UNIVERSITY OF SOUTHAMPTON

A Learning Based Approach to Modelling Bilateral Adaptive Agent Negotiations

by

Vidya Narayanan

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

School of Electronics and Computer Science

January 2008

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Vidya Narayanan

In large multi-agent systems, individual agents often have conflicting goals, but are dependent on each other for the achievement of these objectives. In such situations, negotiation between the agents is a key means of resolving conflicts and reaching a compromise. Hence it is imperative to develop good automated negotiation techniques to enable effective interactions. However this problem is made harder by the fact that such environments are invariably dynamic (e.g. the bandwidth available for communications can fluctuate, the availability of computation resources can change, and the time available for negotiations can change). Moreover, these changes can have a direct effect on the negotiation process. Thus an agent has to adapt its negotiation behaviour in response to changes in the environment and its opponent's behaviour if it is to be effective. Given this, this research has developed negotiation mechanisms that enable an agent to perform effectively in a particular class of negotiation encounters; namely, bilateral negotiation in which a service provider and a service consumer interact to fix the price of the service.

In more detail, we use both reinforcement and Bayesian learning methods to derive an optimal agent strategy for bilateral negotiations in dynamic environments with incomplete information. Specifically, an agent models the change in its opponent's behaviour using Markov Chains and determines an optimal policy to use in response to changes in the environment. Also using the Markov chain framework, the agent updates its prior knowledge of the opponent by observing successive offers using Bayesian inference and hence strategically responds to its opponent. This framework for adaptive negotiation in non-stationary environments incorporates two novel learning algorithms that use reinforcement and Bayesian learning techniques to respond to the various forms of dynamism. Having devised the algorithms, we analytically show that the former learns an optimal policy for negotiating in a non-stationary environment and the latter converges over repeated encounters to the opponent's true strategic model. These empirical results show that the reinforcement learning algorithm successfully concludes 83% of the negotiations in dynamic scenarios and that when using the Bayesian algorithm the opponent learns the true model of an adaptive opponent's behaviour in 95% of the encounters. Both of these results compare very favourably with the previous state of art. We have also done a comparison of the these two algorithms. The empirical results show that using reinforcement learning a very high percentage (90%) of dynamic negotiation encounters end in agreement, whereas using Bayesian learning techniques the agent earns a large share of the profits (89%) in the negotiation process.

Contents

List of Figures

List of Tables

Nomenclature

s_i	Strategy of player i.
p_i	Payoff function of player i.
T^a	Deadline of agent a.
f^a	Negotiation Decision Function of agent a.
ψ	Control Parameter.
V^a	Value function of agent a.
S_i	Strategy Space of player i.
ψ	Control Parameter
u_i	Utility function of agent i.
I	Information State.
IP^a	Initial Price of agent a.
RP^a	Reserve Price of agent a.
RA	Resource Availability for the negotiation process
$\alpha_j^a(t)$	Polynomial Negotiation Decision Function of agent a.
R^t	Return function.
$A(s_t)$	Set of available actions for the agent.
S	Set of all possible states that the system can be in.
ψ_t	Reinforcement learning policy.
$(X_n, n = 0, 1, 2, ...)$	Markov Process.
s	a state of the system.
$T^a_{ss'}$	State transition function $s \rightarrow s'$.

$R^a_{ss'}$	Reward function associated with state transition $s \to s'$.
$V^\pi(s)$	State value function.
$Q^\pi(s,a)$	Action value function.
$V^*(s)$	Optimal state value function.
$Q^*(s,a)$	Optimal action value function.
$P_n(s,a,s')$	Estimate function.
$P^n(t)$	Probability that opponent strategy changes at the n^{th} step of the process.
$u^n_{s_x}(t)$	Payoff function of buyer X
$u^n_{s_{max}}(t)$	Maximum payoff function of buyer X
P^n_{ij}	n step probability transition function.
p^n_k	Probability that the process is in state k at the n^{th} step of the process.
$O_n(t)$	Sequence of offers made by opponent.
$O_n(t)$	Event that the offer at the n^{th} step of the process is p.
$H_n(t)$	Sets of hypotheses about $P^n(t)$.
L	Likelihood function.
$\alpha^n(t)$	Prior distribution for bayesian learning model.

Acknowledgements

I would like to thank my supervisor Prof. Nick Jennings for giving me the opportunity to work in the area of automated negotiations. He has always been receptive to new ideas and has actively encouraged independent research. His guidance has been instrumental in my evolution into a seasoned researcher. It has been a pleasure and a privilege to work with him. I would also like to thank the MVCE project for funding the initial years of my study. I would also like to thank all my colleagues and friends at IAM for providing a very stimulating research environment. I have thoroughly enjoyed the time spent with them. I would also like to thank all my family, in particular my husband, Nitin, and my daughter Shreya for all their emotional support and encouragement. I would also like to thank my parents for always valuing academic endeavour and for being excellent role models. Finally I would also like to thank the excellent IAM staff, Jane, Jo, Becky and others for their support.

Chapter 1

Introduction

Many large systems, in diverse fields like mobile communications, defence logistics, and medical systems, are being implemented as multi-agent systems [Jennings et al., 1995; Luck et al., 2003]. Now, in most of these cases, the individual agents in these systems need to interact with one another in order to achieve their aims. Moreover, in many cases, these goals may conflict with one another. Given all of this, the agents need to negotiate with each other in order to reach an agreement that is acceptable to all the parties [Rosenschein and Zlotkin, 1994]. However, as these systems are being situated in ever more complex environments, it is becoming increasingly harder to build agents with sufficient sophistication to handle all the demands being made on them. In particular, one of the most important such demands is for the agents to sense and adapt their negotiation behaviour to changes in the environment (e.g. when the resource available for negotiations change or the time available for the negotiation changes). With this objective in mind, we have focussed our efforts on building an automated negotiation model that can adapt to changes in the environment and in the behaviour of the negotiation opponents.

By means of an illustration, consider a typical business-to-consumer e-commerce scenario in which an agent (which forms a part of a large multi-agent system), acting on behalf of a retailer, negotiates online with a specific consumer for the price of a certain product [He et al., 2003]. Consider the case in which the retailer is negotiating for many products simultaneously and, therefore, has many such agents acting on his behalf. The agents may well share resources like bandwidth for communication and computational power and they are likely to reside in the same computational space. Now, in such situations, an agent might suddenly find that its messages take considerably longer to reach its opponent because of increased network activity or that it has to wait longer for CPU time since one of the other agents is using a large number of cycles. In either case, the resources available to carry out the negotiation change, either increasing or decreasing, depending on the nature of the changes. In yet other cases, an agent's allocated budget for acquiring the desired service might be cut. For instance, in a large supply chain (implemented as a multi-agent system), where several agents pursue independent goals [Christopher, 2005], the budget spent in one part of the supply chain may well impact the budget further down the line (because the agent is suddenly given more money as a result of savings or because it is given a lower budget as a result of overspending). In this case, in negotiation terms, the agent's reservation price (RP) [1] changes. Finally, in view of the goal of the entire supply chain system, the agents may need to shorten or lengthen the time available for them to complete their negotiation because earlier activities take a shorter or longer time to procure than was initially expected. In this case the agent's negotiation deadline changes. Now in all these cases, the agents need to adapt their negotiation strategies if they are to be effective in their changed environment. Failure to adapt may lead to poor outcomes (such as no agreement reached before the deadline, an agreement that is disadvantageous to one agent or the inability of an agent to meet its existing commitments) and may leave the owner dissatisfied because he has not been able to exploit the negotiation process for his benefit.

From the above examples, it is clear that in such complex systems the agents have to adapt to changes in the environment in order to bargain effectively. Now, having de-

[1] The reservation price is the threshold price beyond which the agent is not authorised to spend.

scribed the general characteristics of multi-agent domains and the need for adaptation in such systems, we move on to describing a number of real-life systems. To ground this discussion, we focus, in particular, on the domain of mobile communications, in which automated negotiation technology has been identified as playing a central role (see section 1.2 for more details). We focus on mobile communications here because as the use of mobile phones, Personal Digital Assistants (PDAs) and other wireless devices become increasingly widespread, there is an urgent need, dictated by prevailing market conditions, to develop advanced and robust software solutions in these domains [MVCE, 2004]. However, although we use this domain for illustrating how our learning techniques can be applied, our model is not restricted to this domain. Moreover, it has been argued that automated negotiation will form an important part of these application contexts as distinct stakeholders bargain for services like television, broadband, bluetooth and resources like bandwidth [Kraus, 2001].

