2, we introduced the two related issues of abstractions and function approximation for managing large or continuous state spaces. There are many methods for both generalization and function approximation, and work on the two areas frequently overlaps. However, we will deal with each in turn.

First, abstractions are a mapping from a large intractable space into a smaller one. There are several spaces which can be mapped in this way within a learning problem: combining several states into a single abstract state (such as hand type in poker), combining several actions into a single action, or combining several rounds (of a game) or timesteps into a single step (Pfeffer et al., 2000). We focus on state abstraction techniques, since the size of the state space is typically the bottleneck in the applications we consider. However, many of the approaches we discuss are equally applicable to other problems.

In more detail, the simplest state abstractions are those in which the state space is just broken up into tiles or buckets, for example by laying a grid over the state space. A slightly more sophisticated method of dividing the state space is to form clusters using standard clustering techniques such as nearest neighbours or k-means (Hoar, 1996). A slightly more sophisticated clustering technique makes use of a topological mapping (Smith, 2002b) which exploits the form of the state space to provide more clusters in denser parts of the space.

As a slightly more intelligent alternative in a high-dimensional input space, the fact that many combinations of input variables rarely or never occur can be exploited by mapping the input space into a higher level feature space. Such feature spaces may also be used to encode intuitive knowledge about the structure of the state space. Automatic means of encoding include coarse coding (Sutton and Barto, 1998), in which ellipses or other overlapping partitions in the state space represent binary features, and dimensionality reduction techniques such as PCA (Roy and Gordon, 2002). Features may also be learned through supervised learning, exploiting the intuition that humans can recognise abstractions intuitively, but not always easily define them (Tanner et al., 2007). Alternatively, features can be manually defined (Fogel, 2002). This manual definition will be sufficient for our

initial work, although we propose to explore more automatic ways of reducing the state space in the future.

Finally in some kinds of problem, such as that in example 2.8, different levels of abstraction may be appropriate. This motivates the use of hierarchical learning techniques (Fischer et al., 2004). Such techniques integrate action and state space abstractions, mapping high-level states to high-level actions. Depending on the particular action, its execution may require traversing the hierarchy to consider more details of the state or of a localised part of the state. In a large real-world problem such techniques may become necessary, although we do not address them further in this initial work.

We now turn to functional approximations. The above techniques all define abstractions over the whole state space before beginning to explore that space. However, we do not always know beforehand what characteristics of a state determine its "similarity" to other states. Rather than pre-define a set of classifications, we can use state variables as the inputs to functions which may compute: (1) the value of the state, (2) if states and actions are input, the value of the state-action pair or (3) the optimal action for the state. This last case is closely related to clustering, since essentially the function acts as a classifier.

Such functions can be learned by assuming some form for the function, and then learning the appropriate parameters. It is possible to use any standard method which is able to handle incremental learning (the accumulated experience forms the training data) and nonstationarity. In such situations, neural networks are often used (Sutton and Barto, 1998). In particular, a neural network with two hidden layers can be used to approximate any function with an arbitrarily small error, given sufficient training data. Alternatives to neural networks include radial basis functions or linear approximations (Sutton and Barto, 1998). For most general problems we consider that neural networks will be satisfactory. In this work, we do not do any form of functional approximation, but we expect it to prove necessary in as we experiment in larger domains in the future.

When carrying out functional approximation in this way, the state variables themselves may be used as inputs to the function. However, if there are a large number of state variables it may be helpful to reduce them in some way. Furthermore, in any difficult problem, the learner may not find a good approximation within a reasonable time unless the input features are carefully selected to provide guidance (Fogel, 2002). We therefore believe that the abstraction techniques described above should be combined with functional approximation techniques, using, say,

cluster centres or manually defined features as the inputs to the neural network (Smith, 2002b), (Fogel, 2002), although we leave this to future work.

2.4.2 Abstractions in POMDPs

State abstraction techniques such as clustering and feature mapping can be applied in partially observable domains to reduce the underlying state space. However, methods which operate during exploration can no longer be applied to the underlying state space, as the exploration is being carried out in belief-space. In principle, any approach suited to continuous space can be used to approximate POMDPs. However, POMDP belief space has a particularly sparse structure. Typically, given a particular observation o_i, there will be a small set of states S_i which could have given rise to the observation. For a set of observations $O = o_1, ..., o_n$, the set of states which could have given rise to the observations is given by $S_O = \cap_{i=1..n} S_i$. Belief in any states other than these will be small or zero. A nice approach to exploiting this sparsity is to use some form of dimensionality reduction to map belief space into a lower-dimensional feature space (Roy and Gordon, 2002).

Finally, as well as addressing partial observability, we must use techniques which are applicable to multi-agent domains. In general, the abstraction techniques of the single agent domain are equally applicable to multi-agent domains (Abul et al., 2000). Indeed, in large partially observable domains it is likely that we will only be able to attempt to find solutions by combining abstraction techniques. One possible approach would be to first reduce the state space using a learned mapping from low level percepts into a higher level feature space, then use the variant of principal components analysis (PCA) described in (Roy and Gordon, 2002) to reduce the belief space induced over these features, and finally use functional approximation techniques to estimate the value of states in the reduced space. In section 3.5 we outline an algorithm which combines abstraction techniques in this way.

However, a disadvantage of these approaches is that a certain amount of preprocessing is necessary. In the PCA-based algorithm outlined above, it is necessary to do some sampling of state and belief space before the reduction techniques can be applied. This is not always feasible in real, dynamic scenarios. Less principled approaches such as manually defining abstractions from human input may turn out to be more practical.

2.5 Summary

In this chapter we have emphasized uncertainty as a key requirement which we intend to address, and discussed agent decision making under uncertainty, highlighting a principled Bayesian approach for the single agent problem. Since coordinated multi-agent systems are our goal, we pursue the multi-agent extension to this approach, emphasizing its correctness and its flexibility. However, in order to use the complete Bayesian approach on anything other than toy problems, and especially if we wish to scale it up to large problems, some compromises or approximations will be necessary. To this end, we discuss ways in which the Bayesian model can be adapted to larger state spaces. However, in these larger state spaces, agents are unlikely to have full observability, which the models of (Dearden et al., 1999) and (Chalkiadakis and Boutilier, 2003) both assume.

We have therefore extended the discussion of learning techniques into partially observable domains, noting that there is very little work on learning within partially observable domains which explicitly takes into account the behaviour of the other players—which we believe would lead to better performance. We have considered other coordination mechanisms which could be used in such domains, and demonstrated that in large and uncertain domains, learning techniques are likely to be integrated with any other coordination mechanism.

Given this, we propose to develop a model for Bayesian learning which explicitly considers the other agents and which is appropriate to partially observable domains. In order to render such a model computationally feasible, especially in domains with large state spaces and many agents, a number of approximations will be necessary. Using the Bayesian model of (Chalkiadakis and Boutilier, 2003) as a foundation, we develop the formal principles of our model in the next chapter, and suggest some simple approximation techniques.

Chapter 3

A Bayesian model for coordination in partially observable systems

In the previous chapter we motivated the use of a Bayesian learning model as a principled approach to the problem of acting under uncertainty, in the context of other agents. However, previous work on such models focuses on fully observable problems, while many realistic problems have some degree of partial observability. In the the rest of this section we describe an extension of the model into partially observable domains, and suggest a method for managing state space abstraction in such domains. The chapter is arranged as follows: section 3.1 gives a formal outline of the problem. In section 3.2 we recap the model which we will extend and explain the use of Bayesian network diagrams in understanding this and similar models. Section 3.3 extends the model into domains where actions are partially observable, while section 3.4 describes the extension of the model into domains where states are partially observable. Finally, in section 3.5 we outline some approximations to facilitate computational tractability.

3.1 Problem definition

Formally, we define

- A set of agents ("players") $I = \{I_1, ..., I_p\}$.

- A finite set of states $S = \{s_1, ..., s_n\}$.
- A finite set of actions $A = \{a_1, ..., a_m\}$. We use light text, a_i to refer to an action selected by a single agent i, and bold text, to refer to a joint action of all the agents, $\mathbf{a} = (a_1, ...a_n) \in A^n$
- A finite set of rewards, $R = \{r_1, ..., r_q\}$
- A reward pdf $P(r = R|s \in S, \mathbf{a} \in A^p)$ [1].
- A transition pdf $P(s_{t+1} = s'|s_t = s, \mathbf{a_t} = \mathbf{a})$. Together with the reward pdf, this describes the environmental dynamics M.
- Strategies for every non-learning agent $\sigma = \{\sigma_1, ..., \sigma_p\}$

In uncertain scenarios the first four items above are known to this agent while the latter three are not. This represents the multi-agent reinforcement learning problem introduced in previous chapters. We will describe a strategy for a single learning agent.

Objective: The agent aims to maximise the common *expected discounted future reward:*

$$\sum_{t=0}^{\infty} \gamma^t r_t$$

where r_t is the reward obtained at time by the system. The parameter γ controls the tradeoff between immediate and future rewards: if γ is close to 1 then future rewards are given almost as much importance as current rewards (the agent is said to be *farsighted*) while if γ is close to 0 then only immediate rewards are given much consideration (the agent is said to be *myopic*). Although ultimately we are interested in finite-horizon scenarios, this quantity provides a simple objective for our initial work. We can estimate it by calculating the average discounted reward over a sequence of timesteps.

As described in previous chapters, we assume that the world evolves in a step-wise fashion. At every step, each agent selects, based on its current strategy and its observations from the state around it, an action from the available actions. The

[1] For the moment, we assume a cooperative system in which all the agents have the same rewards

transition model for the environment determines the resulting state, and some reward is emitted according to the reward model, which all the agents are able to observe.

In each of the following sections, the underlying intent of our learning agent is to achieve the given objective using the following variant of the standard learning algorithm:

1. Initialise prior estimates for M and σ in the form of probability distributions over possible dynamics and strategies. Let P_M be the probability distribution over models M and P_σ the probability distribution over strategies σ.
2. Specify the current belief state, b, to contain the current state, the aforementioned distributions and, if appropriate, any state or action history involved in the strategy models.
3. Determine the action which has the greatest Q-value (as defined in section 2.2.1) given this belief state: if P_M is the probability of an MDP M and P_σ the probability of a strategy σ, then we take
$$Q(a, b) \approx \int_{M,\sigma} Q(s, a|M, \sigma)P(M, \sigma|b)$$
4. Carry out this action, and observe the actions of the other agents where observable, the changes in state (locally or globally), and the reward obtained
5. Given the observations, update the estimates P_M and P_σ for all M, σ, using Bayes' rule $P(M|obs) \propto P(obs|M)P(M)$.
6. Return to 2 and repeat.

Over the next sections, we discuss how the Bayesian updates and the definition of $Q(a, b)$ may vary under differing observability assumptions within the model. We begin with a recap of the fully-observable case, outlined in section 2.3.1.

3.2 Recap: Bayesian multi-agent learning

The aim of this Bayesian model is to supply a principled approach to the exploration-exploration problem, explicitly taking into the account the effects of discovery in

the value estimate for a state. In the multi-agent learning model, the agent's *belief state*, b contains the following:

- A probability distribution over the transition and reward model: a single model M represents the distribution $P(s', r|s, \mathbf{a})$ and the belief state contains $P(M)$ for every such M.
- A probability distribution over strategies for each agent: the strategy σ for agent j is the set of distributions $P(a_j|h)$ (one for each history trail $h = [s_1, a_1, ...s_n, a_n]$) and the belief state contains $P(\sigma)$ for every such σ.
- The current state s
- The current history trail $h = [s_1, a_1, ...s_n, a_n]$

At each timestep, the agent selects the action a_i which leads to the greatest expected value, given its beliefs. The Bellman equation for this system is (Chalkiadakis and Boutilier, 2003):

$$Q(a_i, b) = \sum_{\mathbf{a_{-i}}} P(\mathbf{a_{-i}}|b) \sum_{s'} P(s'|a_i \circ \mathbf{a_{-i}}, b) \sum_{r} P(r|s', a_i \circ \mathbf{a_{-i}}, b)[r + \gamma V(b < s, \mathbf{a}, r, s' >)]$$

where

$$V(b) = \max_{a_i} Q(a_i, b)$$

and

$$P(a_j|b) = \int_{\sigma_j} P(a_j|s, \sigma_j) P(\sigma_j) \tag{3.1}$$

$$P(r, s'|\mathbf{a}, b) = \int_{M} P(r, s'|s, \mathbf{a}, M) \tag{3.2}$$

(For any specific model, we assume that we will factorise $P(r, s'|s, \mathbf{a}, M)$, computing $P(r|obs)$ and $P(s'|obs$ separately).

Finally, $b < s, \mathbf{a}, r, s' >$ defines the belief state which arises from being in the belief state b in which the current state is s, and carrying out joint action $\mathbf{a}$, resulting in new state s' and reward r. The new history trail and state in this fresh belief state are trivial. The environmental and strategy models are updated using Bayes' rule to obtain (Chalkiadakis and Boutilier, 2003):

$$P(M|obs) \propto P(s', r|\mathbf{a}, s, M)P(M) \tag{3.3}$$

$$P(\sigma_j|obs) \propto P(\mathbf{a_j}|h, s, \sigma_j)P(\sigma_j) \tag{3.4}$$

where *obs* are the observations: the joint action and the state transition.

To understand this, it may be helpful to consider the Bayesian network diagram representing the system (figure 3.1). In all our Bayesian network diagrams we use the convention that clear nodes represent hidden variables and filled nodes represent observed variables (evidence). For any particular variable set $V = v_1, ...v_n$, with evidence E and hidden variable set H we can compute the marginal probability:

$$P(V|E) \propto \sum_H P(V, H, E)$$

The sparse Bayesian network offers insights on how to factorise $P(V, H, E)$ to exploit the conditional independences in the system (see, for example, (Mitchell, 1997), chapter 6 for a full explanation of how to do this). From the figure, we can see clearly how the two updates (equations 3.3, 3.4) arise.

This completes the specification for the fully-observable multi-agent learning model using Bayesian belief states, first described in (Chalkiadakis and Boutilier, 2003) and outlined previously in section 2.3.1. In section 2.3.1 we also identified properties which may be desirable in a learning algorithm, such as convergence and no-regret. In particular, in co-operative settings, convergence properties can allow us to determine whether a group of learning agents would be guaranteed to find an optimal solution to the problem. We believe it is useful to investigate all the important properties (convergence, rationality, no-regret) of a learning model. Therefore, we now briefly examine the convergence properties of the above model, leaving further investigation of its properties for future work.

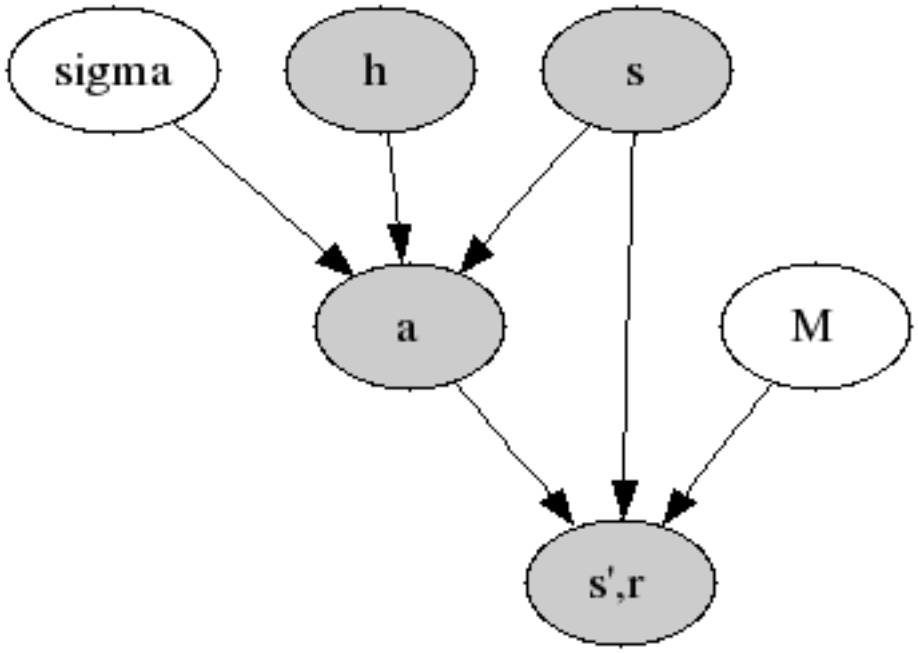


FIGURE 3.1: Bayesian network diagram

First, consider the environmental model and the models of the other players separately within a fully-observable system. If the environmental model is assumed to be correct, then the algorithm is reduced to simulating fictitious play, and the agent's strategy will converge to a stable policy under the same conditions as fictitious play algorithms (Fudenberg and Levine, 1998). Conversely, if the other players' strategies are known, then the beliefs about the environmental model will converge to a point estimate providing: (1) The system is static and the Markov property holds, (2) the learning rate reduces over time, and (3) every (state, action) pair is visited sufficiently often (Sutton and Barto, 1998).