In particular, it is our aim to provide practical solutions for automated negotiation in this domain and to develop a strong theoretical foundation for our negotiation strategies. This foundation is important as it helps to establish a concrete relationship between the conditions prevailing in the domain in which the agents negotiate and the possible outcomes of the negotiation process (for instance, the maximum benefit that can be obtained by engaging in negotiations, the time at which the negotiation will conclude or the best strategy to use given that the opponent is using a particular strategy). This, in turn, enables the agents, given the current conditions, to decide whether to enter into negotiations, to decide what their final offer should be, and to ascertain when to make this offer. The practical aspect of the domain is important in this context because typically here the deadlines of the agents would be very short and, therefore, the computation of a suitable negotiation strategy should also be done quickly. Hence our solution should also be designed in a manner keeping this aspect of the mobile communication domain in mind.

This uncertainty and dynamism requires an agent to continuously update its knowledge during its negotiations if it is to bargain effectively. Thus we believe that *learning* about the other agents in the system and about their common environment is essential for negotiating in this domain [Narayanan and Jennings, 2005]. To this end, we develop a novel method, based on learning for adapting to different situations and verify that our method can indeed improve an agent's negotiation performance.

Being more specific still, we consider negotiation between a pair of agents over a single issue (price). We then use the Markov chain theory [Howard, 1960] to model how the negotiation process changes. From this point of departure, we use reinforcement learning (RL) techniques to develop an algorithm that deals with changes in the agent's negotiation parameters (see Chapter 3) and use Bayesian Learning (BL) techniques to show how the agent can adapt to changes in its opponent's behaviour (see Chapter 4). We have also empirically evaluated these techniques and compared their performance under a range of different conditions (see Chapter 5).

The rest of this chapter is organized as follows: in Section 1.1 we describe the main components of a negotiation model in order to show how our model incorporates these elements in its structure. Then in Section 1.2 we describe some scenarios that motivate the requirements of our model, following which, in Section 1.3, we spell out the main requirements of our model. In Section 1.4 we state our research contributions and, finally, in Section 1.5 we describe the thesis structure.

1.1 Components of a Negotiation Model

Having highlighted the importance of automated negotiation, we now describe in more detail the main components of an automated negotiation model [Jennings et al., 2001]:

- *The Negotiation Protocol*: Formally specifies the rules of the negotiation process — who can participate, the states of the negotiation process, and the events that change the state of the process. Now, there are literally millions of protocols including auctions that are in used in agent negotiations [Wurman et al., 1998]. These cover one-to-one, one-to-many and many-to-many encounters. However in this work we focus on the bilateral case. Specifically, we consider the case in which a buyer and a seller agent negotiate over the price of a service. Like most work in the field, we adopt the *alternating offers protocol* [Rubinstein, 1982] in which one of the agents makes an offer in one time step and then waits for the opponent's response in the next step before making another offer. Thus the agents alternate in making offers.

- *The Negotiation Objects*: Represents the issues over which the agents are negotiating. The agents could negotiate over a single issue, say the price of a commodity or the quality of a service, or they could negotiate over multiple issues. The work in this thesis considers negotiations only over a single issue. However future work may remove this assumption.

- *The Negotiation Parameters*: Represents factors like the negotiation deadlines and the reservation prices that play an important part in deciding the strategies of the agents.

- *The Participants' Negotiation Preferences*: These represent the objectives of the agents participating in the negotiation process. One of the most common ways of representing such preferences is through the medium of utility functions [Kannai, 1977; Keeney and Raiffa, 1976]. Intuitively, these can be described as mathematical entities that represent the stake that the agents have in the negotiation process. By using these functions, we can formally study how the actions of the agent affect its utility during the negotiation process. However, in our case, we do not want to have a single utility function that the agents attempt to maximize (since this is not fixed and could change with the environment in our work). Given this, we want to be able to maximize a numerical signal that the agents receive from the environment as a reward for choosing a specific action from a specific state.

This formulation gives us maximum flexibility in dealing with evolving agent preferences (e.g. cost over quality or time over cost).

- *The Participants' Negotiation Strategies*: A strategy specifies how an agent should respond to a given situation. Good strategies will guide an agent towards achieving its objectives which in our case are typically to maximize its reward signal and to respond effectively to changes in its opponent's strategies.

Having detailed the components of a general negotiation model, we now move on to describing scenarios from the mobile communications environment which illustrate the dynamism inherent to this domain and which our solution must be capable of coping with.

1.2 Motivating Scenarios

In order to ground this work still further, we consider two scenarios from the mobile communications domain [MVCE, 2004]. We have chosen these two scenarios because they specifically illustrate how the different negotiation components (as noted above) can change during the course of the negotiation process. Together these two scenarios highlight the key characteristics of environmental change and opponent adaptability in a generic multi-agent system. In more detail, Scenario 1 principally talks about how changes in the environment can affect negotiation parameters like deadlines and reserve prices. Whereas, Scenario 2 describes a situation in which the agent's opponent is adaptive and illustrates the need to develop strategies to counter the opponent's play.

1.2.1 Scenario 1

A salesman needs to travel to another city for a meeting. He has a number of devices at home to perform certain tasks while he is away. Among these is an agent that functions as a set-top box (STB) to record his favourite TV programs, an agent that forwards his important mails to him, and an agent that transmits the phone messages of certain important callers to him. These are connected with wireless devices and form a Personal Area Network (PAN). These agents share a common environment and compete for computational resources (see figure 1.1). The salesman has requested his STB agent to record and transmit an episode of his favourite program to him. This episode is available on two separate TV networks: a large terrestrial broadcaster (pay per view, but no advertisements) and a satellite network specializing in old TV shows (less expensive, but with advertisements). The STB would therefore have to negotiate with the broadcasters to obtain the telecast of the episode. The STB would have to take into account the preferences of the salesman (like the time he would like to watch the program, the amount he is willing to spend for viewing the program) while negotiating on his behalf. Based on the salesman's initial preferences, the STB agent begins negotiations with broadcasters. Initially, it begins negotiating with the terrestrial broadcaster because the salesman doesn't want to spend too much time on watching the program and is willing to spend a little more for watching the program without advertisements. Both the STB agent and the terrestrial broadcaster are initially unwilling to concede on price and while the negotiation is in progress, the mail agent indicates to the STB agent that it has some important emails to transmit to the salesman and needs to use some of the bandwidth that the STB agent has been using for its negotiations. Now the STB agent cannot continue to be stubborn in its dealings with the broadcaster because it would soon have to give up some its communication space to the email agent. The STB agent now has two options, either to adopt a more conceding approach with the broadcaster in the hope of concluding the negotiations quickly or switch to the satellite broadcaster from whom it can get the program for a much lower price. Thus the STB agent has to adapt to changes in resource availability and depending on the current state of the environment it has to choose an appropriate action in order to satisfy the demands of its owner.

Now assume that the agent decides to be conceding with the terrestrial broadcaster, meanwhile the salesman at the meeting receives the news that there will be a cut in his bonus due to some bad investments by his company. He decides therefore that he cannot afford to spend as much money on watching the program as he previously thought. Accordingly he sends this new reduced price to the STB agent. For the STB agent its RP has changed and therefore it has to adjust to this new situation. Specifically, the agent has to deal with the satellite broadcaster and also adopt a stubborn approach so that it can get the program at a lower price. Again at the meeting the salesman is requested to go to another city before he returns to his base. The salesman sends his updated schedule to the STB agent. The agent realizes that the salesman will be able to watch the program only on the next day and therefore that it has plenty of time to conduct the negotiations. In this case, the deadline of the agent has changed and therefore it has to decide how it can use this information to secure a better deal for its owner. Specifically, it can decide whether to conclude the negotiations at this point and resume after the mail agent has finished its operations, or to switch back to using a stubborn approach with the satellite broadcaster or even resume negotiations with the terrestrial broadcaster in the hope of getting the broadcaster to concede over a prolonged period of time.

In this scenario we have seen that the environmental conditions are dynamic and that the agent has to adapt its negotiation strategies to these changes in order bargain effectively. Specifically, the agent in this scenario has to adapt to changes in resource availability, reservation price and deadlines.

1.2.2 Scenario 2

Here we consider a scenario in which the opponent's strategies change (see figure 1.2). A regional manager for a large electrical retail group needs to travel to attend a meeting in Edinburgh. Again located within her Personal Area Network are various devices for communication and work. They include a PDA, a laptop, an electronic wallet, and a cellular phone. The various devices are interconnected via wireless links. She has

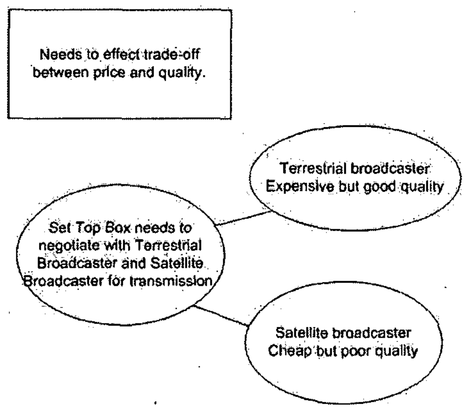
Needs to effect trade-off between price and quality.
Terrestrial broadcaster Expensive but good quality
Set Top Box needs to negotiate with Terrestrial Broadcaster and Satellite Broadcaster for transmission
Satellite broadcaster Cheap but poor quality

FIGURE 1.1: Scenario 1: Trade-off between price and quality

to negotiate with a local network operator in the area for guaranteed delivery of some services. The agent initially knows nothing about the operator and also the operator can change its behaviour during the course of the negotiations. In order to strategically bargain with this unknown opponent, the agent has to learn the pattern of behaviour of the operator. This type of strategic play is quite common in game theory [Nash, 1950] and several researchers have addressed this issue. However it is important to note here that the agents are also continuously changing their behaviour in response to changes in the environment. Both agents are thus functioning in a dynamic environment, where, as described in Scenario 1, their reservation prices and deadlines can change during the course of the negotiation encounter. This can be compared to a situation in a game where not only does the agent have to deal with a strategic opponent, but also has to deal with changes in the rules of the game. It is this aspect of adapting both to changes in the environment and the opponent's behaviour that separates our research from other work in the literature in this area.

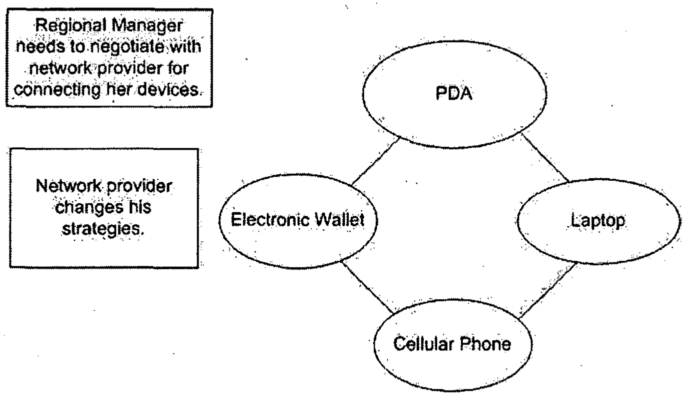
Regional Manager needs to negotiate with network provider for connecting her devices.
Network provider changes his strategies.
PDA
Electronic Wallet
Laptop
Cellular Phone

FIGURE 1.2: Scenario 2: Adapting to changes in strategy

1.2.3 Scenario Commonalities and Differences

In both scenarios it can be seen that the state of the system changes dynamically (e.g., in Scenario 1 the negotiation parameters like deadlines, budget constraints and resource availability change and in Scenario 2 the opponent's strategies change in response to changes in the environment). These type of conditions are not restricted to the mobile communications domain, but, they are also prevalent in many other multi-agent systems. Thus the agents have to develop strategies in the absence of complete information about the opponents and the state of the system. Also, as can be seen in both scenarios, since the agent cannot predict all the situations that it might encounter, it will have to *learn* through experience to map appropriate actions to situations (e.g. in Scenario 1 the agent does not know the salesman's schedule in advance or when another agent would need its communication space). Also, in Scenario 1 the agent is unaware of the salesman's preferences which change because of budget constraints, while in Scenario 2 the agent is unaware of how the network operator will change its strategies. Now, since the opponent's strategy can change the agent will have to develop a model of the opponent's behaviour and evolve counter strategies in order to be effective in the negotiations.

When taken together, these conditions motivate the requirements of our model and in the next section we detail them more formally.

1.3 Requirements for our Negotiation Model

In our work we wish to build a bilateral negotiation model for environments whose characteristics evolve over time as a result of the interactions between its individual components in general and for the future multi-agent systems like the mobile communication domain in particular. From the scenarios described in Section 1.2, we can extract a number of general characteristics of agents in such systems, as a motivation for the requirements of the negotiation model. In general, agents working in the multi-agent domain have to be [Jennings et al., 1995]:

1. responsive to their environment, i.e., they must sense changes in the environment and effectively respond to them,
2. proactive, i.e., they must be able to take the initiative in accomplishing their tasks and
3. social, i.e., they must be capable of interacting with their environment and other agents in it.

Thus in the multi-agent domain one of the key defining characteristics is that of dynamism. Moreover, such dynamism means it is impossible to predict in advance all the situations that an agent might encounter. Thus it is not possible to specify effective strategies for negotiation at the start of the negotiation process. Also, if an agent is negotiating with another adaptive agent, then in order to achieve a better bargaining result and to avoid being exploited by an intelligent opponent it becomes necessary to be able to predict and adapt to changes in the opponent's behaviour. Given all of this, our negotiation model must:

1. Allow an agent to negotiate autonomously, meaning that it should have the capability to make appropriate decisions on its own.

2. Allow an agent to negotiate in the absence of complete information about the opponent and the environment.

3. Cope effectively when the agent's own negotiation preferences are subject to change due to changes in the state of the environment (for instance resource availability changes or the time available for negotiations changes) during the course of the encounter. This implies that the strategies are non-stationary and need to evolve with time and the state of the environment.

4. Cope effectively when negotiation parameters like deadlines and reservation prices change during the course of the encounter.

5. Cope effectively when the opponent's negotiation behaviour changes during the course of the encounter.

6. Allow an agent to reach a suitable agreement before the resources and time available for negotiations are exhausted.

Having now listed our requirements in detail, we proceed to detail our research contributions.

1.4 Research Contributions

We have developed an automated negotiation model that adapts to dynamic environments and adaptive opponents by using machine learning techniques. Specifically, by developing learning based algorithms for non-stationary scenarios we significantly extend the state of art in the area of adaptive negotiation theory. We also contribute to the domain of multi-agent systems by developing this automated negotiation mechanism.

In more detail we have used, game theoretic ideas to provide a well established mathematical framework to study negotiations. This was chosen as the conceptual basis for this work because it provides us with several formal methods to study interactions between agents. Also adaptation in multi-agent systems has been studied extensively using the game theoretic format (see Chapter 2 for more details). Therefore in our negotiation model we use a combination of game theoretic ideas and machine learning concepts to describe the negotiation process. Specifically the agents in our model sense changes in the environment, learn to classify the opponent's behaviour and learn a strategy that maximizes a reward signal for a specific state of the environment and an opponent's perceived strategy. In particular, we use Reinforcement Learning (RL) techniques to learn an optimal mapping between states and actions by estimating the value of the future reward that they would obtain by choosing a specific action from a given state and Bayesian Learning (BL) techniques that use the history of the negotiation process to model an opponent's behaviour and thus develop counter-strategies.