In fact, condition (2) is not relevant for our model (which doesn't have an explicit learning rate), so providing condition (1) holds for the environment, and the other agent strategies are such that condition (3) holds, then the environmental model will converge regardless of the strategy models. Conversely, the strategy models will converge given the relevant fictitious play conditions, regardless of the environmental model.

However, in a dynamic system, clearly there are no general guarantees of convergence—indeed, convergence would be the wrong thing for a dynamic system as there is no static point to converge to. However, let us consider the special case in which the environment is static but the agents all have learning strategies. Providing the strategies include some degree of exploration, the conditions will hold so that

the environmental model is learned. If the environmental model is known, then under certain conditions, fictitious play for all the learners will eventually reach a (mixed) equilibrium (Fudenberg and Levine, 1998). In environments where there is more than one equilibrium, there is no guarantee that the equilibrium reached will be optimal.

While this is not ideal, there is no way to force convergence to an optimal equilibrium in general. However, in many larger problems (as our example domain), optimality is not necessarily a key concern. Furthermore, convergence itself may not as important as the speed at which a "satisfactory" solution can be found. For example, it is of no use figuring out the perfect order for rescuing victims from an earthquake site long after all the victims have died.

Another factor in larger problems, discussed in section 2.3.2, is that the full observability assumption made above begins to break down, with neither actions nor states necessarily being fully observed by any one agent. Instead, agents will make observations which allow them to make inferences about the unobserved parts of the process. Over the rest of this chapter, we develop a reinforcement learning model for partially observable scenarios, extending the above model of Chalkiadakis and Boutilier (2003).

3.3 Partially observable actions

As discussed in section 2.3.2, it is not always possible to fully observe the actions of the other agents, however it may be possible to make some inferences about what the actions were, if we have a partial world model. We therefore modify the above to account for partial observability of actions. We consider the two cases discussed in 2.3.2: in section 3.3.1 the case where the underlying action is not visible, but its effects are, and in section 3.3.2 we outline the case where not all actions can be observed, but some effects on the state may be visible.

3.3.1 Action-effect model

In this setting, we assume that the agent knows the effect model $P(e|\mathbf{a}$ where e are the observed effects and $\mathbf{a}$ is the joint action. The actions themselves are not observed, only the effects. Figure 3.2 shows the new Bayesian network. In this case, the agent's belief state contains the effects, rather than the actions, and the

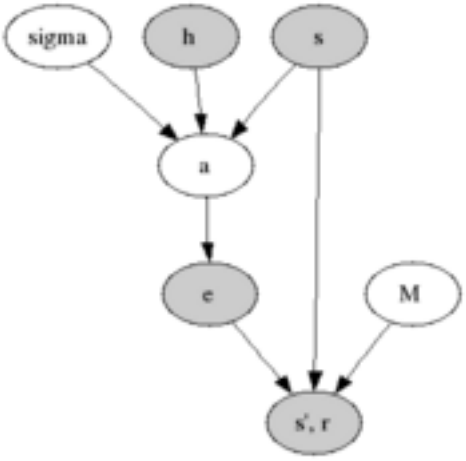


FIGURE 3.2: Bayesian network when action effects are observable

history trail contains a sequence of states and effects. Referring to the diagram, we obtain the following updates:

$$\begin{aligned} P(M|obs) &\propto \sum_{\mathbf{a},\sigma} P(M, e, s', r, \mathbf{a}, h, s, \sigma) \\ &\propto P(M)P(s', r|M, e, s, h) \end{aligned}$$

$$\begin{aligned} P(\sigma|obs) &\propto \sum_{\mathbf{a},M} P(M, o, s', r, \mathbf{a}, h, s, \sigma) \\ &= \sum_{\mathbf{a},M} P(s', r|M, e, s)P(e|\mathbf{a})P(\mathbf{a}|\sigma, h, s)P(M)P(\sigma) \\ &\propto P(\sigma)\sum_{\mathbf{a}} P(e|\mathbf{a})P(\mathbf{a}|\sigma, h, s) \end{aligned}$$

In this case, the environmental MDP and the strategies are independent. Indeed, it may seem that there is little difference between incorporating effect uncertainty into the environmental MDP and modelling it separately in this way. However, if the model $P(e|a)$ is known to the agent (for example, because all agents are of the same type), then it can be useful to explicitly include it in the model so that it can be quickly updated independently of other environmental conditions.

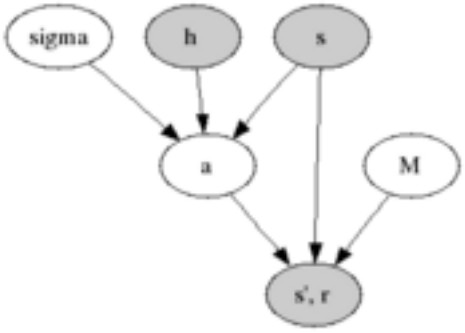


FIGURE 3.3: Bayesian network when joint actions are not observable

3.3.2 Limited visibility

Figure 3.3 shows the Bayesian network for this setting. It is identical to that of the fully observable case, except that the joint action is no longer an observed variable (for ease of reading, we have not included the agent's own choice of action separately in the diagram). The agent's belief state is modified to contain only the individual agent's action rather than an observed joint action. The history trail now consists only of a list of states.

Referring to the Bayesian network to justify the factorisation, we make the following modifications to the fully-observable model:

$$\begin{aligned} P(M|obs) &\propto \sum_{\mathbf{a},\sigma} P(M, s', r, \mathbf{a}, h, s, \sigma) \\ &= \sum_{\mathbf{a},\sigma} P(s', r|M, \mathbf{a}, s) P(\mathbf{a}|\sigma, h, s) P(M) P(\sigma) \\ &= P(M) \sum_{\mathbf{a}} P(s', r|M, \mathbf{a}, s) \int_{\sigma} P(\mathbf{a}|\sigma, h, s) P(\sigma) \end{aligned}$$

$$\begin{aligned} P(\sigma|obs) &\propto \sum_{\mathbf{a},M} P(M, s', r, \mathbf{a}, h, s, \sigma) \\ &= \sum_{\mathbf{a},M} P(s', r|M, \mathbf{a}, s) P(\mathbf{a}|\sigma, h, s) P(M) P(\sigma) \\ &= P(\sigma) \sum_{\mathbf{a}} P(\mathbf{a}|\sigma, h, s) \int_{M} P(s', r|M, \mathbf{a}, s) P(M) \end{aligned}$$

(Although the agent's belief state no longer contains a joint action, the previous definition for $Q(a_i, b)$, including the belief state with a complete joint action, is

still applicable, since the latter is only used inside a summation where the joint action has been defined).

Notice that unlike in the previous case, in the hidden variable model M and σ are not independent given the available observations. The order in which the two updates are performed may therefore affect the models. We will perform the two updates in parallel, so that each update at time $t+1$ uses the models from time t. At this point, we cannot make any guarantees about convergency.

This dependence also makes the problem harder to solve. We expect to use this formulation only if we know that the transition function can be broken up in such a way that even the prior can be used to make some inferences about state information. For example, the state might consist of several variables, of which one is the location of the agents. This location variable would permit inference about when an agent had chosen a move action.

3.4 Partially observable states

We now turn to the case where the underlying global state is not visible to the agent, only some local observations o. We assume that the model $P(o|s)$ is known to the agent. Furthermore, each state gives rise to a deterministic set of observations of which the agent will see some subset. This means that for each set of observations, there will be a known, fixed set of states which may have given rise to the observations. Each set of observations o is local to a particular agent, so the agent's strategy must depend on its own local observations. The belief state now contains, instead of the current state, the current set of observations and a distribution over state probabilities.

As discussed in section 2.2.3, although the underlying process is still assumed to be an MDP (i.e., the current state is dependent only on the previous state and the action choices), the sequence of observations is no longer Markov. If the environmental models were completely known, then we could define a probability distribution over states at each step, and the sequence of such distributions would itself be Markov. However, in our case the probability distributions are themselves being estimated. Consequently, the sequence of belief states is non-Markov (alternatively, each belief state should contain (and use in the updates), the complete history trail for the problem).

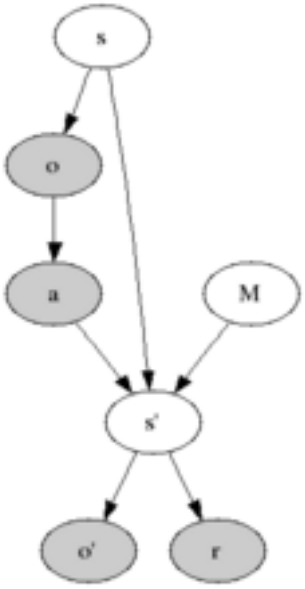


FIGURE 3.4: Bayesian network diagram for a single agent system with partially observable states

In the models below, we form a sequence of belief states as though the Markov property holds. However, further investigation is necessary to determine what the exact properties of these models are. We expect that to demonstrate any kind of convergence guarantee, it will be necessary at least to include some history trail in the belief state.

We begin by considering the single-agent case, a Bayesian treatment of learning in POMDPs, before continuing to the multi-agent model.

3.4.1 Single-agent case

In this section, we apply the Bayesian probability model to an ordinary POMDP, without explicitly considering the other agents. This extends the POMDP model with a Bayesian technique for learning the dynamics, providing an intermediate step between the models of Dearden et al. (1999) and our construction in the next section (3.4.2). Although the multi-agent construction in section 3.4.2 does follow directly from that of (Chalkiadakis and Boutilier, 2003), we feel that by supplying these two alternative progressions, it may be possible to understand the final construction more easily.

In more detail, figure 3.4 shows the Bayesian network diagram for the system. The new state is dependent on the observed action, the unknown previous state, and the unknown model.

There is only one model update to make,

$$\begin{aligned} P(M|obs) &\propto \sum_{s',s} P(o'|s')P(r|s')P(s'|a,s,M)P(M)P(a|o)P(o|s)P(s) \\ &\propto P(M)\sum_{s'} P(o'|s')P(r|s')\sum_{s} P(s'|a,s,M)P(o|s)P(s) \end{aligned}$$

This appears complicated, but is worth noting that it is slightly less complex than it may appear at first glance: for any observation o, only a subset of the possible states can give rise to the observation. For all the rest of the states, $P(o|s)$ is zero, and there is therefore no need to compute any of the other terms in the equation.

We turn now to the multi-agent case.

3.4.2 Multi-agent case

In the multi-agent case, the learning agent must retain in its own belief state beliefs about the belief state of the other agents—not other agents' beliefs about the models or strategies, but their beliefs about the current underlying state. These beliefs will depend on the sequence of observations made by those agents, which will be only partially known to us. We use the "history" variable to represent this, noting that each agent will have an independent history variable unknown to the other agents.

Figure 3.5 gives the Bayesian network diagram for a multi-agent system with partially observable states. In this diagram we have separated the observations and the actions of the agent under consideration (subscript i) and the other agents (subscript j). Tracking the network flow downwards, the root node is the original state, s, giving rise to a visible set of observations for the agent, and unknown sets of observations for the other players. We assume that the other players update their belief states given their observations and act accordingly. As previously, the actions, previous state and the system dynamics determine the next state, which emits a reward and which all agents make new observations from.

Referring to this diagram, we obtain the following updates (note that *obs* refers to all the observations made by our agent, including rewards and actions of other agents, as distinct from o_i the observations made from the state):

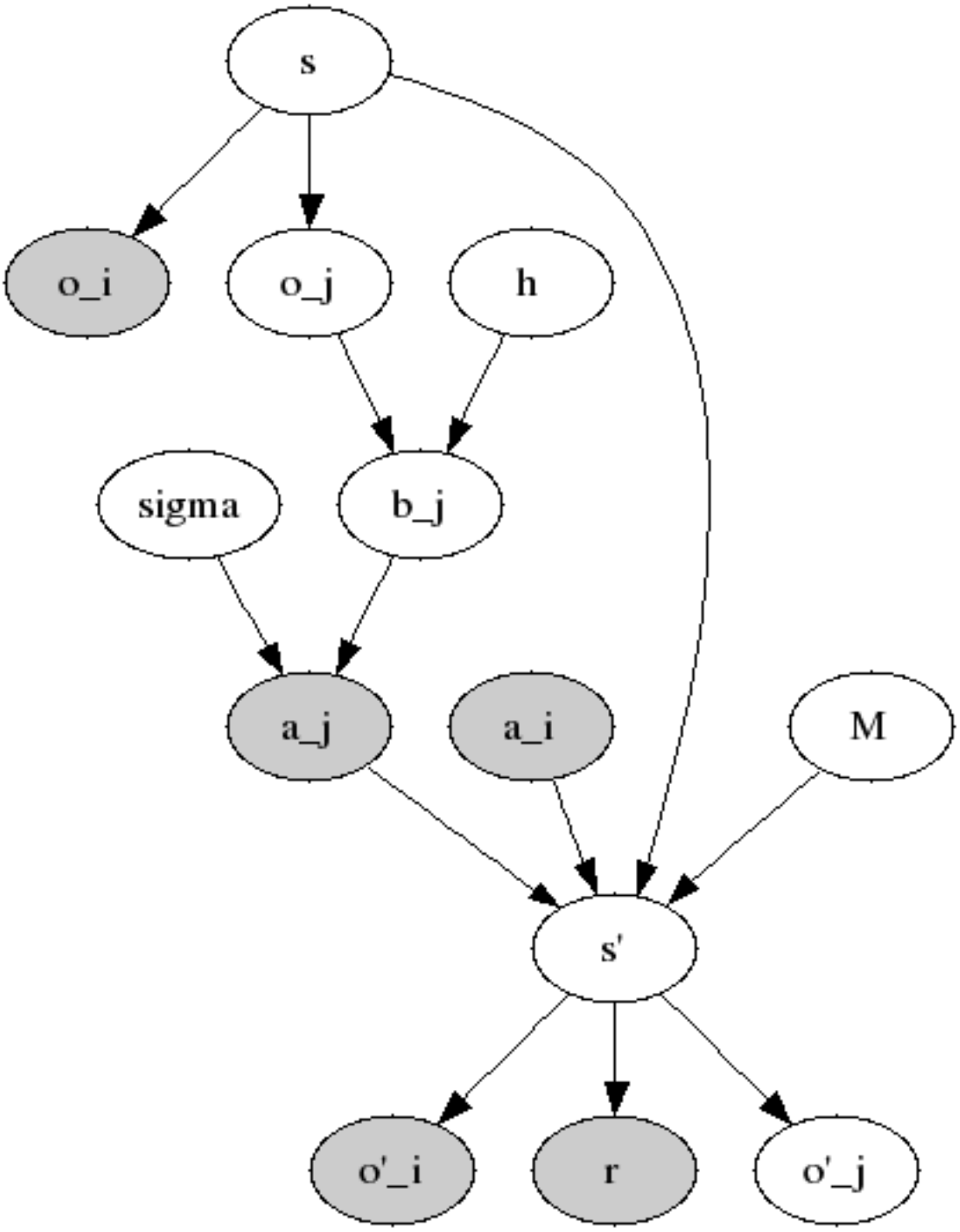


FIGURE 3.5: Bayesian network diagram for a system of partially observable states

$$
\begin{aligned}
P(M|obs) &\propto \sum_{s,s',\sigma,o_j,b_j,h} P(s,s',\sigma,o_j,b_j,h,o_i',o_i,r,\mathbf{a},M) \\
&= \sum_{s,s',\sigma,o_j,b_j,h} P(r,o_i'|s')P(s'|M,\mathbf{a},s)P(M)P(a_j|\sigma,b_j)P(b_j|o_j,h)P(o_j|s)P(s)P(\sigma) \\
&= P(M)*\ldots \\
&\quad \ldots \sum_{s',s} P(r,o_i'|s')P(s'|M,\mathbf{a},s)P(s)\sum_{o_j} P(o_j|s)\sum_{b_j,h} P(b_j|o_j,h)\int_{\sigma} P(a_j|\sigma,b_j)P(\sigma)
\end{aligned}
$$

$$
\begin{aligned}
P(\sigma|obs) &\propto \sum_{s,s',M,o_j,b_j,h} P(s,s',\sigma,o_j,b_j,h,o_i',o_i,r,\mathbf{a},M) \\
&= P(\sigma)*\ldots \\
&\quad \ldots \sum_{s,s'} P(o_i',r|s')P(s)\sum_{b_j,h} P(\mathbf{a}|\sigma,b_j)P(b_j|o_j,h)\sum_{o_j} P(o_j|s)\int_{M} P(s'|M,\mathbf{a},s)P(M)
\end{aligned}
$$