In more detail, the RL algorithm does not specify the best policy to use for a given situation, but rather allows the agent to learn this state-action mapping autonomously (thus satisfying our requirement 1). The changes in the negotiation preferences are captured in the state definition and the agent learns to adapt to these changes. Also the utility function of the agent is represented by a reward signal which can change with time and thus the agent learns to cope with non-stationarity (thus satisfying requirement 3). Using our algorithm the agent also adapts to changes in negotiation parameters like deadlines and reserve prices thus satisfying requirement 4). Finally the agent learns these strategies by assuming only a probabilistic knowledge of the system and therefore it copes with absence of complete information about the environment and its opponents (thus addressing requirement 2). Using this algorithm, the negotiation allows an agent to reach an agreement before its deadline is up (thus satisfying requirement 6). Then using the BL algorithm the agent learns to model its opponent's behaviour and develops strategies in response (thus satisfying requirement 5).

In so doing, this work extends the state of the art in the following ways:

- We develop and empirically evaluate the first algorithm to apply learning techniques to non-stationary negotiation scenarios in order to adapt to changes in negotiation preferences, parameters and environmental changes in the absence of complete information about the environment.
- We develop and empirically evaluate the first algorithm to apply learning techniques to develop strategies in response to non-stationary opponents.
- We theoretically prove that these algorithms converge to the optimal solution over repeated negotiations.
- We provide the first detailed comparison between RL and BL techniques as they relate to the automated negotiation in multi-agent domains. Our comparison identifies that RL techniques are more suitable for situations in which the environment is very dynamic and that BL techniques are more suitable for developing strategies against adaptive opponents.

To summarize, we develop two efficient adaptive algorithms (one using reinforcement learning and the other using Bayesian learning) to adapt to changes in the environment and to changes in opponent's behaviour.

The following papers have been published based on this work:

- V. Narayanan and N. R. Jennings (2005) "An adaptive bilateral negotiation model for e-commerce settings" Proc. 7th Int. IEEE Conf on E-Commerce Technology, Munich, Germany 34-39.
- V. Narayanan and N. R. Jennings (2006) "Learning to negotiate optimally in non-stationary environments" Proc. 10th Int. Workshop on Cooperative Information Agents, Edinburgh, UK, 288-300. *Winner of Best Paper Award*

1.5 Thesis Structure

The remainder of this thesis is organized in the following manner.

- In Chapter 2 we present an in-depth review of the main negotiation models in the literature that are relevant to our work and machine learning techniques used in game theory and for negotiations.
- In Chapter 3 we describe the main elements of our negotiation model and cast our problem as a RL problem. We also present our adaptive algorithm for negotiations and our empirical results.
- In Chapter 4 we describe how we use BL techniques to adapt to changes in the opponent's behaviour. We also present our adaptive algorithm and our empirical results.
- In Chapter 5 we present an in depth comparison of RL and BL techniques as they relate to the automated negotiation context.
- In Chapter 6 we conclude and outline the directions for further avenues of research.

Chapter 2

Literature Review

Negotiation is the key to resolving conflicts among self-interested individuals who share a common environment and have personal goals to achieve. This is especially the case in trading scenarios as explained in Chapter 1. As such, it has been studied in a number of different contexts in the past (including [Nash, 1950], [Rosenschein and Zlotkin, 1994]). However the study of negotiations among software agents is relatively new. Specifically, the study of agent negotiation has become increasingly popular after the recent advent of digital marketplaces and other trading agent systems [Chavez et al., 1997]. Now, we are specifically interested in studying how trading agents can evolve strategies to buy and sell commodities in dynamic multi-agent systems like the mobile communications domain and in digital marketplaces (see Chapter 1). In order to do this we need to study previous models of negotiation among agents in non-stationary environments. Here, by non-stationary we mean that negotiation parameters like deadlines and prices and also the negotiation strategies used in the negotiation process can change during the course of the negotiations. Thus in this chapter we present a detailed survey of the main negotiation models discussed in the literature in order to provide a formal background for our work.

In more detail, section 2.1 provides the essential background for this work, then particular details of traditional bilateral negotiation models are discussed in section 2.2. We then consider the machine learning background in section 2.3 and we give an overall assessment of the key learning techniques used in developing these algorithms and highlight what components can be re-used in this research and which components need to be developed anew. In Section 2.4 we describe some specific adaptive negotiation models and in Section 2.5 we summarize.

2.1 Negotiation Background

Given the importance of negotiation techniques in trading environments, there has been a considerable body of work in this area and the aim here is to give a brief description of the evolution of negotiation theory, before starting to look at the models that are of greatest relevance to our work.

Negotiation was regarded as the classical bargaining problem in economic theory and was first treated mathematically in [Neumann and Morgenstern, 1944]. In this work, negotiation was formally described as a *zero-sum game* in which one player's gain is the other player's loss. Following this, John Nash presented a new treatment of this problem of bargaining between individuals and proposed an alternative solution, based on the intuition that the players in this game are not adversaries and that during the negotiations they try to find a common ground where both their goals would be satisfied (see [Nash, 1950]). In particular he argued that the solution should satisfy certain properties:

1. Players are rational: They wish to maximise their utility and have prefect information about the game which means that they know their own utility functions, as well as the utility functions of their opponents.

2. Efficiency: No player can deviate from this solution without loss of utility.

Following this conceptualisation of a game, he envisaged an agreement to the problem as being one from which neither player will have any incentive to deviate. These types of games are called *cooperative games* in which the agents' interests are neither completely opposed, nor completely coincident [Nash, 1953]. Then [Nash, 1951] extended this work to describe other games in which he assumed that each player acted independently and without collaboration and termed them *non-cooperative games*. These first models, however, assume that the players have *complete information* about the game which implies that they know each other's payoff functions. In more detail, these are the functions that define the monetary gain that the agents could get from the negotiation process. In any negotiation encounter the agents aim to maximise their payoff functions. However, as detailed in our requirements, in most multi-agent settings the assumption of *complete information* is clearly invalid [Kraus, 2001]. Also an important facet of our work, related to the conditions prevailing in many multi-agent domains, is the ability to adapt to changes in the environment. Again this is not considered in these earlier models. Nevertheless, the ideas described in Nash's models form the foundation of all later work in this field (see Section 2.2 for further details) and so our work falls under the broad range of non-cooperative games.

Recent research in negotiation among agents within a multi-agent system has focussed on addressing some of the aforementioned shortcomings. Amongst them [Sycara, 1988; Zlotkin and Rosenschein, 1989] were the first to apply formal game theoretic notions of negotiation to the artificial intelligence and multi-agent domain. In [Sycara, 1988] a case based reasoning method is used to resolve conflicts among agents. This reasoning process is called a "Persuader" method and uses an iterative argumentation process to persuade the agents to change their intentions in order to come to an agreement. Zlotkin and Rosenschein [1989] formally modelled the negotiation process among agents and considered how strategies could be developed for situations when the agents have complete information about the negotiation process and also for encounters when some information (like negotiation parameters or utility functions) is unknown to the negotiating agents. These types of encounters are classified as negotiations with incomplete information.

By this time, several researchers had discovered the rich potential of applying game theory to solve research problems in multi-agent systems. Shoham et al. [2007] provide a good review of the main work in this area. Littman [1994] pioneered much of this work in his paper where he explicitly cast interactions between agents as a zero-sum Markov game and obtained Nash equilibrium solutions using Reinforcement Learning (RL) techniques. Then Hu and Wellman [1998] extended this idea to general-sum games [1]. Although their work does not talk about negotiation explicitly, they have demonstrated how game theory and machine learning can be used in conjunction to study agent interactions and thus their work is closely allied to our own.

The above two models assume that the agents jointly learn an equilibrium solution. However in their work no reference is made to changes in the environment which is one of our key requirements. Following from this, [Claus and Boutilier, 1998] in their work presented an algorithm in which agents individually learn an equilibrium for general sum games and in [Weinberg and Rosenschein, 2004] the agents learn a best response strategy against adaptive opponents in non-stationary environments. However these models assume that the agents have complete information about each other's payoff functions which does not meet our requirement 2. Next, in order to model opponent's behaviour and thus negotiate strategically in multi-agent environments Binmore and Vulcan [1997] used Bayesian Learning techniques (BL). Zeng and Sycara [1998] also used BL techniques to develop an adaptive agent negotiation system called BAZAAR. However both these models are descriptive in nature and do not consider adaptive opponents. By this, we mean that, although the agents are assumed not to know the distribution of players' strategies and their negotiation parameters, it is assumed that this distribution does not change over time. Now, this is a serious shortcoming in the types of open environment in which multi-agent systems are often deployed and is something

[1]Games in which the agents share in the profits

that we wish to rectify in this work.

More sophisticated models, like [Kaelbling et al., 1998] developed a new concept of Partially Observable Markov Decision Processes or *POMDPs* in which the agents can only see part of the state space in order to address the issue of incomplete information available to agents. They then developed algorithms for solving them. In the same vein, Emery-Montemerlo et al. [2004] extended this concept to Partially Observable Stochastic Games (*POSGs*) and outlined a solution procedure using Bayesian learning techniques. However these models make a number of assumptions about the stationarity of the environment and are, therefore, of limited use in our work.

Finally, other models like [Bui et al., 1999] and [Oliver, 1996] have used other forms of learning like Radial Basis functions and Bayesian classifiers. But again these models are geared more towards working in incomplete information settings and learning the opponent's preferences than looking at adapting to dynamic changes in the environment and opponent's strategies (all of which are central to our requirements 2, 3 and 4).

Given this background, in order to satisfy our requirements we adopt ideas from two main branches of machine learning (RL and BL), but crucially we will assume that both the negotiation parameters and the opponent's behaviour are subject to change during the course of the negotiation process. That is, the negotiation process is non-stationary. From this standpoint, we will attempt to learn the patterns of variation, while dealing with an adaptive opponent and a dynamic environment.

For the sake of completeness, we also introduce here some of the main heuristic models in this field. Here, by a heuristic model we mean a negotiation model which uses heuristic search techniques like evolutionary algorithms (see [Holland, 1992]) to identify good strategies. In this context, [Kraus, 2001] is an important work that specifically considers strategic negotiation among agents. It recognises the need for adaptive negotiation within an agent context and gives a broad overview of the possible solutions. In this work she argues that heuristic methods are more suitable for developing adaptive negotiation strategies in multi-agent systems than pure game theoretic techniques. This is because in game theory, essentially, the negotiation problem is formulated as a system of linear equations and the solution to the negotiation problem is the solution to this system. But, as we introduce the complications of dynamism and incomplete information settings into the problem, it becomes non-linear and therefore increasingly harder to obtain an analytical solution. In contrast, heuristic methods are iterative search methods, where at each step, the heuristic algorithm searches through a number of possible solutions and selects a good solution according to certain criteria and is therefore not restricted by the linearity of the problem.

In this vein, we find in [Matos et al., 1998] a heuristic evolutionary technique for developing negotiation strategies that adapts to changes in the environment and the opponent. Binmore and Samuelson [2001] also talk about integrating heuristic evolutionary methods with formal game theoretic concepts. However, generally speaking, these heuristic approaches assume that the agents have complete information about their own utility functions and further that these utility functions do not change with the state of the system or with time (i.e., they are stationary). Because of these assumptions the strategy selection process is also fixed and does not change with time. This is against requirement 3 of our model. However for future research it might be useful to examine if in the evolutionary approach, the criteria for selecting strategies can be made dynamic in order to satisfy our requirements. Therefore for now, we need a different approach in which the agents ascertain the current state of the system and learn to evolve appropriate strategies.

Finally, we examine work which looks at agent negotiation from the perspective of developing a mechanism for generating offers [Faratin et al., 1998]. In this body of work, the authors are not attempting any kind of learning or adaptation. Rather, the focus is on formally defining negotiation parameters, developing functions for generating offers and determining the possible ways in which an agreement can be reached. We use these definitions and the form of the offer generating function in our work, therefore we review this work in Section 2.2. Fatima et al. [2004] then extended this to develop Nash equilibrium strategies for multi-agent negotiation in incomplete information settings. Since incomplete information scenarios are of interest to us we describe this work briefly also in Section 2.2.

Having completed this introduction to the fundamental concepts of negotiation theory, we now move onto looking at these models in more detail. The rest of this chapter is structured in the following manner. First in section 2.2, we look at traditional game theoretic models of bargaining and negotiation in the literature. In section 2.3, we consider work in machine learning with special reference to RL and BL techniques and explain why we think these techniques are suitable for meeting our requirements. In section 2.4 we look at some models that have used adaptive negotiation methods and finally, in section 2.5, we summarize.

2.2 Bilateral Negotiation Models

In this subsection, we give a detailed review of game theoretic bilateral negotiation models for multi-agent settings.

As discussed in section 1.1, while designing any negotiation model there are five main issues to be addressed:

- What negotiation protocol to use?
- What are the issues or objects that are being negotiated?
- What are the negotiation parameters?
- What are the agents' negotiation preferences?
- What reasoning process the agents employ to make decisions? [Faratin et al., 1998].

There are many protocols that have been used for trading. These include the alternating offers protocol (see section 1.1), single-sided auction mechanisms like English and Dutch variations [Milgrom, 1982], and, more recently, double auctions [Wurman et al., 1998]. However since we wish specifically to model bilateral negotiation (see Chapter 1) and since auctions are mainly used for trading between multiple agents, we will use the alternating offers protocol (see section 1.1). Moreover, since we are looking at trading agents negotiating over the price of a commodity, we focus only on single issue negotiation. However, both single issue negotiation and negotiation with multiple agendas have been studied extensively in the literature [Fatima et al., 1997, 2002; Faratin et al., 1998]. Thus we will cover these models in more detail when we discuss the strategies that are used in agent negotiation. Now, however, we will proceed to give an overview of the development of negotiation theory and describe in detail some of the key work in this area.

As discussed in section 2.1, one of the earliest works that considered strategies in bargaining was due to Nash. Here, the notion of a *strategy, s_i*, was defined as the action alternatives that an agent has in the game and the *payoff function, p_i*, as the reward that i would get at the end of the game. Nash then went on to define the important notion of an *equilibrium point* as a strategy profile (a collection of strategies in which each agent is represented by a single strategy) of the agents in which each strategy is a best response of one agent to the strategies of the others. Thus he proposed that no agent would have any incentive (as defined by the *payoff function*) to deviate from the *equilibrium point*. Formally a strategic game can be defined as [Fudenberg and Tirole, 1991]:

1. The set of players, $i \in I$ where I is the finite set $1, 2..., I$.
2. The strategy space $S = (s_1, s_2, ..., s_i)$ for each player i.
3. The utility or payoff functions $u(S)$ for each profile or set of strategies $(s_1, s_2, ..., s_i)$ where each $s_i \in S$

In this definition of a strategic game the strategy space is called a *pure-strategy space.* Fudenberg and Tirole [1991] have also described an alternate strategy space called the *mixed-strategy space.* The mixed-strategy space is a probability distribution over the space of pure-strategies and for mathematical ease of analysis it is often more convenient to deal with the mixed-strategy space. In our model (see section 4.1) we will use a mixed-strategy space to describe the agent's strategy space. Many later models, like [Faratin et al., 1998; Kraus, 2001; Fatima et al., 2004], use this notion of strategic games to build negotiation models for multi-agent systems. In more detail, [Faratin et al., 1998] study strategic negotiation on multiple issues between two agents. Specifically, this model is a bilateral negotiation model and uses the alternating offers protocol for conducting negotiations. The agent (a) has a deadline (T^a) before which it should complete the negotiations and has a reservation price (RP^a) which represents the price below which the agent is not allowed to concede. The key contribution of this work is the development of parameterized functions called Negotiation Decision Functions (NDFs) to generate offers. The NDF for agent a can be represented as:

$$f^a(t) = k^a + (1 - k^a)(min(t, T^a)/T^a)^{1/\psi} \tag{2.1}$$

Here k^a is a pre-defined constant which determines the initial offer of agent a and T^a represents its negotiation deadline. As can be seen, equation 2.1 actually corresponds to a family of NDFs defined by the parameter ψ. Thus by varying ψ, the agent can exhibit a variety of behaviour patterns. These are given by:

1. *Conceder*: When $\psi > 1$: The agent quickly reaches its reservation value.
2. *Boulware*: When $\psi < 1$: The agent maintains its initial offer until the deadline is almost reached and then concedes quickly.
3. *Linear*: When $\psi = 1$: The agent increases its price in a steady fashion.