As before, the definition of $Q(a_i,b_i)$ remains unchanged. However, we now define

$$
\begin{aligned}
P(a_j|b) &= \sum_{\sigma_j,b_j} P(a_j|b_j,\sigma_j)P(b_j,\sigma_j|b) \\
&= \sum_{\sigma_j,b_j} P(a_j|b_j,\sigma_j)P(b_j|b)P(\sigma_j|b)
\end{aligned}
$$

$$
P(r,s'|\mathbf{a},b) = \int_{M} P(r,s'|o,\mathbf{a},M) \tag{3.5}
$$

where

$$
P(b_j|b) = \sum_{h,o_j} P(b_j|h,o_j)P(h|b)\sum_{s} P(o_j|s)P(s|b)
$$

and

$$
\begin{aligned}
P(r,s'|o,\mathbf{a},M) &= \sum_{s} P(r,s'|s,\mathbf{a},M)P(s|\mathbf{a},M) \\
&\propto \sum_{s} P(r,s'|s,\mathbf{a},M)\sum_{b} P(\mathbf{a}|b)P(b|s)P(s)
\end{aligned}
$$

(Note that the diagram in figure 3.5 shows the reward dependent on the initial state and action choice. However, the network (and equations) can be easily modified to make the reward dependent on the resulting state).

These new equations are more computationally complex than those defined for the case of partially observable actions, and impractical to evaluate on any realistic problem. Notice that $P(M|obs)$ is no longer independent of the strategy model, despite the action observability. This is because we are using the action choice to make inferences about the state, which then affect our estimate for M, making the model rather more involved. This Bayesian approach to multi-agent strategies is similar to the approach described by Emery-Montemerlo et al. (discussed in section 2.4). However, Emery-Montemerlo et. al only consider games with known dynamics. Despite this, several approximations are required to make computation feasible in our model. Thus for us also, several approximations are necessary. In the next section we identify some possible approximations.

3.5 Approximations

As discussed, the models above, in particular the last, cannot be practically solved.

We use a combination of techniques to render the system tractable. Firstly, we use the approximation techniques—sampling and myopic evaluation—described by Dearden et al. (section 2.2.1) and Chalkiadakis and Boutilier (section 2.2.3). Secondly, where possible we manually define a set of abstract states with corresponding abstract actions, and the mapping from states to abstract states and from abstract actions to actual actions. These approximations are sufficient to permit us to evaluate the partially observable action model on smaller examples. In our immediate work, we do not go beyond this. However, to evaluate this model in larger systems, or to begin to contemplate evaluation of the partially observable state model, more powerful approximation techniques are needed. We propose for future work that in addition to the previous approximations, some combination of the following may be practical:

1. The approximation of (Emery-Montemerlo et al., 2004) models POSGs as a series of Bayesian games: that is, fully observable systems in which agents may have private information. Investigation of how this model would extend into a learning system appears a natural step.

2. The principal components analysis-based technique of Roy and Gordon (section 2.4.1) provides a principled way to reduce large or continuous state spaces.

3. Hierarchical models, as described in section 2.4.1, are natural to large and complex systems, permitting several layers of abstraction.

4. In any large state space described by a series of variables, known indepdences between variables can be exploited to simplify the model (e.g. Hoey (2001) describes a way to do this within a hierarchical model, for a single agent system. We do not anticipate any particular difficulty in extending the model to a multi-agent system).

3.6 Summary

Firstly, we have defined Bayesian models for multi-agent reinforcement learning in for two different cases of partially observable actions. In order to render the latter model computationally tractable, we have proposed a sequence of approximation techniques. Secondly, we have described the Bayesian model for reinforcement learning in a partially observable multi-agent system. This model is computationally intractable, and we have outlined some possible approximation techniques which might permit solving such models. In the next chapter we evaluate the effectiveness of the former model on an example problem and sketch an instantiation of the latter.

Chapter 4

Demonstration and evaluation of the Bayesian model

In chapter 3 we proposed Bayesian exploration models for two cases of partial observability in an uncertain system. Firstly, we considered a system in which the actions of other the agents cannot be fully observed, but where some inferences about the actions can be made. Secondly, we considered a system in which the state itself cannot be fully observed. In this chapter, we evaluate a demonstration of the first model. At the end of the chapter we outline a potential instantiation of the second model and discuss some of the issues involved in its implementation, although the implementation itself is left to future work.

4.1 Partially observable actions

The first case of partially observable actions which we discussed (section 3.3.1) was that in which the effects are observable but the underlying strategy is not. In a static scenario, this is equivalent to the fully-observable multi-agent problem with a factorised transition pdf. We have proposed scenarios in which explicitly separating knowledge about strategies from knowledge about effects may be beneficial (for example 2.6 in section 2.3.2) . However, there is nothing new to test beyond factorised transition functions, so we do not discuss the implementation of these scenarios further in this section.

However, many disaster scenarios have the property of limited visibility of other agents, while coordinating with them. Consider, for example, a simple scene where

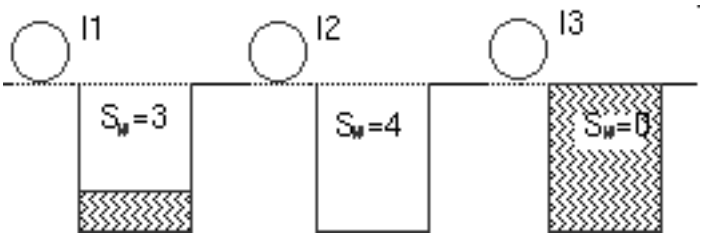


FIGURE 4.1: An illustrative state in the well problem

a rescue worker is digging at a pile of rubble to try and find trapped victims, while other rescue workers dig at the same pile of rubble from around the other side. We use an abstract problem inspired by this kind of scene to test the partially observable action model.

In more detail, we consider the following simple coordination problem:

- Each agent is digging a "well"
- At each turn, each agent may "dig" or "fill" its well
- Each time all the wells are at the same depth, reward is emitted
- For every well which is at the maximum or minimum depth, there is a penalty

Figure 4.1 illustrates a sample state within this problem. In the state depicted, two of the wells (those associated with I2 and I3) obtain the penalty. No reward is achieved. The total reward is therefore negative: 2**penalty*. In more detail, this model contains :

- A set of agents. We begin by considering the problem with just two agents, $I = \{I_1, I_2\}$
- A state space describing for each well the depth of the well. We suppose that there may be five different depths to each well i (where well i is associated with agent I_i): $d_i = \{0, 1, 2, 3, 4\}$. For two agents, this gives rise to $5^2 = 25$ distinct states.
- A set of actions available to the agents. In this case, there are two actions, "dig" or "fill". These are expected to alter the depth of the well by one level, except that "dig" in a maximum-depth well or "fill" in a zero-depth well will be no-ops. We use $d' = Act(d, a)$ to refer to this deterministic function, where d and d' refer to the old and new depths of the well respectively, and a is the action performed by the agent associated with the well.

- A deterministic reward function. The total reward r is computed by $r = r_s + \sum_i r_{i,d}$, where $r_s = 10$ if all depths are the same, otherwise 0, while $r_{i,d} = -5$ if agent i is at depth 0 or depth 4, otherwise 0.

- A transition pdf for each well, describing the possibilities of wall-collapse (decreasing the depth) or sinkage (increasing the depth) for each well. We write this as $P_t(d_i'|d_i)$, where i identifies the well. The collapse and the action are assumed to take place at the same time. The overall depth change is determined by summing the effect of the action and the effect of the unprovoked movement: $P(d_{i,t+1}|d_{i,t}, a_i) = P_t(d_{i,t+1}|Act(d_{i,t}, a_i))$

  Initially, we consider a transition pdf in which the probability of unprovoked movement is the same for each well and the same for either sinkage or collapse. We vary this probability between 0.5 and 0. We use the value 'delta' to refer to the probability of *no* unprovoked movement, varying delta between 0.5 and 1.

We implemented this problem along with with several sets of agents functioning within the system, in order to compare learning agents using our model of section 3.3 with other agents. In the next section we discuss our experiments on this problem in detail.

4.1.1 Experiments

We compared our model with two learners: a standard single-agent Q-learner (run by all the agents, using varying learning rates to avoid cycling) and a multi-agent learner based on the model of Chalkiadakis and Boutilier described in section 3.2, which is able to observe all the actions.

In order to make the system more computationally tractable, we made a further simplification to the model of section 3.3.2: rather than sum probabilities over all possible actions, we estimate the maximum likelihood action given the current belief state, and perform updates as though this action had been carried out. More precisely, instead of computing (equation 3.5 in section 3.3.2):

$$P(M|obs) = \sum_{\mathbf{a}} P(s', r|M, \mathbf{a}, s) \int_{\sigma} P(\mathbf{a}|\sigma, h, s) P(\sigma)$$

we compute:

$$\begin{aligned} \mathbf{a_{ML}} &= \max_{\mathbf{a}} P(\mathbf{a}|s, s', b) \\ &= \max_{\mathbf{a}} P(s'|\mathbf{a}, s, b) P(\mathbf{a}|s, b) \\ P(M|obs) &= P(s', r|M, \mathbf{a_{ML}}, s) \int_{\sigma} P(\mathbf{a_{ML}}|\sigma, h, s) P(\sigma) \end{aligned}$$

As well as reducing the depth of summation necessary, this approximation allows us to use the Dirichlet priors and posteriors as described in (Chalkiadakis and Boutilier, 2003).

Despite this simplification, we found the system to be unfeasibly slow to run with more than two agents (taking several days to run a single experiment of three runs, on a desktop PC) or when using many samples (above 100 samples inside the double integral in the best response computation) to estimate integrals. This could be mitigated in the future by using a more efficient language for implementation (we used MATLAB for convenience); by using the sparse priors described in (Dearden et al., 1999), and ultimately by using a set of approximations as outlined in section 3.5.

Subject to these resource constraints, we also paid attention to the following parameters:

Number of agents We believe that the effects of using our explicitly multi-agent model over a single agent model should become more marked as the number of agents increases. However, for this problem the number of states, hence the computational complexity, increases exponentially with the number of agents, making it infeasible for us to test on many agents. In fact, we limited our experiments to just two agents, but we believe that future work should begin by making the system more computationally efficient and trialling it with up to five agents.

Number of steps Of interest to us is not just whether the system manages to settle to a good stable strategy, but how quickly this occurs. We found that for the two agent 300 steps was typically sufficient for the system to stabilise. Once again due to resource constraints, most of our tests were limited to 120 steps, which we found sufficient to get past initialisation effect and compare the behaviour of the algorithms.

Delta We investigated different values for delta between 0.5 and 1.0. When varying the sample size, we fixed delta at 0.65, as a middle ground with some randomness but not so much that nothing can be learned.

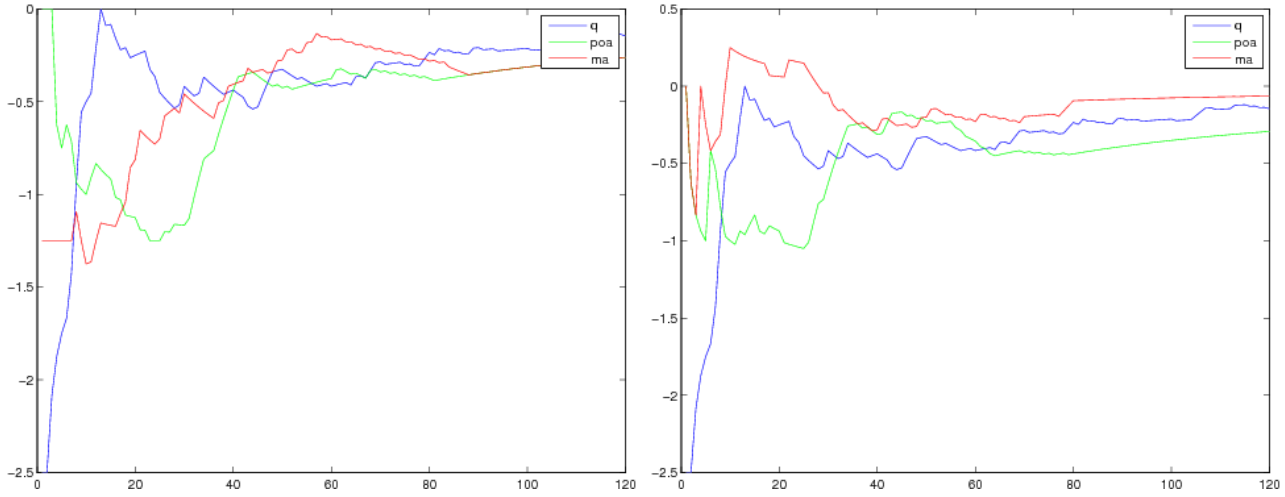
Sample size In our partially observable action model, it is necessary to estimate a double integral (over all models for the dynamics and all models for the strategies). We chose to do this by taking *sample_size* samples from possible dynamics and *sample_size* samples from possible strategies, and evaluating best responses over all possible combinations of these, so that for a chosen sample size we in fact evaluated $sample\_size^2$ models. For the fully observable comparison, we used the same sample size as the current partially observable tests. We experimented with sample sizes of 7, 10 and 15. However, the experiments with sample size 15 never completed.

Other parameters For the q-learner, it is necessary to set a learning rate. After experimenting to find good configurations for both two and three agents, we chose a base learning rate α of 0.2, and set the learning rate for each agent to be $\alpha_i = \alpha * 0.1^{(i-1)}$. We used ϵ-greedy action selection (as described in section 1.2) with a fixed ϵ of 0.1 (again, after experimenting with several fixed and varying values). Finally, we set γ, the agents' myopia to be 0.75. This means that they allocate only a quarter of the original importance to states five steps in the future, and by ten steps the contribution of new states is negligible. We expect this to reflect this digging problem with a small number of states.