These three patterns only represent a broad classification of the agent's behaviour. By assigning specific values to ψ we can do a much more fine grained analysis of the agent's behaviour, although the behaviour pattern would still belong to one of the three main prototypes. In short, ψ determines whether the agent will linearly or non-linearly approach its reservation price. The agent thus generates offers using this NDF. In this case, the offer from agent a to its opponent agent $\hat{a}$ is represented as $p^t_{a\to b}$ and is given by:

$$\begin{aligned} p^t_{a\to\hat{a}} &= IP^b + f^b(t)(RP^b - IP^b) for\ buyer\ b \\ &= RP^s + (1 - f^s(t))(IP^s - RP^s) for\ seller\ s. \end{aligned}$$

Here IP^b, RP^b, IP^s and RP^s are the initial and reservation prices of the buyer and seller respectively and $f^a(t)$ represents the NDF of agent a. Now, the value function or utility function associated with each agent, a, is given by:

$$V^a.: (Offers \times Times) \rightarrow \Re \qquad (2.2)$$

Here the set *Offers* is the set of all prices and the set *Times* is the set of positive integers. Now the agents use the utility function to evaluate the offers that they receive. The agent b rates the offer using this utility function and if the utility from this offer is greater than the utility from the counter-offer that b proposes to make at the next time instant, b accepts $a's$ offer. This process of evaluating an offer and responding with a counter offer is called an action, A. This action, taken at time t', where $t' > t$, is represented for the agent, a, as $A^a(t', p^t_{b\to a})$ and is defined as:

$$\begin{aligned} A^a(t', p^t_{b\to a}) &= Quit\ if\ t' > T^a\ where\ T^a\ is\ agent\ a's\ deadline \\ &= Accept\ if\ V^a\ from\ p^t_{b\to a} \geq V^a\ from\ p^{t'}_{a\to b} \\ &= p^{t'}_{a\to b}\ otherwise. \end{aligned}$$

The process of making offers and counter-offers continues until either the agent's deadline passes or an agreement is reached. If the deadline is reached before an agreement on all the issues is made then the negotiation ends in a conflict. Overall then, the NDF governs the process of generating offers and counter-offers.

This model thus gives us a concise mathematical function which generates a non-decreasing (or non-increasing in case of the seller) sequence of offers and the ability to control the speed of convergence of this sequence to the RP. This feature has made the NDF approach very popular among researchers in the field. For example, [Fatima et al., 2004; Jennings et al., 2001; Kraus, 2001; Paurobally et al., 2003] have all used different forms of the NDF to analyse various negotiation scenarios. Also, in this work, the authors have suggested ways of using NDFs to generate offers that adapt to changes in time and resource availability by controlling the value of the parameter ψ. Thus using the NDF appropriately, the agent can exhibit responsive, adaptive behaviour by varying the values of ψ. This provides us with the motivation to use the NDF to generate offers in our model. But this model does not implement these strategies or examine ways of choosing appropriate strategies to suit the situation. Thus these strategies are static. Nevertheless, this work is still very significant because it develops the important notion of an offer generating function, even though it does not satisfy any of our requirements.

We next consider the model developed by [Fatima et al., 2004]. They were the first to extend Faratin et al. [1998]'s model to study negotiation in incomplete information scenarios. The main concept introduced here is the notion of an agent's *information* *state*, which is denoted as I, and describes the information that the agent has about its opponent. Here it is assumed that each agent in the negotiation process has a reservation limit and a deadline. Together, these define the agent's negotiation parameters. It is also assumed that the agent is aware of its own negotiation parameters, but is unaware of the parameters of its opponents. The information state is formally represented as $I =< F^a, V^a >$, where F^a represents the agent's own parameters and V^a represents the information the agent has about its opponent's parameters. Optimal strategies are determined for the buyer and the seller on the basis of the information state of the agents. Specifically, a strategy is defined by the 4-tuple $< IP^a, RP^a, t^a, NDF^a >$ where IP^a is the initial price of agent a, RP^a is the reservation price of agent a, t^a is the deadline and NDF^a is the negotiation decision function of agent a. This strategy is optimal if the agent derives maximum utility from it. Using the basic framework of this model, the authors have analyzed bilateral negotiations for different incomplete information scenarios. These scenarios can be classified into two categories:

- *Symmetric information scenarios* in which the agent has the same information about the parameters of its opponent, as the opponent has about the agent's parameters.

- *Asymmetric information scenarios* in which the agent and its opponent have information about different parameters of one another.

The authors have solved a non-linear mathematical program involving the negotiation parameters (where some parameters are unknown) and the payoff functions to determine a Nash equilibrium in the different scenarios. Thus they determine the value of parameter ψ and therefore the strategy — Conceder, Boulware, Linear — that yields a Nash equilibrium. This work is important in this context because it deals with the issues of incomplete information in negotiation and motivates our research to the next level where we build models for both incomplete and dynamic negotiation scenarios.

Next, we review some of the preliminary models that first looked at dynamic negotiation within multi-agent systems. Among these, the work of [Kraus, 2001] is probably the most significant. In this context, we present the important aspects of the discussion in this work to deal with negotiation in multi-agent environments. Here a detailed discussion of various strategic negotiation models for multi-agent systems can be found. The author adopts a formal game theoretic approach to formulate negotiation problems in different systems. Then, having demonstrated the difficulty of obtaining analytical solutions to these problems within this framework in this domain, suggests heuristic techniques for developing negotiation strategies. The negotiation strategies developed in this work have been applied to a number of problems including data allocation, resource allocation and task distribution in computer systems. This work, by formally casting these problems in multi-agent systems as negotiation problems, highlights the need to develop negotiation strategies that adapt to changes in the environment and also to changes in the opponent's behaviour. The work therefore serves as an important point of reference for us for further development of negotiation in multi-agent systems. In the same vein, as a first step towards developing models that adapt to changes in the environment [Faratin et al., 1998] have suggested ways in which the NDF can be modified to deal with important changes in the environment such as resource availability and changes in the behaviour of the agent when the deadline is approaching. In more detail, the various different types of NDFs can be classified as:

1. *Time-Dependent*: where agents concede more rapidly as the deadline approaches. There are two families of time dependent negotiation decision functions.

   - Polynomial: $\alpha_j^a(t) = \kappa_j^a + (1 - \kappa_j^a)(\frac{min(t,t_{max})}{t_{max}})^{\frac{1}{\beta}}$ which defines the family of functions used in [Fatima et al., 2004].

   - Exponential: $\alpha_j^a(t) = exp((1 - \frac{min(t,t_{max})}{t_{max}})^{\beta} * ln\kappa_j^a)$

   where κ_j^a is an arbitrary constant and t_{max} represents the agent's deadline. Depending on the value of β, these functions can be classified as Boulware, Conceder and Linear functions (as described earlier in this section).

2. *Resource-Dependent*: where agents change their negotiation strategies based on resource availability. Here the availability of resources, like bandwidth for communication, computational power and time for communications [Paurobally et al., 2003], varies dynamically. This can be modelled in two ways:

   - By making the deadline dynamic (i.e., t^a_{max} is not fixed, but is a function of time.)
   - Estimate the available resource and make the negotiation decision function depend on this estimated value. Here the negotiation decision function is represented as: $\alpha^a_j(t) = \kappa^a_j + (1-\kappa^a_j) * exp(-resource(t))$. Here *resource(t)* defines how the resource varies with time.

3. *Imitative*: where agents imitate their opponents' behaviour. These tactics are adopted by the agents when sufficient resources are available for negotiation. Three types of imitative behaviour are discussed:

   - Relative Tit-For-Tat where the agent imitates the action taken by its opponent $\delta \geq 1$ steps ago.
   - Random Absolute Tit-For-Tat where the agent implicitly imitates the opponent's behaviour.
   - Averaged Tit-For-Tat where the agent computes the average of percentage of changes of its opponent's negotiation history in a specified time window.