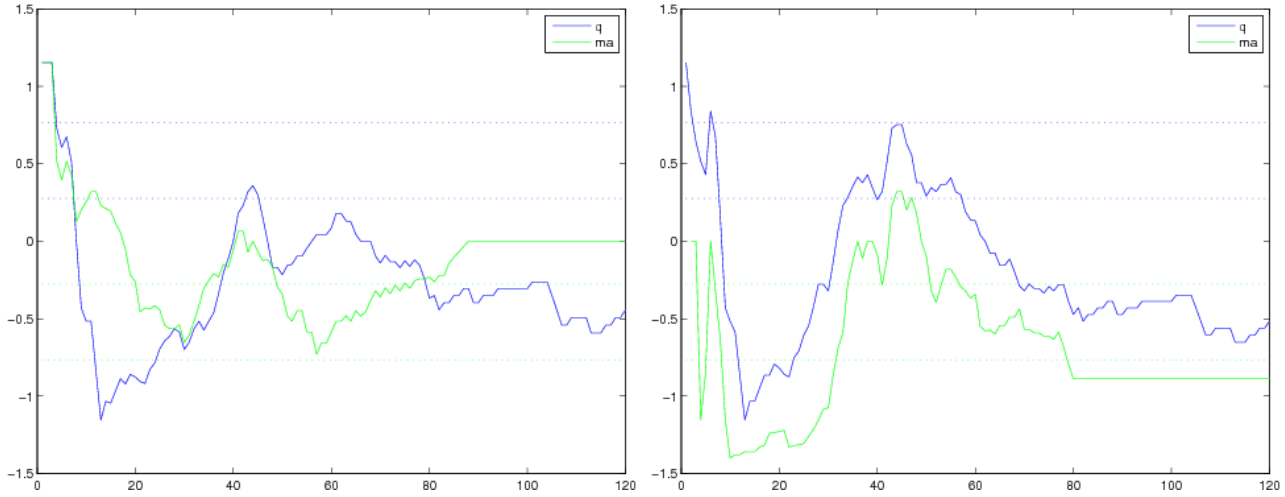
Number of runs For each setting (sample size and delta), we ran four tests, averaging the results over these tests. In the next section we also discuss the variation in the results and show the t-tests for them.

4.1.2 Results

A complete set of results is supplied in the appendix (A). Here, we discuss a selection of the more interesting results. For each of the reward graphs, the average total discounted reward has been computed for each run, and the mean of these over the set of runs is shown. The errorbar charts show the standard deviation over the set of runs. Finally, we performed t-tests on the outputs as follows: for each time step a t-test was carried out comparing our partially-observable model with the q-learner and separately with the fully-observable model. The results of these tests were then plotted, and the relevant values for 60% confidence and 75%



(a) Agent scores: Delta=0.65. Sample size 7^2 (left), 10^2 (right)



(b) t-test results: Delta=0.65. Sample size 7^2 (left), 10^2 (right)

FIGURE 4.2: Agent scores and t-tests for two different sample sizes

confidence have been dotted onto the graphs. These are unusually low confidence rates for determining statistical significance. However, as we will discuss within the following sections, the low sampling rates and small number of runs mean that most of our experiments fall within these confidences.

In all the figures included, 'q' refers to the q-learner, 'ma' to the fully observable multi-agent learner, and 'poa' to the partially observable action learner.

Effect of sample size Figure 4.2 shows the scores and the t-test results for sample sizes of 7 and 10. Although the q-learner behaves the same in each case, the two multi-agent learners both perform better with the larger sample size, as is to be expected from the greater accuracy in evaluating the best response at

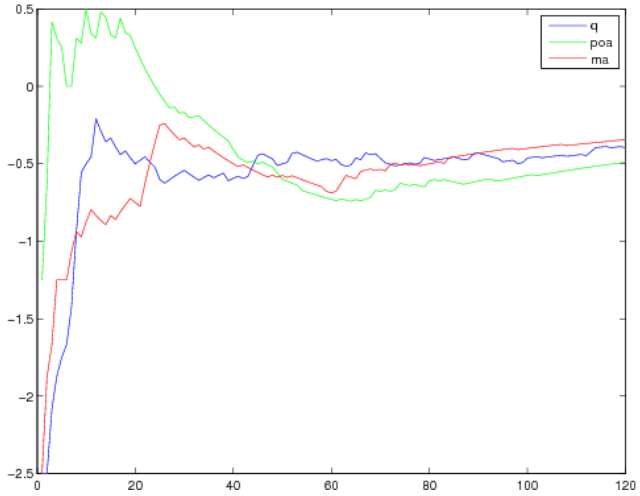
each step. From the t-tests below we can also see that the performance is less variable with the larger sample size; the results for the same number of runs are more significant. Again this is a consequence of the increased accuracy of the evaluation of the best response.

In this case, our learner does not perform as well as the q-learner although both are still improving at the end of the run. However, as our learner has improved with the larger number of samples, we would expect this trend to continue as improve again with a still larger number. More investigation would demonstrate how many samples might profitably be used before the improvements ceased to be sufficient to justify the computational cost. Ultimately, rather than having a fixed sample rate, we might at each step compute the confidence interval for the estimated Q-values, and sample until the confidence reached some predefined threshold. We could try and reduce the number of samples necessary by using techniques such as importance sampling (described in (Chalkiadakis and Boutilier, 2003) and (Dearden et al., 1999)).

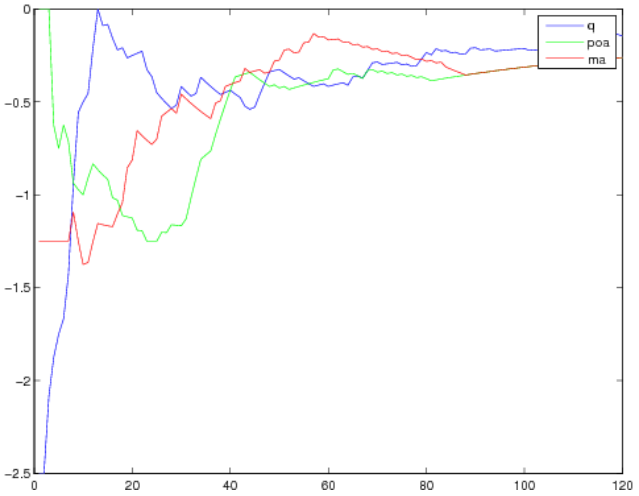
Effect of delta Fixing the sample size at 7, we next ran tests with different values of delta. Figures 4.3 and 4.4 show the agent scores and the t-tests for these experiments respectively. From figure 4.4 we can see that as the delta value was increased, making the system more deterministic, the differences between algorithms become correspondingly more significant. In particular, the q-learner's performance improves rapidly as the system becomes more deterministic. By comparison, the multi-agent learners are slightly less susceptible to randomness in the system than the q-learner. This is because they separate randomness due to the other agents from randomness due to the delta parameter, and consequently have to learn several relatively simple functions rather than one complex (combined) one.

However, their performance is consistently less good rather than consistently better. As discussed, higher sampling rates would improve the performance.

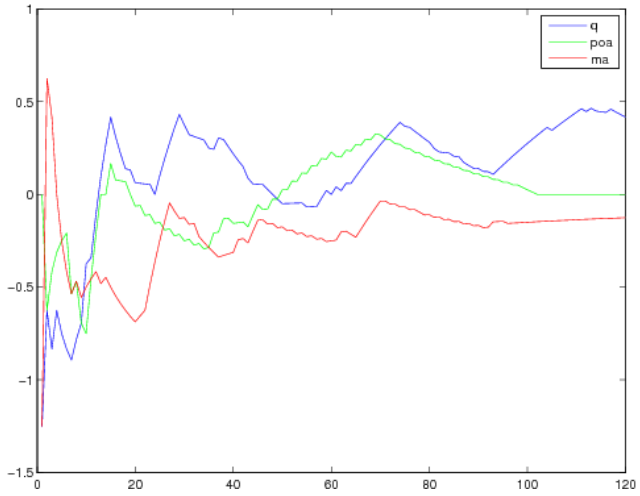
Changing delta Figure 4.5 shows the mean of three longer runs, using a sample size of 10 in which the delta value was altered from 0.81 to 0.5 after 100 time steps. This represents a scenario where the environmental conditions change unexpectedly—for example, in the rescue plane scenario of section 1.4, the wind might change rapidly. In a digging scenario, perhaps there has been some earthquake or a rainstorm loosening the earth. From the figure we can see the q-learner's



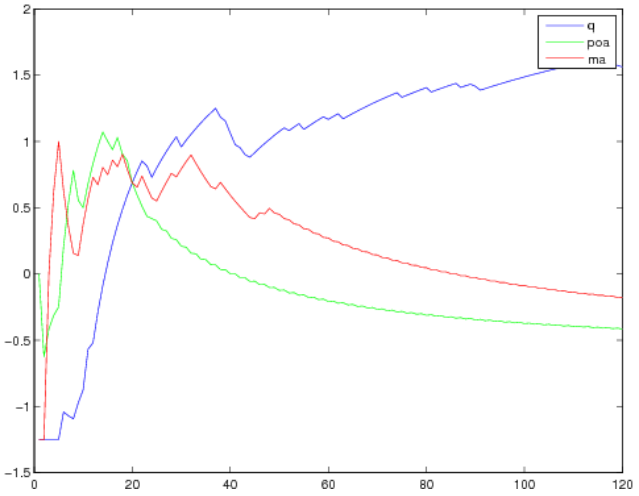
(a) Sample size 7^2; delta=0.5



(b) Sample size 7^2; delta=0.65



(c) Sample size 7^2; delta=0.8

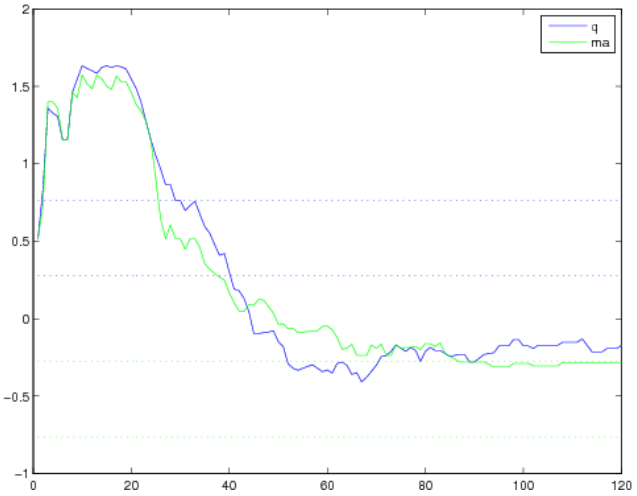


(d) Sample size 7^2; delta=1.0

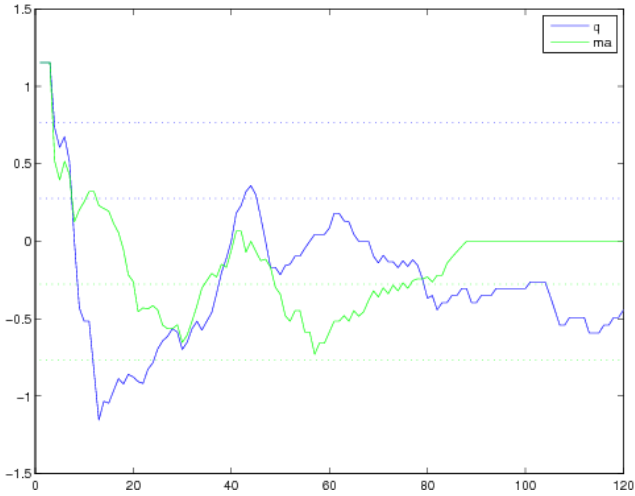
FIGURE 4.3: Varying delta: agent scores

scores promptly droppping where the change is made, only slowly beginning to climb again after many timesteps. By contrast, neither the multi-agent learner nor the partially-observable action learner appear to suffer, again because they are learning the transition function separately from the agent functions and are so able to quickly identify changes in the environment (which are distinct from changes in agent strategy and in this problem should not cause a change in strategy).

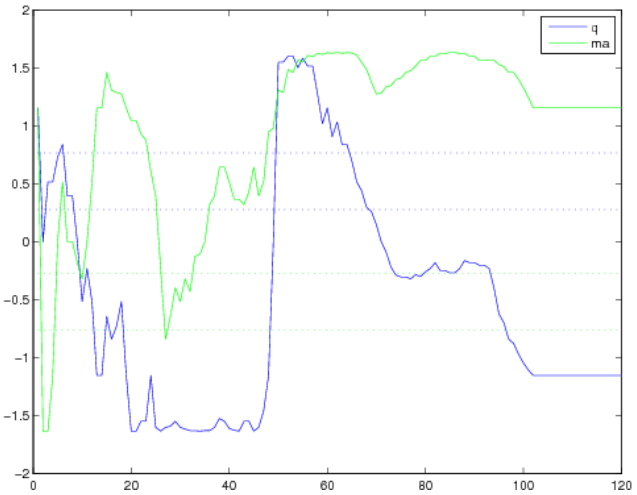
Error bar shape Characteristic of all the error graphs is a high deviation early on, followed by a momentary reduction to negligible deviation and then a return to a slowly reducing high deviation. Figure 4.6 shows two examples. The



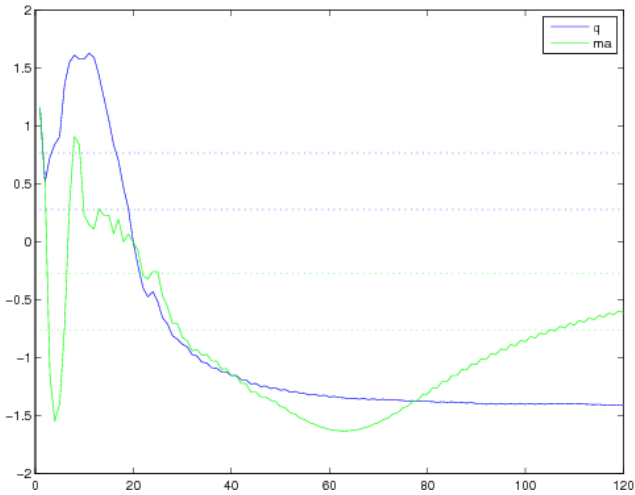
(a) Sample size 7^2; delta=0.5



(b) Sample size 7^2; delta=0.65



(c) Sample size 7^2; delta=0.8



(d) Sample size 7^2; delta=1.0

FIGURE 4.4: Varying delta: t-test results

reduction to negligible deviation is unexpected and the reason for it is not clear. Furthermore, in the q-learner there can often be seen what appear to be cycles of increasing and decreasing deviation (figure 4.7). Since these graphs are only measuring deviation over a small number of runs, we propose that the behaviour should be more fully investigated with larger numbers of runs and higher sampling rates for the multi-agent algorithms.