This model addresses the issue of dynamism in the negotiation environment by suggesting different NDFs for generating offers depending on the availability of time and resource. It also suggests ways in which the agent can adapt to changes in its opponent's behaviour. But, this model assumes that the pattern of resource availability is known to the agent and, therefore, it does not meet our requirement of functioning in the absence of information. Moreover, the authors have only suggested a method of adapting to the environment and have left the implementation of this model to future work.

In summary, in this section we have reviewed the work which has shaped the development of the theory of automated negotiation in multi-agent systems. These models explain the fundamental principles involved in designing negotiation strategies for multi-agent systems. However they do not consider adaptivity. Therefore, in the next section we consider RL and BL techniques and examine how they can be used to address this issue.

2.3 Machine Learning Background

We now move onto describing the mathematical underpinnings of RL and BL techniques, before we detail the most important work done using game theory and machine learning to analyse interactions between agents in multi-agent settings in the next subsection.

2.3.1 Reinforcement Learning in Multi-Agent Systems

The reinforcement learning methodology is an unsupervised machine learning technique in which an agent learns to achieve its goals by interacting with its environment. The main principle of this technique is to guide the agent through a progression of states in a manner that maximizes its utility. The negotiation process, with its protocol of alternating offers, can naturally be broken down into episodes in which agents either make or respond to offers. It is therefore easy to describe the negotiation process in terms of discrete states and define actions for the agents that would enable them to move from one state to another. This problem thus lends itself in a straightforward manner to being formulated in the reinforcement learning framework. In [Sutton and Barto, 1998], [Kaelbling et al., 1996] we can find a detailed analysis of the reinforcement learning (RL) technique and its applications. However, here, we just present the features of this learning mechanism that are necessary to meet our requirements.

Generally speaking, amongst learning techniques, reinforcement learning, and in particular Q-learning, is often used to study agent interactions in multi-agent systems, since it does not need a model of the domain for learning and can be used online [Hu and Wellman, 1998; Sutton and Barto, 1998]. In this case, the agent learns an optimal policy of mapping actions to states by maximizing a series of numerical reward signals. Adopting these learning methods, several researchers have used the stochastic game framework for multi-agent reinforcement learning and have developed solution techniques like Nash Equilibrium [Hu and Wellman, 1998; Littman, 1994] and best response strategies [Weinberg and Rosenschein, 2004]. Others, like [Fudenberg and Tirole, 1991], have used *fictitious play* techniques to analyse learning in games. However, in our case, we are not trying to model learning in multi-agent systems using game theory, but more specifically, we are trying to develop negotiation techniques for multi-agent systems for which learning is necessary. Also, these approaches work well when the objective of the learning process can easily be described as a state-action mapping policy. However the desired outcome in a negotiation process can be defined in a number of ways, such as reaching an agreement within the deadline, concluding the deal at a favourable price and so on. In such situations, the gains that can be obtained by participating in the process, cannot readily be translated into numerical reward signals and, therefore, we have to modify the existing reinforcement learning algorithms. Moreover, these (at least for proving convergence) rely on the assumption that the underlying environment is *stationary* (i.e., the strategies of the opponent do not change over time, they are simply unknown). Now, this is clearly not the case in many realistic negotiation encounters, so we need to significantly extend these algorithms for our type of negotiations. Specifically, for this context we need an approach that in conjunction with learning about the environment also allows the agents to continuously update their knowledge of the negotiation process so that their decisions are in tune with the current state of the system.

Specifically, in RL the decision-maker is called an *agent* and everything else that it interacts with, including its opponent, is termed the *environment*. The agent in any RL problem has a goal to achieve. The agent interacts with its environment in discrete time steps $t = 0, 1, 2, ...$ in order to do this. At each time step the agent receives some information about the state of the system, $s_t \in S$ where S represents the set of all possible states. At each state the agent has some actions available to it. This set of available actions from state s_t is denoted by $A(s_t)$. In the next time step the agent receives a *reward*, $r_t \in \Re$, as a consequence of its action and finds itself in a new state s_{t+1}. A *return function* is defined as the sum of rewards received by a single agent at each time step and is denoted by $R_t = r_{t+1} + r_{t+2} + ... + r_T$ where T is defined as the final time step of the negotiation process. This return function is represented as a probability distribution. The objective of the agents is to maximize the *expected return*. At each time-step the agent maps a state to an action using a *Policy* (denoted by π_t). The *Policy* defines a mapping between a state-action pair (s, a) and the probability of selecting action a from state s. The probability of selecting an action a from a state s is defined as $\pi_t(s, a) = Pr(a_t = a|s_t = s)$. Now in order to use this formulation for decision making in a RL problem we need the Markov property. We now formally define the Markov property and give a detailed description of the Markov Decision Process [Howard, 1960].

Intuitively, a process is Markovian if and only if the state transitions depend only on the current state of the system and are independent of all preceding states. Formally, the sequence of random variables $\{X_n, n = 0, 1, 2, ...\}$ is defined to be a Markov process iff their conditional probability density function, P, satisfies the following relationship [Howard, 1960]:

$$P\{X_n|X_1, X_2, ..., X_{n-1}\} = P\{X_n|X_{n-1}\} \qquad (2.3)$$

Then a process that uses this property of the state space to implement all decisions, based on a reward scheme, that need to be made within this space is called a Markov Decision Process. Now, although in real-world scenarios state transitions are rarely purely Markovian, several of them approximately satisfy the property and therefore it is reasonable to make this assumption for such situations [Sutton and Barto, 1998]. For our model we will assume the state transitions to be Markovian in nature. Having defined the Markov property, we are now ready to describe the process that uses this property for decision making.

Reinforcement Learning tasks that satisfy the Markov property are called Markov Decision Processes or *MDPs* [Kaelbling et al., 1996]. An MDP is formally defined as:

- a discrete state space S
- a set of discrete actions A
- a reward function $R: \ S \times A \rightarrow \Re$
- a probabilistic state transition function, $T : \ S \times A \rightarrow [0,1]$, $T(s, \ a, \ s')$ is defined as the probability of making a transition from state s to state s' using action a.

If the state and action spaces are finite then the MDP is termed a finite MDP. In more detail, a finite MDP is defined by the state and the action spaces, the probability of transition from one state to another and the reward function associated with every transition. The probability of transition from state s to state s' by choosing action, a, is called the transition probability and is given by:

$$T^a_{ss'} = T(s_{t+1} = s' | s_t = s, a_t = a) \tag{2.4}$$

In the RL context, as stated earlier in this section, there is a reward associated with every state transition. This reward is determined by a probability distribution and hence we define the expected value of reward associated with the state transition $s \longrightarrow s'$ by choosing action a as:

$$R^a_{ss'} = E(r_{t+1}|s_t = s, a_t = a, s_{t+1} = s') \tag{2.5}$$

When taken together, equations (2.4) and (2.5) completely describe the Markov Decision Process and hence the RL problem.

Having defined the transition probabilities and the expected reward associated with each transition, we are now ready to define functions that will evaluate the value of a state s under a policy π, in terms of the expected return (sum of rewards obtained in t time steps) function. This can intuitively be defined as the return obtained by starting in state s and following a policy π thereafter. The **state-value** function for policy π is formally defined as:

$$V^{\pi}(s) = E_{\pi}(R_t|s_t = s) = E_{\pi}(\Sigma_{k=0}^{\infty} r_{t+1+k}|s_t = s) \tag{2.6}$$

Next we define the value of adopting action, a, from state, s, and thereafter following a policy π. This is termed the **action-value function** and is formally defined as:

$$Q^{\pi}(s,a) = E_{\pi}(R_t|s_t = s, a_t = a) = E_{\pi}(\Sigma_{k=0}^{k=\infty} r_{t+1+k}|s_t = s, a_t = a) \tag{2.7}$$

For our problem we will use these two value functions to provide the agents with an evaluation mechanism that will guide them towards achieving their objectives. Hence, we will principally concern ourselves with estimating these value functions using the RL theory. This theory also provides us with a method of evaluating the value of possible successor states of state s, under the policy π, from the value of state s [Sutton and Barto, 1998]. This relationship is encapsulated in the following equation:

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} T^{a}_{ss'}(R^{a}_{ss'} + V^{\pi}(s')) \qquad (2.8)$$

This equation is called the Bellman equation and this relationship will enable us to predict the value of successor states from a given state. This equation therefore gives us a powerful method of looking ahead in the negotiation process.