Summary The above graphs demonstrate that even with low sample rates, our agent is able to perform close to the q-learner on this simple problem. However, we also note that in all cases the scores are tending to be around 0, indicating that the agents in each case are successfully keeping away from the penalties at the

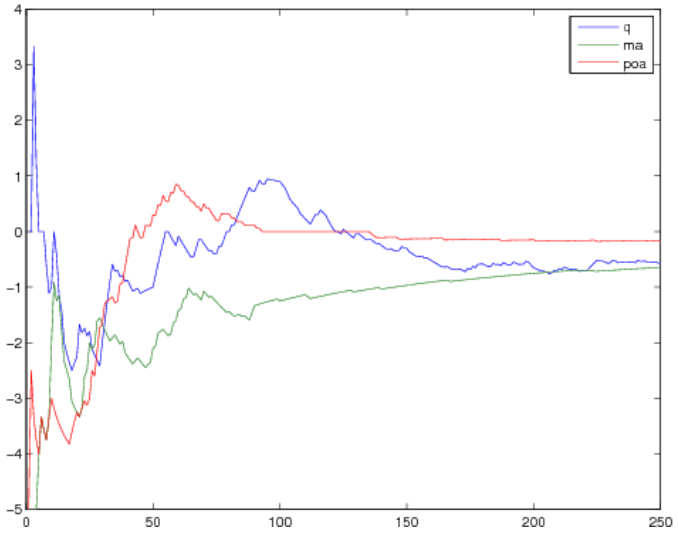
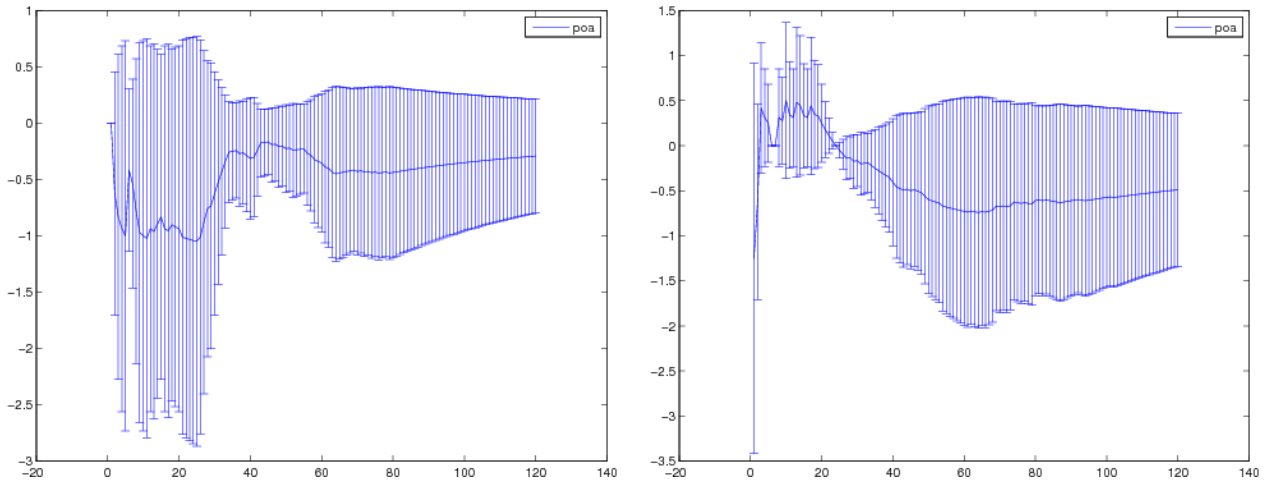


FIGURE 4.5: Effect of changing delta from 0.81 to 0.5 after 100 time steps (sample size: 10)



(a) Sample size 10^2; delta=0.5. partially-observable algorithm (b) Sample size 7^2; delta=0.5. partially-observable algorithm

FIGURE 4.6: Example error graphs

edge, but not necessarily coordinating effectively. A simple handwritten strategy for this problem which selects "fill" if the agent's well is deeper than 3, "dig" if the agent's well is shallower than 3, and randomly when the well is at depth 3, achieves average scores of around 1 for the two-agent problem. So there is room for improvement with all the learners.

From the above we also observe that our agent appears to be less susceptible than the q-learner to the variations in transition function. We expect this effect to become more marked when there are larger numbers of agents, and propose to investigate this further when a more efficient implementation of the system is

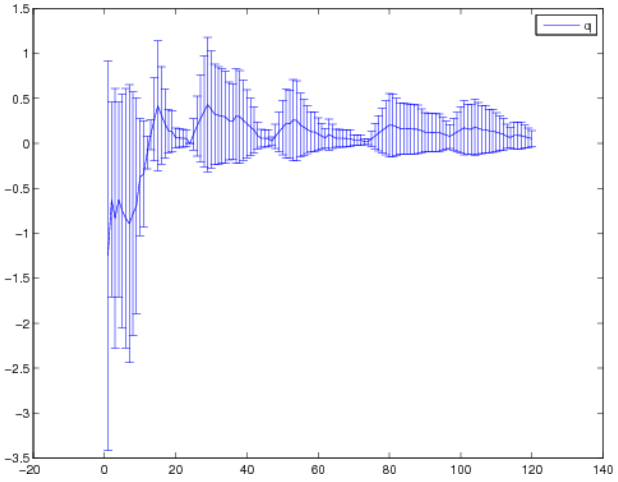


FIGURE 4.7: Sample size 10^2; delta=0.8 $\rightarrow$ 0.5. q-learner

available. Finally, there is a noticeable improvement in the behaviour of our agents as the number of samples is increased and once again we propose to investigate this effect further with a more efficient implementation.

4.2 Partially observable states: Example from the disaster response domain

In this section, we instantiate the model of section 3.4 on a larger coordination problem in which the agents are only able to observe part of the current state. We suppose that there has been some disaster such as an earthquake and a team of ambulances is deployed to find and rescue victims who have been trapped under the debris. However, initially the locations of the victims are not known to the ambulances and they have a restricted range of vision and hearing as they travel around the region. Furthermore, further shock waves may result in more victims being buried. This problem is inspired by the ambulance sub-problem of Robocup Rescue[1].

In more detail, we propose the following grid world of victims and ambulances:

State space We use a factorized state space containing the following variables:

- Number of victims n_v : between 0 and $2k$ where k is the number of grid squares

[1] `http://www.rescuesystem.org/robocuprescue/`

- Location (grid square) of each victim l_v
- Buried depth of each victim d_v : an integer value between 1 and 100
- Number of ambulances n_a : between 1 and l where l is the length of the longest side of the grid
- Location (grid square) of each ambulance l_a
- It is assumed that ambulances cannot be buried.

Initial state We initialise the scenario as follows:

- A grid is defined with length g_l and width g_w, hence $k = g_l * g_w$ grid squares.
- n_v is selected from a Dirichlet distribution peaking around $\sqrt{(k)}$.
- n_v victims are distributed uniformly across the grid (index all the grid squares and select (with replacement) a grid square from a uniform distribution).
- Each victim is assigned a depth, selecting integers from a uniform distribution between 1 and 100.
- n_a is defined between 1 and $\max(g_l, g_w)$.
- n_a ambulances are distributed uniformly across the grid.

Initial information From the outset, every ambulance knows:

- The structure of the world (length and width of the grid)
- Its own position in the grid
- The number and location of other ambulances

Observations At each turn the agent makes some action and is able to observe:

- The locations and depths of any civilians in its current square, to its immediate left or right, above or below it.
- The locations (after acting) of every other ambulance.

Therefore, for any given state, where the state includes the locations of the agents, the observations available to each agent can be exactly determined. (Conversely, for a particular set of observations, there may be many consistent states, as the observations will not necessarily cover all states).

Actions The actions available to the agents are:

- Move (Left, Right, Up, Down)
- Dig

Transitions The states and actions give rise to a probabilistic transition function as follows:

- Move actions: the requested move is performed unless it would take the agent off the grid, in which case no action is performed. **Traffic:** We assume infinite road space. However, a possible adjustment would be to assume that two agents cannot occupy the same grid square unless there is a buried victim there, so that if two agents aim to move to the same location only one of them will succeed.
- Dig actions: for each dig action at a location where a victim is buried, the buried depth of the civilian is reduced by five levels until it reaches zero, whereupon the victim ceases to be a victim and is no longer of interest to the ambulances. If more than one ambulance is digging at a particular locatiom a victim at that location can be released more quickly than if one ambulance were digging alone.
- Every turn: each victim's depth is altered by a random amount selected from a Gaussian with a mean of five levels (and then discretized).

Reward Finally, each ambulance is able to observe the score at each step which is determined by the number of victims who have been rescued (brought to depth 0), N, with deterministic reward $r = N^n$ for some small n. It will be necessary to experiment to determine a suitable value for n which efficiently enables the agent to maximise the score over time.

We have implemented such a world in MATLAB, alongside a simple handwritten strategy. In order to test our model on it, we must define an agent's belief state, with prior beliefs and update rules for the belief state. We do this below.

4.2.1 Applying the model of section 3.4

We define an agent's belief state, and then specify the prior beliefs and update rules for each part of the belief state.

Belief state In this world, the ambulance's belief state holds:

1. Its knowledge about the locations of other agents
2. Its beliefs about the number of victims
3. Its beliefs about the distribution and depths of victims across the grid (the above three items define its belief about the state)
4. Its beliefs about the consequences of moving or digging
5. Its beliefs about the unprovoked transitions of victims (this combined with the previous defines its belief about the transition function)
6. Its beliefs about the beliefs of the other ambulances: this refers only to ambulance beliefs about the distribution and depths of civilians, and not other agent strategies.
7. Its beliefs about the strategies of the other ambulances

Prior beliefs We initialise the agents with the belief that the number of victims lies between 0 and $2k$ where k is the total number of grid squares. We can describe this belief using a Dirichlet distribution. The posterior is then a multinomial describing the likelihood that the number of victims is v, or equivalently the number of victims per square is v/k. We propose to use a prior which peaks at around $\sqrt(k)$ (i.e., if the grid were square there would be around one victim per row), with a longer tail towards the $2k$ end.

The agent's initial beliefs about the distributions of the victims depend on its belief about the number of victims, with a uniform initial prior assigning equal likelihood of n/k victims to each grid square, where n is the mean of the Dirichlet distribution estimating the number of victims and k is still the total number of grid squares. That is, the victims' locations are selected (independent and identically distributed) from a uniform distribution. The joint probability distribution for the locations of all victims is the product of the independent distributions for each victim. Then computing the probability of n_s victims being in a particular square s is achieved by summing the probabilities of all joint allocations which place n_s victims in that square.

As long as moving and digging are deterministic and there are no traffic rules, the beliefs in 4 are straightforward: it expects move actions to affect the location

of the moving agent as requested, except where the action will move the agent off the grid, and digging actions to reduce the buried depth of the agent by five levels. The agent assumes a (discretized) Gaussian for unprovoked transitions, with mean μ and variance σ. The mean may be assumed to have a Gaussian distribution, initially with mean, variance of 0 and 1, and the variance to have a gamma distribution initially with shape parameter of 0.5 and scale parameter of 1.0.

All the probability belonging to invalid transitions (to depths of less than 0 or greater than 100) is assigned to the nearest valid depth.

Finally, we consider the beliefs about the strategies of the other ambulances. In this problem, the other ambulances are assumed to be cooperative since they all aim to maximise the same shared score. However, the strategies of the other agents are unknown; in particular they are not known to be rational. Therefore, we do not try and initialise ambulances with any particular bias concerning strategies, assuming uniform priors (i.e. any action is equally likely, regardless of state).

Belief updates These beliefs are updated at each time step based on the observations, using the Bayesian update rules as described in section 3.4:

When a previously unvisited square is visited In this problem, we assume a uniform distribution of victims across the map. We can therefore assume that the distribution of victims across observed squares will provide information about the distribution of vicims across the unvisited squares. Gradually the strength of information from observed squares will outweigh the prior information.

We can update first the estimated number of victims:

$$P(n_v = n|obs) \propto P(obs|n_v = n)P(n_v = n)$$

where *obs* is the mean number of victims observed per observed square. We can then update the likelihood of seeing a victim in an unobserved square using the uniform distribution for location of victims as described in the *prior beliefs* section, but using the new distribution for estimating the number of victims.

Future adaptations to the problem might suppose that victims are likely to be found in clusters (for example, all the victims which were in a building

which has collapsed will be close together). This update must then be adjusted accordingly, assigning more probability to unobserved squares close to observed victims. The density of victims in unobserved squares will no longer necessarily match the density in observed squares. Indeed, even if we begin with the assumption of a uniform spread of victims, we might after some exploration form beliefs about the spread of victims.

When a known victim is observed to change depth As long as digs are deterministic, effects due to agent actions can be deterministically observed. The agent can therefore precisely identify the effect due to unprovoked fluctuation and update its estimate of the random fluctuation function, assumed to be a Gaussian with mean μ and variance s.

$$P(\mu, s|obs) = P(obs|\mu, s)P(\mu, s)$$

Updating the distribution $P(\mu, s)$ is not straightforward, because the parameters μ and s are not independent and the joint distribution does not have a simple form. We propose therefore that we can estimate the mean using the technique described in (David et al., 2007): discretize the possible values for (μ, s) (μ has a known maximum of 99; we must estimate a maximum for s), and then assign $P(\mu, s|obs)$ numerically.

Note further that the depths of all observed victims are not observed at every time step. Rather the random fluctuation function must be iterated over the number of steps since a victim was last observed in order to estimate the current depth of a victim.

Finally, we assume that this function is not depth or location dependent, but future adaptations of the function may have different fluctuation pdfs for each depth and location. Assuming some continuity in these, we might use some form of function approximation such as a neural network to learn them, rather than trying to learn by visiting every depth and location individually.

When an agent is observed to dig at location l or move between locations We must update our beliefs about the agent's strategy σ, based on an estimate of its belief state. For agent i observing agent j performing action act_j and making state observations obs_i:

$$P(\sigma_j|act_j, obs_i, b_i) \propto \sum_{b_j} P(act_j|\sigma_j, b_j)P(b_j|obs_i, b_i)$$

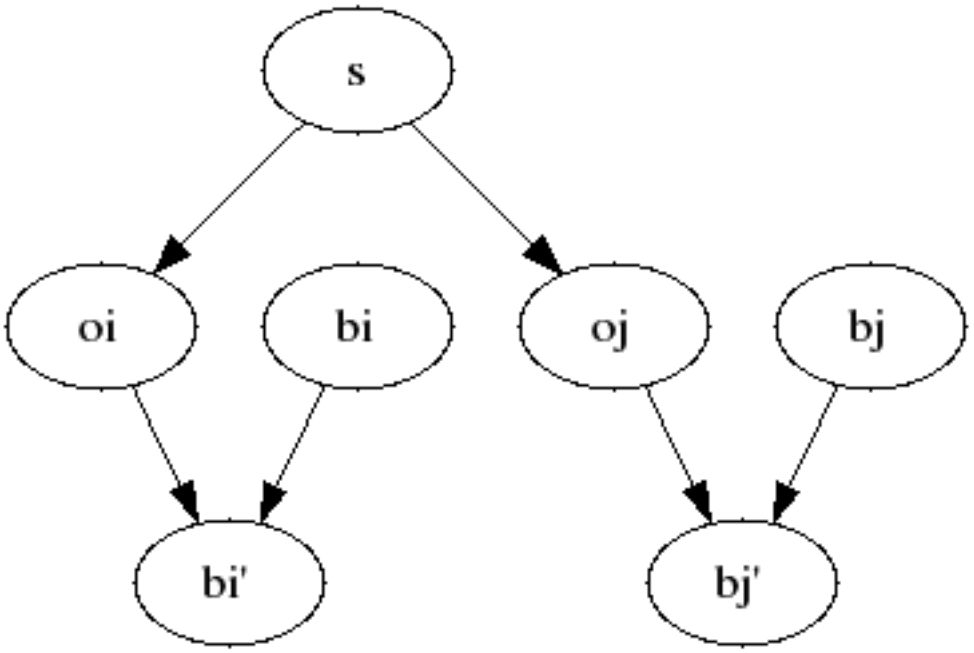


FIGURE 4.8: Bayesian diagram for belief states

where b_i and b_j refer to the belief states of agents i and j respectively.

We may then update our own belief state based on our beliefs about the agent's strategy. For simplicity, we do not propose to do this update initially. By not using this available information we will converge more slowly than we would have otherwise, but will not become "more wrong".

Finally, we can update our beliefs about the other agents' belief state (beliefs about the state, but not about the models). These must be maintained in order to be used above. We assume that other agents update their beliefs about the state based on their observations using the same updates we have just described for this agent. Referring to the Bayesian diagram in figure 4.8 (as discussed in section 3.4):

$$P(b'_j|o_i) = \sum_s P(s|o_i) \sum_{o_j} P(b_j|o_j)P(o_j|s)$$

Since the observations for every agent are known if the state is known, $P(o_j|s)$ is 1 for exactly one set of observations, and 0 otherwise. Writing $o_j = O(s, j)$ for the true set of observations allows us to simplify the update to

$$P(b'_j|o_i) = \sum_s P(s|o_i)P(b_j|O(s, j))$$

The order of the previous updates may affect the behaviour of the system. We choose to assume that they are all performed simultaneously, consistently using the beliefs from the previous step.

4.2.2 Practical considerations

Like the small well problem, this problem has a score function which is deterministic given the resulting state. Also like the small well problem, the transition function in this system is factorised into several variables which have some independences among them, with some of the variable updates being deterministic. However, the partial state observability in this problem makes it significantly more complex than the well problem, even with a fairly trivial transition function, primarily because we try to make inferences about the distribution of victims based on the behaviour of other agents, while simultaneously learning about that behaviour.

Furthermore, the problem as stated provides more information than might be realistic; in practice the movements of agents are likely to be observable only within the same limits as victims are observable—if all agent actions are observable, then this may be because the agents are in radio communication with each other and they may convey information about their local observations. The scenario could be further complicated by the conveyance of partial of false information.

Finally, we note that although we have supplied a mostly-uniform set of priors, we would expect that disaster response teams would collect information up from many different disasters, and use a meta-learning technique to learn appropriate priors both for the scenario and to learn about the behaviour of other agents.

4.3 Discussion

The initial results in section 4.1 serve as no more than a proof of concept, and there is a great deal more investigation to be done in order to fully exploit our new models. In the above work, we used low sampling rates, did not supply strong priors, and did not investigate with transporting agents into new environments, or transporting new agents into the same environment, which we have argued are key applications for this model. The effects of the various approximations should be more carefully investigated, and more sophisticated sampling methods could

be used. However, from the preliminary results we have shown that our model can perform almost as effectively as a fully-observable model, at least in this toy example.

For the more challenging scenario of partially observable states, we have outlined a potential example problem and shown how this example problem maps to the model described in section 3.4. However, this system is challenging to implement and would be computationally infeasible without use of a good set of approximations. In the next section we propose a schedule for extending the current experiments, defining such approximations and further testing and developing our model.

Chapter 5

Conclusions

5.1 Summary

Over the preceding sections we have highlighted the problem of coordination in uncertain multi-agent systems, and motivated the use of Bayesian learning models as a possible approach to this problem. We proposed using abstraction techniques from more traditional reinforcement learning to make such solutions scalable. We then described a particular Bayesian model capable of handling systems in which actions are partially observable, and demonstrated our model working on some test cases. Although we have shown that the model is usable for these cases and in some kinds of scenario is competitive with alternative techniques, these results are very prelimary and there is scope for a good deal more work in this area. In the following sections we identify several specific areas for this future work (section 5.2) and propose a timeline for the work (section 5.3)

5.2 Future work

In section 1.1 we defined several key properties of the disaster response domain, our focus domain. We took uncertainty as a key motivator for developing learning based models which explicitly take uncertainty into account. We proposed that by explicitly modelling other agents, we would pave the way for better adaptation to heterogenous, open systems and possibly dynamic systems. Our schedule for future work proposes to investigate more thoroughly the behaviour of our model in such heterogeneous, dynamic systems and its suitability for large-scale scenarios.

In more detail, we propose the following areas for investigation:

Properties In section 2.3.1 we described several guarantees which may be desirable in multi-agent learning algorithms: convergence, no-regret and rationality. As we have extended our model into more complex state spaces, we have not investigated what properties hold, or under what conditions these desirable properties may hold. Such an investigation should include the implications of the various approximations we have used.

Dynamic systems Another key requirement in section 1.1 was that the system should be usable in dynamic scenarios. Many reinforcement learning systems can be effectively applied to dynamic scenarios, adjusting the discount factor and the learning rate to give more weight to recent findings. This applies also to the model of Chalkiadakis and Boutilier (2003), with a dynamic environment, or if the strategies of the other agents. We have discussed the special case of dynamic strategies where all the agents are learners; further investigation should take place, investigating how an agent can determine if the scenario is changing rapidly and adjust its behaviour accordingly, and in what circumstances some kinds of guarantees may be available.

Approximations In order to test our model even in small scenarios it has been necessary to use many approximations, and we have put together an ad-hoc collection of approximations and generalisations to achieve this. However, we should investigate the properties of these various approximations and determine a formal model for combining approximation techniques more effectively.

Large scale Extending the previous point, in section 1.1 we said that we should explore approaches capable of operating in larger systems with hundreds or thousands of actors. However, the complexity of the approach as it stands makes it intractable for large systems. By implementing some of the techniques discussed in section 2.4 we hope to be able to render the problem more tractable in larger scenarios. For example, in large systems, agents will typically not need detailed information about most of the system, and so a hierarchical system of approximations could be used, with agents maintaining only very approximate information about parts of the problem not relevant to them, and more detailed solutions for parts of the problem which become relevant. Furthermore, information sharing (Dutta et al., 2004) could be used between agents as they move around and require information about

new subproblems. As discussed in Dutta et al. (2007), bandwidth limitations mean that this information sharing must prioritise vital information. Such a scenario might be analogised by the fireman entering a building receiving information from a fireman exiting the building, "rescue the old lady on the top right first" or "beware of the landing on the third floor, it's about to go". Finally, using the approximation and state generalization techniques, and extensions thereof, we should be able to handle continuous state, action and reward spaces.

Reasoning about rewards When actions or states are only partially observable, it may be possible to reason about their likelihood using the observed reward. In particular, in many cases rewards are deterministic functions of the state. Even if these functions are not precisely known at the outset, the probability from them can contribute to the likelihood of a particular state having occurred. We should develop this model formally, and investigate its use under different kinds of reward functions.

Individual goals Rather than consider only cooperative systems in which there is a single shared reward at each step, we would also like to investigate systems in which each agent has a distinct individual reward. In some cases, rewards may be identical or positively correlated for a set of agents. Other agents may be directly competing. Initially, we will assume that all rewards are visible to all agents. We expect this model to be particularly applicable to the partially observable case. This is because in the fully observable case we need only consider our own reward whether or not it is cooperative, but in the partially observable case we may use information about rewards to reason about our estimates of the other agents' strategies.

Partially observable rewards This extension perhaps contrasts with the previous: we may not always know what individual reward another agent receives from certain behaviour. However, if we are able to make assumptions such as rationality in the agent (which may not always be possible), we can begin to guess at its reward function from its behaviour. Whether this will prove useful within our target environments will also be subject to investigation.

Open systems In disaster scenarios such as those described in section 1.1, or simpler rescue scenarios therein, it is likely that agents will be continually entering or leaving the domain. With our separate models for each agent, it should be possible to discard models as agents leave and create new ones for new agents entering the domain. A further challenge in such systems would

be to try and determine similarity of the new agents to the agents already modelled, and thus form accurate models of their behaviour more quickly. For example, when driving in a particular country, one can assume that all drivers conform to similar social rules. Once those rules have been discovered through observation of a subset of the drivers, they can be applied to any new vehicles encountered on the roads. In a UK rescue scenario, one might assume that all teams from a particular county, or a particular hospital, have had similar training, and will respond similarly to a particular problem. Finally, information sharing as mentioned previously could also contribute to faster learning about the changing scenario.

Combining partial action and partial state observability In sections 2 and 4 we proposed example scenarios where the state is known, but agent actions are concealed. Equivalently, we can envisage disaster response scenes in which the complete state is built from observations at a number of vantage points. Unless there are agents at every vantage point, the complete state will be unknown even while all agents' actions are fully observable (perhaps because they transmit them over broadcast radio). However, a more common case for partially observable problems is a problem in which each agent has a limited field of view. In such cases, it is likely that not only the states, but the actions will be partially observable. Handling cases where both states and actions are partially observable will be a prerequisite for extending our model into our target domain.

5.3 Timeline

We propose to address the above issues as follows:

- Immediately, we will investigate the theoretical properties of our current models, determining what convergence and regret guarantees may exist, or what conditions are required for such guarantees. In particular, we will focus on the implications of dynamic systems and dynamic agent strategies.
- Concurrently with the theoretical investigation, we will work on better ways to approximate the model, in order to extend the system into larger domains.
- We will then extend the model to include non-cooperative domains in which agents may have distinct individual rewards.

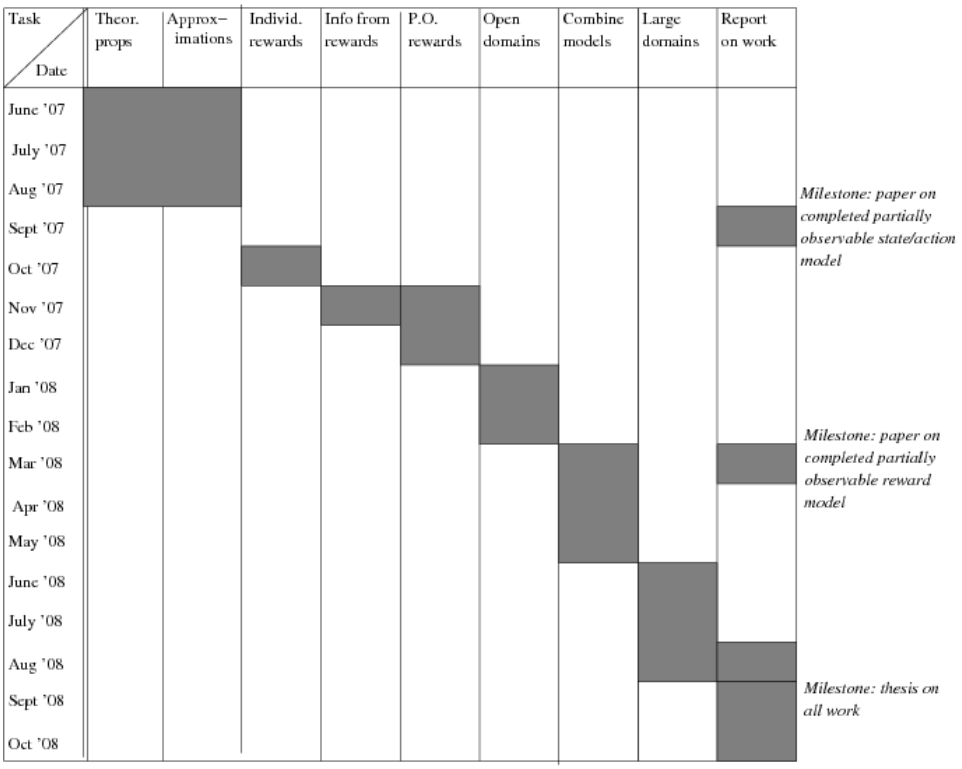


TABLE 5.1: Work plan

- Next, we will extend the model to glean information about likely transitions or actions from the observed rewards.
- Continuing the theme of experimenting with reward structure, we will model systems where the rewards may be partially observable, in particular heterogeneous systems.
- Building on the use of individual agent models, we will investigate open domains in which new agents must be discovered and learned about, while the loss of known agents must be accounted for. Within these systems, we will make use of techniques for sharing knowledge about the other agents between the agents (e.g. (Dutta et al., 2004)).
- Finally, bringing together and extending the above approximation techniques, knowledge sharing, and our various models of partial observability, we will implement our model in a large scale, open, competitive domain.

Table 5.1 gives a timeline for the work.

Appendix A

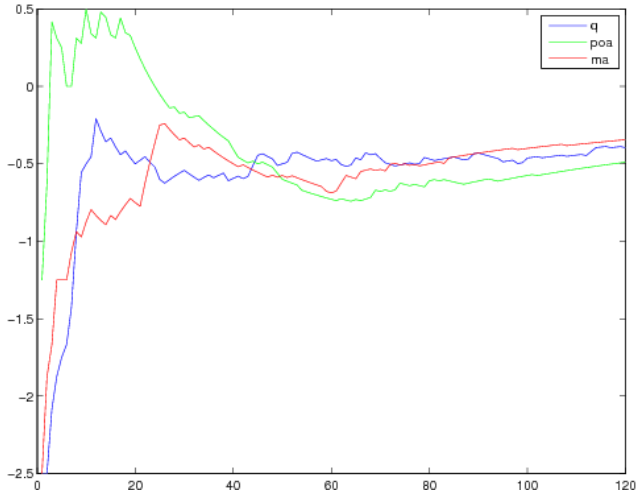
Results

The following graphs show the results of several experiments comparing a q-learner ('q'), a Bayesian partially observable multi-agent learner ('poa') (our learner), and a Bayesian multi-agent learner which is able to fully observe the actions of other agents ('ma'), on the small well problem of chapter 4.

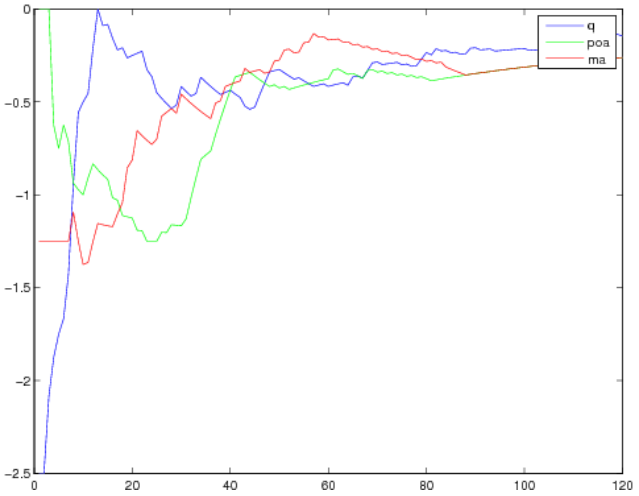
The first set of graphs (figure A.1) shows the effect of varying the randomness parameter in the system, gradually reducing the randomness to 0. The q-learner's performance increases dramatically as the randomness is reduced; by contrast there is little change in the behaviour of the multi-agent learners.

The second set of graphs (figure A.2) shows experiments with alterating the randomness parameter part way through the experiment, simulating a change in environmental conditions. For these experiments, the sample size has been increased from 7 for each model in the belief state (transition model and opponent model) to 10. This affects the multi-agent learners, improving the accuracy of their estimates at each step, and their behaviour is correspondingly improved.

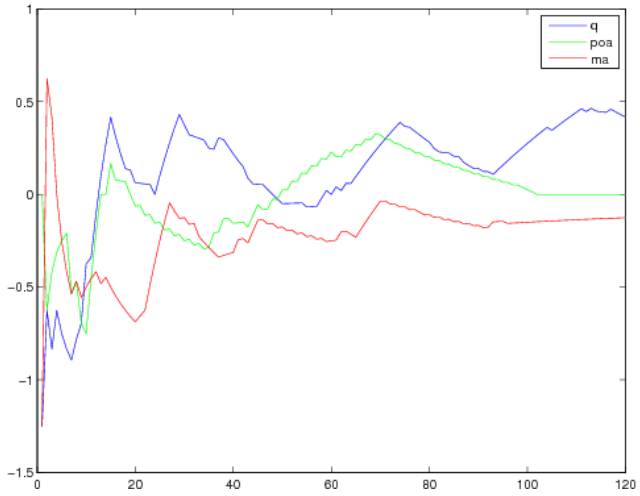
The third and fourth sets of graphs (figures A.3 and A.4) show the results of paired t-tests between our learner and the two comparison learners, for the seven experiments described above. From these we can take away that particularly with a sample rate of 7 the confidence in our results is not very high, although as the environmental randomness is increased the variability of the results reduces, increasing confidence. As the sample rate is increased, so does the confidence in the results. We would expect this trend to continue.



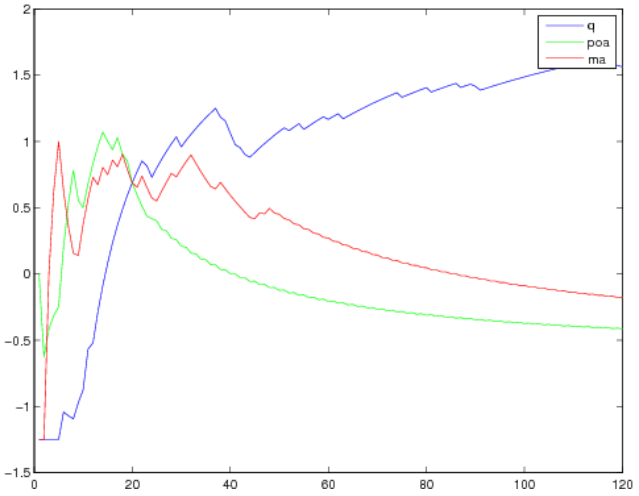
(a) Sample size 7^2; delta=0.5



(b) Sample size 7^2; delta=0.65



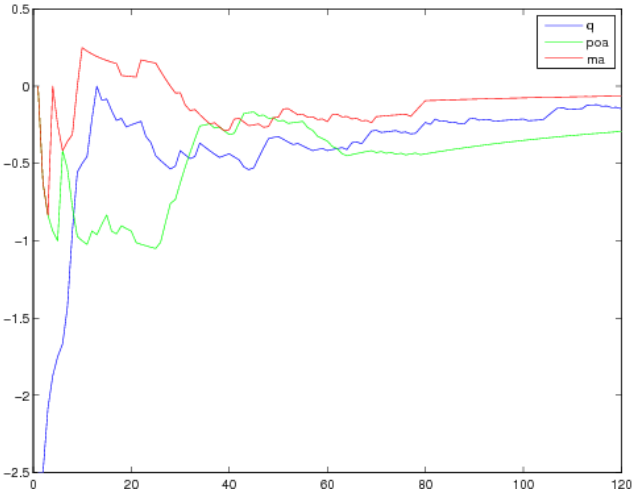
(c) Sample size 7^2; delta=0.8



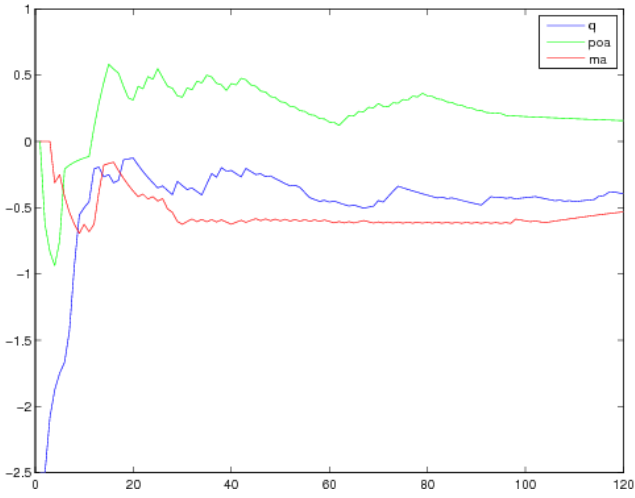
(d) Sample size 7^2; delta=1.0

FIGURE A.1: Two-agent well test: results (1)

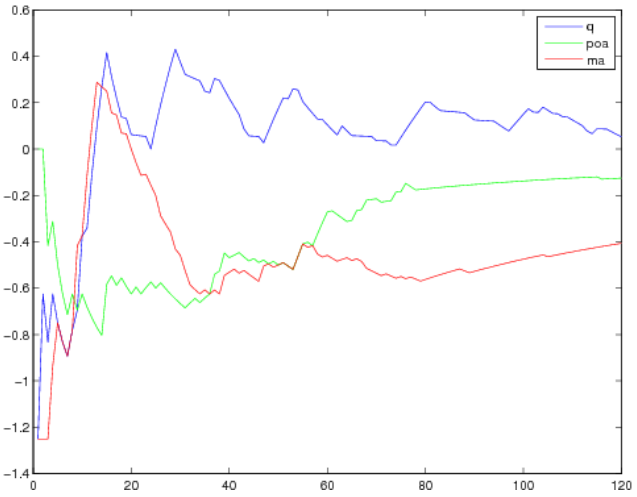
The final two sets of graphs (figures A.5 and A.6) show the error bars for the three learners over the seven experiments, indicating the degree of variability in the behaviour of the learners.



(a) Sample size 10^2; delta=0.65

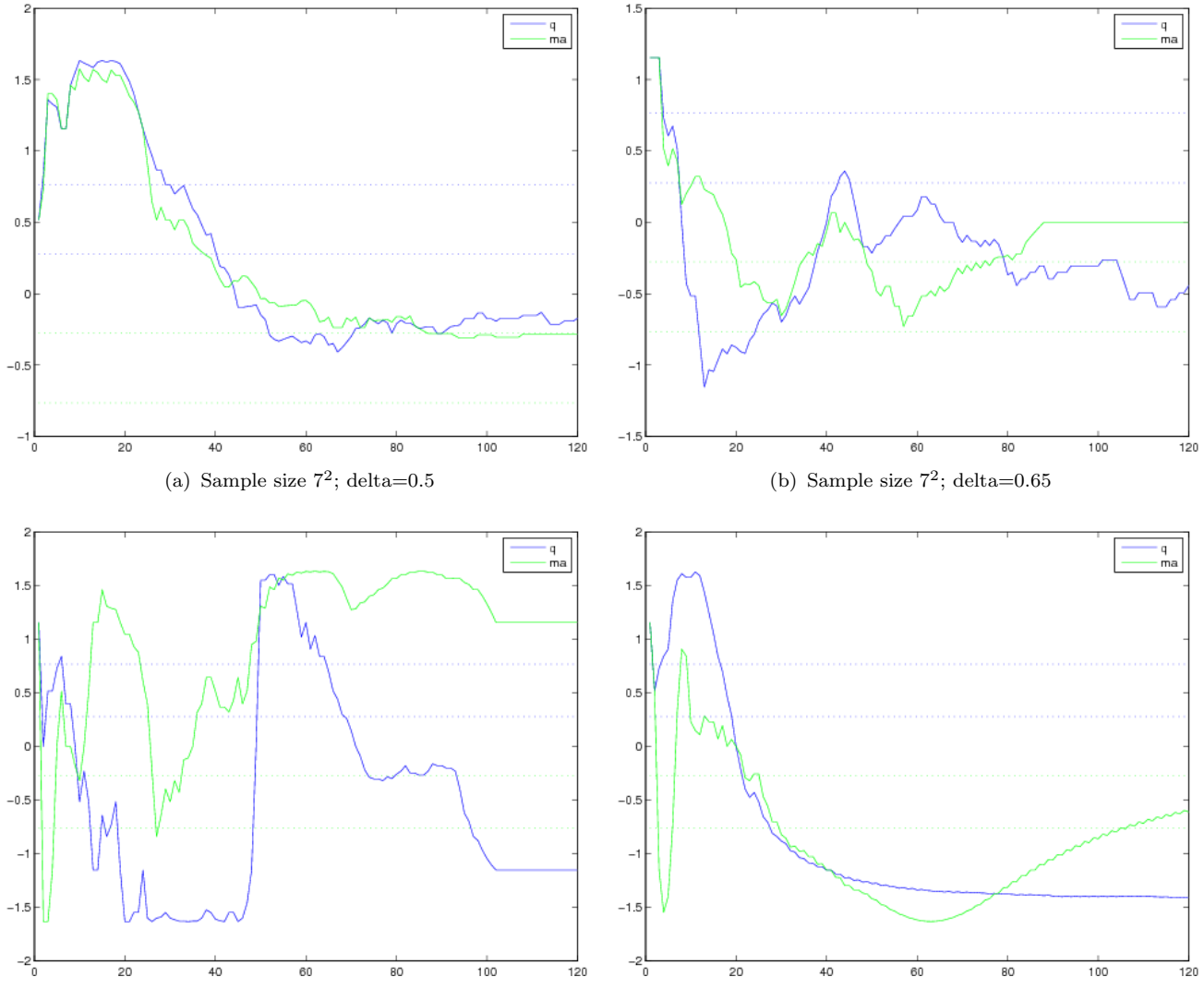


(b) Sample size 10^2; delta=0.5 → 0.8



(c) Sample size 10^2; delta=0.8 → 0.5

FIGURE A.2: Two-agent well test: results (2)



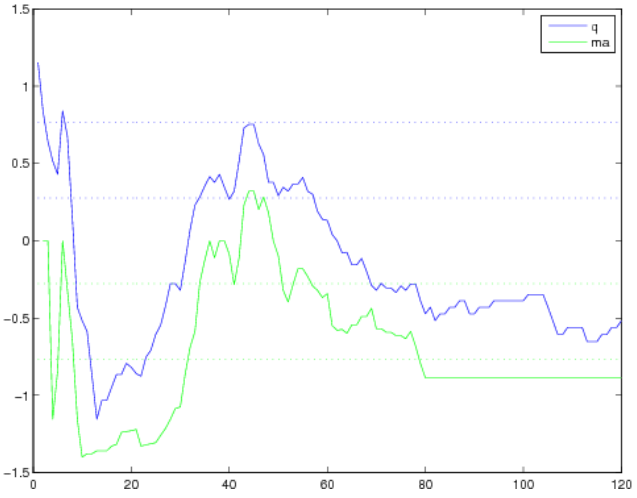
(a) Sample size 7^2; delta=0.5

(b) Sample size 7^2; delta=0.65

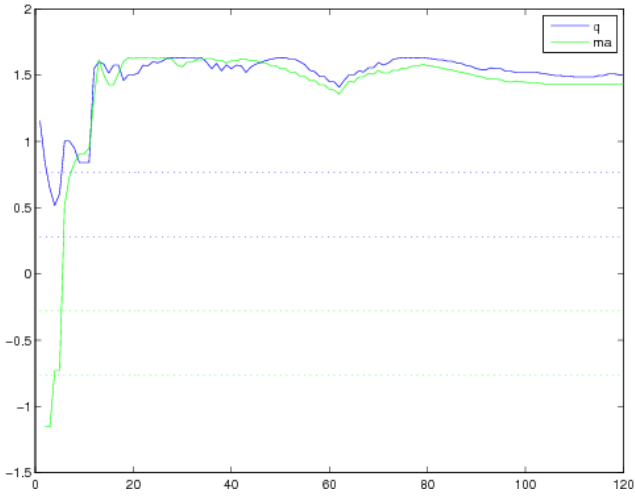
(c) Sample size 7^2; delta=0.8

(d) Sample size 7^2; delta=1.0

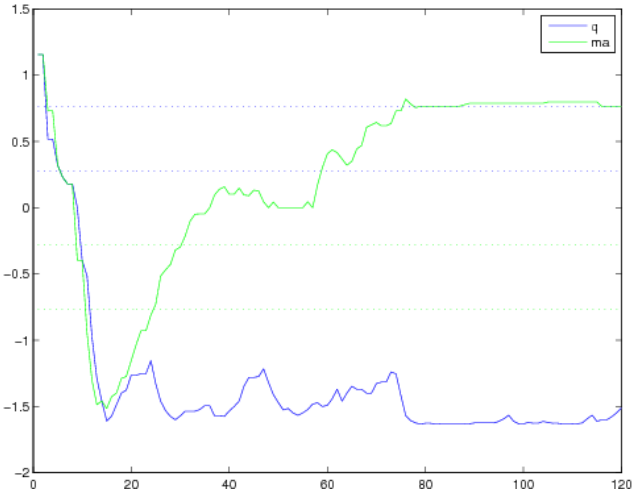
FIGURE A.3: Two-agent well test: t-tests (1)



(a) Sample size 10^2; delta=0.65

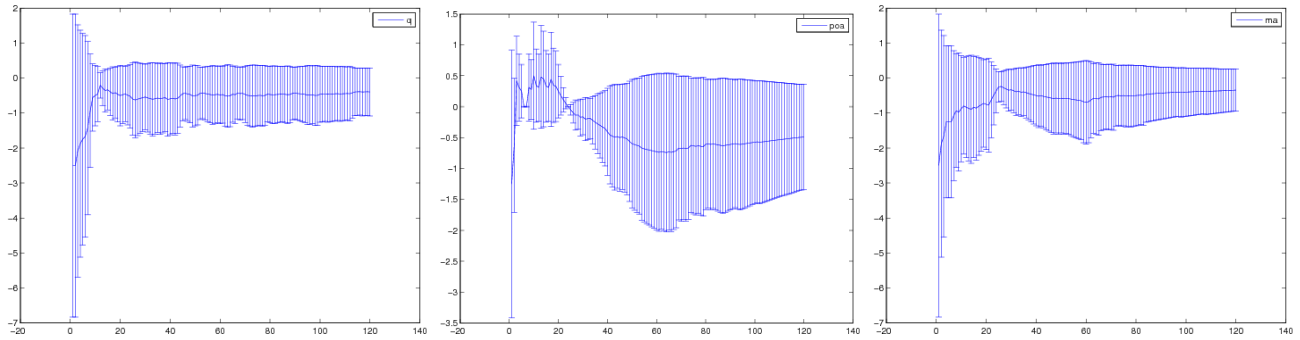


(b) Sample size 10^2; delta=0.5 → 0.8

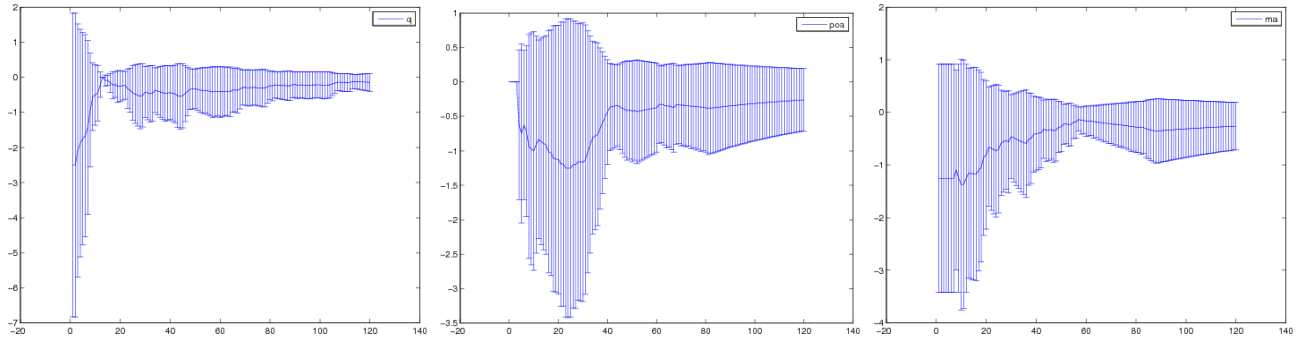


(c) Sample size 10^2; delta=0.8 → 0.5

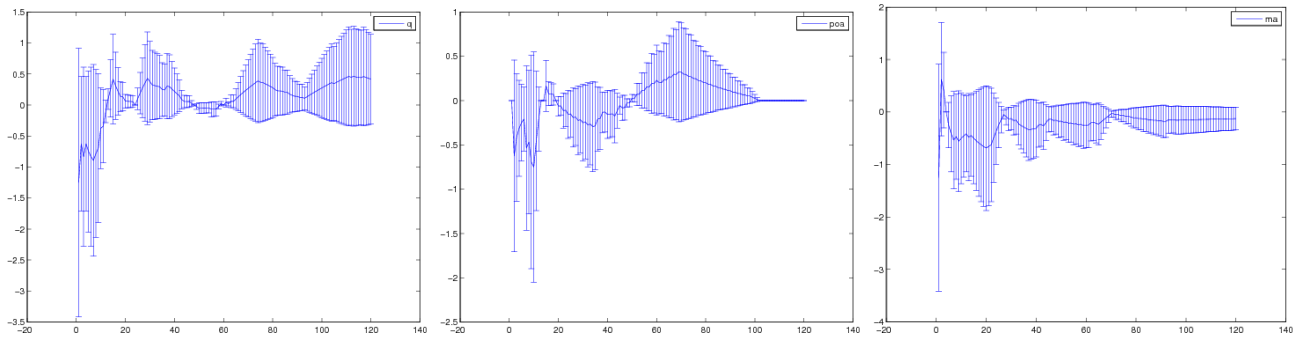
FIGURE A.4: Two-agent well test: t-tests (2)



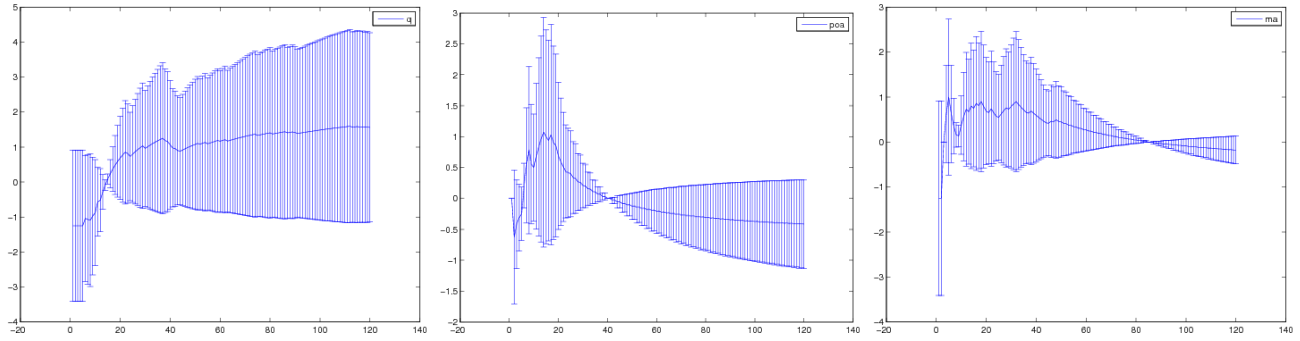
(a) Sample size 7^2; delta=0.5. q, poa, ma



(b) Sample size 7^2; delta=0.65. q, poa, ma

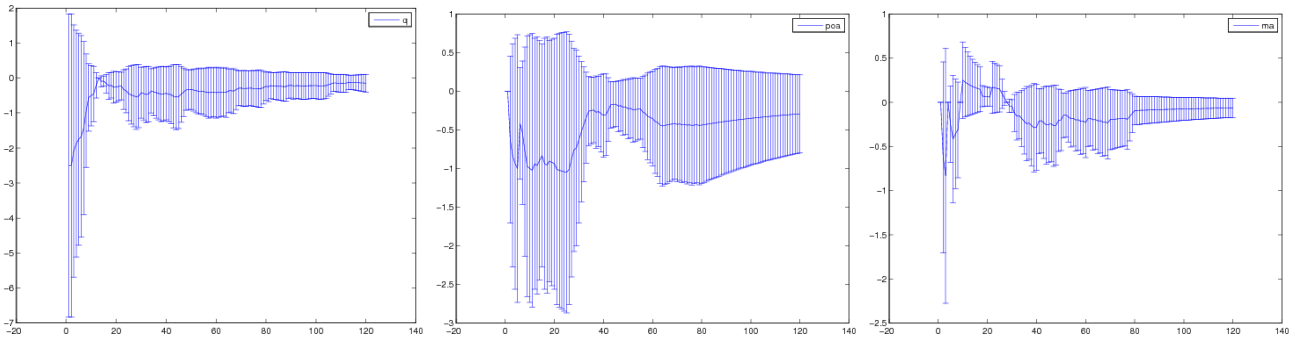


(c) Sample size 7^2; delta=0.8. q, poa, ma

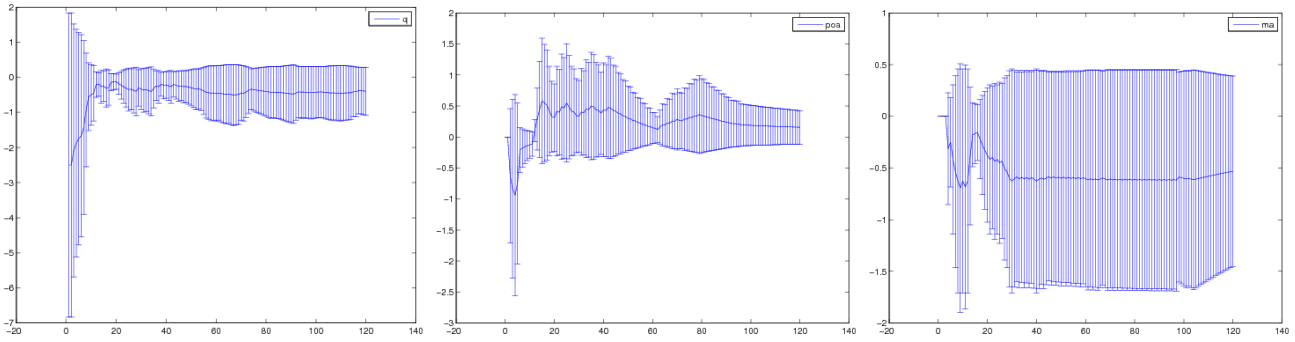


(d) Sample size 7^2; delta=1.0. q, poa, ma

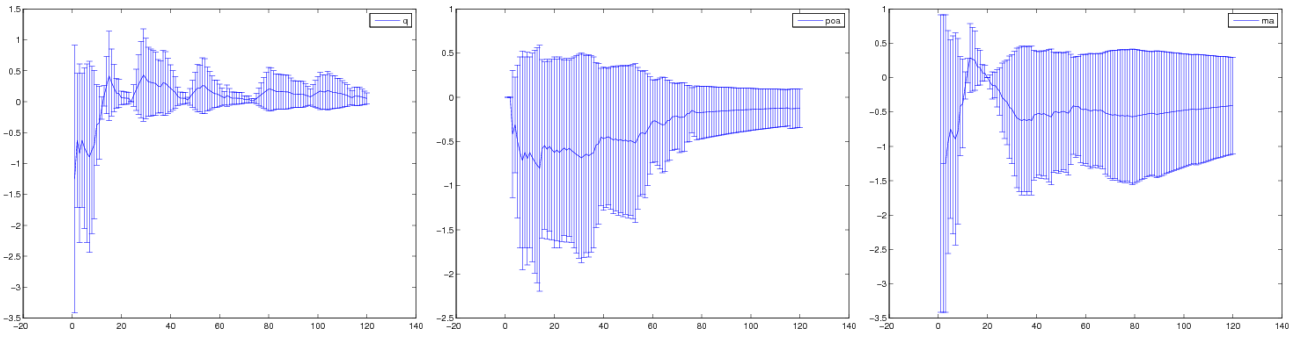
FIGURE A.5: Two-agent well test: error bars (1)



(a) Sample size 10^2; delta=0.5. q, poa, ma



(b) Sample size 10^2; delta=0.5 → 0.8. q, poa, ma



(c) Sample size 10^2; delta=0.8 → 0.5. q, poa, ma

FIGURE A.6: Two-agent well test: error bars (2)

Bibliography

Aberdeen, D. and Baxter, J. (2002). Scaling internal-state policy-gradient methods for POMDPs. In *Proceedings of the 19th International Conference on Machine Learning*, volume 2, pages 3–10, Sydney, Australia. Morgan Kaufmann.

Abul, O., Polat, F., and Alhajj, R. (2000). Multiagent reinforcement learning using function approximation. In *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, volume 30, pages 485–497.

Amato, C., Bernstein, D. S., and Zilberstein, S. (2006). Solving POMDPs using quadratically constrained linear programs. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 341–343, New York, NY, USA. ACM Press.

Baxter, J. and Bartlett, P. L. (2000). Reinforcement learning in POMDPs via direct gradient ascent. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, pages 41–48, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bishop, C. M. (2004). *Neural networks for pattern recognition*. Oxford University Press.

Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*, pages 195–210, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bowling, M. (2005). Convergence and no-regret in multiagent learning. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 209–216. MIT Press, Cambridge, MA.

Bowling, M. and Veloso, M. (2001). Rational and convergent learning in stochastic games. In *International Joint Conferences on Artificial Intelligence*, pages 1021–1026.

Burke, J. (2003). *Moonlight in Miami: A Field Study of Human-Robot Interaction in the Context of an Urban Search and Rescue Disaster Response Training Exercise*. PhD thesis, University of South Florida.

Cassandra, A., Littman, M., and Zhang, N. (1997). Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 54–61, San Francisco, CA. Morgan Kaufmann.

Chalkiadakis, G. and Boutilier, C. (2003). Coordination in multiagent reinforcement learning: a bayesian approach. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 709–716, New York, NY, USA. ACM Press.

Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 746–752, Menlo Park, CA, USA. American Association for Artificial Intelligence.

Committee on using IT to Enhance Disaster Management, N. R. C. (2005). Summary of a workshop on using information technology to enhance disaster management.

David, E., Rogers, A., Schiff, J., Kraus, S., Rothkopf, M., and Jennings, N. R. (2007). Optimal design of english auctions with discrete bid levels. *ACM Transactions on Internet Technology*, 7.

Dearden, R., Friedman, N., and Andre, D. (1999). Model-based Bayesian exploration. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 150–15, San Francisco, CA. Morgan Kaufmann.

Dorais, G., Bonasso, R., Kortenkamp, D., Pell, P., and Schreckenghost, D. (1998). Adjustable autonomy for human-centered autonomous systems on Mars. Presented at the Mars Society Conference.

Durfee, E. H. (1999). Practically coordinating. *AI Magazine*, 20(1):99–116.

Dutta, P. S., Dasmahapatra, S., Gunn, S. R., Jennings, N., and Moreau, L. (2004). Cooperative information sharing to improve distributed learning. In *Proceedings of The AAMAS 2004 workshop on Learning and Evolution in Agent-Based Systems*, pages 18–23.

Dutta, P. S., Goldman, C. V., and Jennings, N. R. (2007). Communicating effectively in resource-constrained multi-agent systems. In *IJCAI*, pages 1269–1274.

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.

Excelente-Toledo, C. B. and Jennings, N. R. (2005). Using reinforcement learning to coordinate better. *Computational Intelligence*, 21(3):217–245.

Fischer, F., Rovatsos, M., and Weiss, G. (2004). Hierarchical reinforcement learning in communication-mediated multiagent coordination. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1334–1335, Washington, DC, USA. IEEE Computer Society.

Fischer, F., Rovatsos, M., and Weiss, G. (2005). Acquiring and adapting probabilistic models of agent conversation. In Koenig, S. and Wooldridge, M., editors, *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 106–113. ACM Press.

Fitoussi, D. and Tennenholtz, M. (2000). Choosing social laws for multi-agent systems: Minimality and simplicity. *Artificial Intelligence*, 119(1-2):61–101.

Fogel, D. B. (2002). *Blondie24: playing at the edge of AI*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Fudenberg, D. and Levine, D. K. (1998). *The Theory of Learning in Games*. MIT Press.

Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *AAAI*, pages 709–715.

Hoar, J. (1996). Reinforcement learning applied to a real robot task. DAI MSc Dissertion, University of Edinburgh.

Hoey, J. (2001). Hierarchical unsupervised learning of facial expression categories. *ICCV Workshop on Detection and Recognition of Events in Video*.

Izadi, M. T. and Precup, D. (2006). Exploration in POMDP belief space and its impact on value iteration approximation. In *European Conference on Artificial*

Intelligence, Workshop on Planning, Learning and Monitoring with Uncertainty and Dynamic Worlds (PLMUDW).

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.

Kazakov, D. and Bartlett, M. (2004). Cooperative navigation and the faculty of language. *Applied Artificial Intelligence*, 18(9-10):885–901.

Kim, Nair, Varakantham, Tambe, and Yokoo (2006). Exploiting locality of interaction in networked distributed pomdps. In *Proceedings of the AAAI Spring Symposium on "Distributed Plan and Schedule Management".*

Leslie, D. (2004). *Reinforcement learning in games.* PhD thesis, University of Bristol.

Lesser, V. (1999). Cooperative Multiagent Systems: A Personal View of the State of the Art. *IEEE Transactions on Knowledge and Data Engineering*, 11(1).

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, New Brunswick, NJ. Morgan Kaufmann.

Matsuno, Y., Ymazaki, T., Ishii, S., and Matsuno, J. (2001). A multi-agent reinforcement learning method for a partially-observable competitive game. In *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, pages 39–40, New York, NY, USA. ACM Press.

Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill.

Panait, L. and Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434.

Paquet, S., Tobin, L., and Chaib-draa, B. (2005). An online pomdp algorithm for complex multiagent environments. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 970–977, New York, NY, USA. ACM Press.

Pfeffer, A., Koller, D., and Takusagawa, K. T. (2000). State-space approximations for extensive form games. Workshop paper at First World Congress on Game Theory.

Rezek, I., Reece, S., Roberts, S. J., Rogers, A., Dash, R. K., Jennings, N. R., and Leslie, D. S. (2006). On similarities between inference in game theory and machine learning. Machine Learning Journal: to appear.

Rogers, A., David, E., Schiff, J., and Jennings, N. R. (2006). The effects of proxy bidding and minimum bid increments within ebay auctions. *ACM Transactions on the Web.*

Roweis, S. (2003). Hidden markov models. SCIA Tutorial.

Roy, N. and Gordon, G. (2002). Exponential family PCA for belief compression in POMDPs. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing 15 (NIPS)*, pages 1043–1049, Vancouver, Canada.

Sallans, B. A. (1999). Learning factored representations for partially observable Markov decision processes. In *Neural Information Processing Systems*, pages 1050–1056.

Sallans, B. A. (2002). *Reinforcement learning for factored Markov decision processes*. PhD thesis, University of Toronto. Adviser-G. E. Hinton.

Scerri, P., Sycara, K., and Tambe, M. (2004). Adjustable autonomy in the context of coordination. In *AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*. Invited Paper.

Schurr, N., Marecki, J., Lewis, J. P., Tambe, M., and Scerri, P. (2005). The defacto system: Coordinating human-agent teams for the future of disaster response. In *Multi-Agent Programming*, pages 197–215. Springer.

Shani, G., Brafman, R. I., and Shimony, S. E. (2005). Model-based online learning of pomdps. In *European Conference on Machine Learning*, pages 353–364.

Shi, J. and Littman, M. L. (2002). Abstraction methods for game theoretic poker. In *CG '00: Revised Papers from the Second International Conference on Computers and Games*, pages 333–345, London, UK. Springer-Verlag.

Shoham, Y., Powers, R., and Grenager, T. (2003). Multi-agent reinforcement learning: a critical survey. Technical report, Stanford University.

Smith, A. J. (2002a). Dynamic generalisation of continuous action spaces in reinforcement learning: A neurally inspired approach. Ph.D. thesis, Division of Informatics, Edinburgh University, UK.

Smith, A. J. (2002b). Dynamic generalisation of continuous action spaces in reinforcement learning: A neurally inspired approach. Ph.D. thesis, Division of Informatics, Edinburgh University, UK.

Stenning, K. and van Lambalgen, M. (2005). Semantic interpretation as computation in nonmonotonic logic: the real meaning of the suppression task. *Cognitive Science*, 29(6).

Sun, R. and Naveh, I. (2004). Simulating organizational decision-making using a cognitively realistic agent model. *Journal of Artificial Societies and Social Simulation*, 7(3).

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Takeuchi, I., Kakumoto, S., and Goto, Y. (2003). Towards an Integrated Earthquake Disaster Simulation System. *First International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster*.

Tambe, M., Adibi, J., Alonaizon, Y., Erdem, A., Kaminka, G. A., Marsella, S., and Muslea, I. (1999). Building agent teams using an explicit teamwork model and learning. *Artificial Intelligence*, 110(2):215–239.

Tanner, B., Bulitko, V., Koop, A., and Paduraru, C. (2007). Grounding abstractions in predictive state representations. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1077–1082.

Wang, F. (2002). Self-organising communities formed by middle agents. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 1333–1339, New York, NY, USA. ACM Press.

Wooldridge, M. (2002). *An Introduction to Multi-agent Systems*. Wiley.