Using equations 2.6 and 2.7 we can define a partial ordering of policies. Here a policy π is defined to be better than policy π' iff $V^{\pi}(s) \geq V^{\pi'}(s) \, \forall \, s \in S$. An *optimal policy* is one that is better than or equal to all other policies. The optimal policy is denoted by π^* and the value function associated with this optimal policy is termed the **Optimal Value Function** and is given by:

$$V^*(s) = max_{\pi} V^{\pi}(s) \, \forall \, s \in S \qquad (2.9)$$

$$Q^*(s,a) = max_{\pi} Q^{\pi}(s,a) \, \forall \, s \in S \; and \; a \in A(s) \qquad (2.10)$$

The aim of any RL algorithm is to obtain optimal state and action value functions and hence the optimal policy π^*. In the literature, Dynamic Programming [Puterman, 1994] and Monte Carlo estimation techniques [Gilks et al., 1995] have been used extensively to determine the value functions. Here we will use Dynamic Programming (DP) techniques because Monte Carlo methods assume that rewards can be obtained only at the end of the negotiation process which is not suitable for our problem. DP however takes in account rewards that are given at all stages of the process. Therefore this is more suitable in our problem.

Now we briefly describe the procedure used to determine an optimal policy using a RL algorithm. There are three basic steps to learning the optimal policy in any RL algorithm. These are:

1. Policy Evaluation: Here we are concerned with determining $V^{\pi}(s)$ for an arbitrary policy π. We consider a sequence of successive approximations of the value function, $V^{\pi}(s)$, $V_0, V_1, ...,$ each function mapping every state $s \in S$ to $\Re$. The initial approximation V_0 is chosen arbitrarily and each successive approximation is obtained by using equation 2.8 as an update rule:

$$V_{k+1} = E_{\pi}[r_{t+1} + \gamma V_k(s_{t+1})|s_t = s] = \sum_a \pi(s,a) \sum_{s\prime} T^a_{ss\prime}[R^a_{ss\prime} + \gamma V_k(s\prime)] \tag{2.11}$$

The sequence has been shown to converge to V^{π}.

2. Policy Improvement: The main reason to determine $V^{\pi}(s)$ is to find better policies. Considering that we have determined $V^{\pi}(s)$ for an arbitrary deterministic policy π, we would next like to know if there are other state-action mappings that yield a higher reward than the current policy. To determine this, we choose an action $a \neq \pi(s)$ from state s and then follow the current policy. The value of taking such an action is given by $Q^{\pi}(s,a)$. In order to effect a policy improvement we simply choose from each state the action that appears best according $Q^{\pi}(s,a)$. The new policy π' is given by:

$$\pi' = argmax_a Q^{\pi}(s,a) \tag{2.12}$$

3. Policy Iteration: Using this new policy we determine the new value function corresponding to this policy $V^{\pi'}$. Using this $V^{\pi'}$ we then compute an improved policy π''. This process converges to the optimal value function and therefore an optimal policy in a finite number of steps in a finite MDP.

This process can schematically be represented as:

$$\pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 ... \xrightarrow{I} \pi^* \xrightarrow{E} V^*.$$

where E represents policy evaluation and I represents policy improvement.

Thus in any reinforcement learning problem these steps are followed to obtain an optimal policy. One of the most significant breakthroughs in the field of RL has been the development of the Q-learning algorithm (see [Watkins and Dayan, 2004] for more details) in which the optimal policy is learned using immediate rewards doing away with the necessity of knowing $T^a_{ss'}$ and $R^a_{ss'}$ and thus enabling model-free learning. In Q-learning we consider transitions from one state-action pair to another state-action pair and learn the combined state-action value function as opposed to learning the state-value or action-value functions separately. In this case the update rule is given by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)])$$

This update rule uses only the immediate reward obtained and thus can be used in situations where the dynamics of the environment are not known. This property of the algorithm makes it ideal for use in multi-agent environments as is satisfies the requirement of developing strategies in the absence of information about the environment or opponent. This is, therefore, the approach taken in this work to model changes in the environment.

Having described the general techniques used in RL to learn an optimal policy, we now move on to present a similar description of BL techniques.

2.3.2 Bayesian Learning in Multi-Agent Systems

Bayesian Learning, like reinforcement learning, is also an unsupervised machine learning methodology. However in contrast to RL which is model free, Bayesian Learning uses the history of a process in order to learn about it. In more detail, Bayesian analysis is often used to estimate the most probable underlying model for a random process, based on some observed data or inference [Bernardo and Smith, 1994]. We choose it here because it enables us to develop a model of the opponent's behaviour by learning from repeated interactions. This is different from the RL approach because using this method we do not explicitly model the changes in the environment. This method therefore will enable us to address requirement 5 (see section 1.4).

More formally, let $A_1, A_2, ..., A_n$, represent n random events. Then, we let $X_n, t = 0, 1, ...$ be the stochastic process we are trying to estimate. Each of these n events can be thought of as representing the hypothesis that the parameters of $\{X_n, t = 0, 1, ...\}$ belong to sets $T_1, T_2, ..., T_n$. Finally, we let the event B represent the set of observed data. Now, *Bayes rule* can be stated as:

$$Pr\{A_i|B\} = \frac{Pr\{B|A_i\} \times Pr\{A_i\}}{\sum Pr\{B|A_i\} \times Pr\{A_i\}} \tag{2.13}$$

where $Pr\{A_i\}$ is the prior probability of model A_i in the absence of any information, $Pr\{\ B|A_i\}$ is the likelihood that observation B was produced given that the model was A_i, and $Pr\{A_i|B\}$ is the posterior probability of the model being A_i given the observation is B. In the BL process by repeatedly applying equation 2.13, we refine the estimation of the stochastic process. This rule is at the core the BL methodology and we use for learning the opponent's strategy. Having highlighted the main aspect of the Bayesian learning technique we proceed to describe models in the literature that have used both these methods for agent negotiation.

2.4 Adaptive Bilateral Negotiation Models

Among the first models to specifically consider learning in the negotiation context were [Kalai and Lehrer, 1993; Kraus and Subrahmanian, 1995]. However, while their work discusses the concept of reasoning based on experience among negotiating agents, they do not explicitly develop a learning model. In [Zeng and Sycara, 1998] this notion is formally modelled, as a sequential learning mechanism based on a Bayesian belief update process. Here, a very general framework is adopted for the negotiation in which multiple agents bargain for multiple items. Moreover, the *a priori* model that the agents have of their opponents is constantly updated using current information which is received as a signal from the environment. In particular, given the prior domain knowledge of an agent and the newly incoming information, the posterior distribution of the domain knowledge of the agents is computed using Bayes rule. Now this is similar to what we wish to do, because we wish to build a model of the opponent's behaviour by observing its offers. However, Zeng and Sycara's model is developed for negotiations in stationary environments. By this, we mean that, although the agents are assumed not to know the distribution of players' strategies and their negotiation parameters, it is assumed that this distribution does not change over time. Unfortunately, we believe that this is a serious shortcoming for describing negotiation in the types of environments in which multi-agent systems are often deployed. To address this issue, in this work we have considered negotiation in non-stationary environments. We have used both Reinforcement Learning (RL) and Bayesian Learning (BL) to develop adaptive negotiation algorithms for non-stationary algorithms. Henceforth in this work, by dynamic or non-stationary environments we mean that both the agent's own parameters (time deadlines and budget constraints) and its opponent's behaviour can change with time during the course of the negotiation process.

In more detail, their model considers the case where two agents negotiate on behalf of their users in a supply chain management scenario. A typical supply chain extends from the procurement of raw material to the delivery of finished products to the customers. Several decisions regarding the purchase of raw materials, quantity of goods to be produced, transportation of goods, and the price of the finished product, need to be made within the supply chain. In this scenario an agreement is expected to be reached on the delivery date, product mix and price of the desired product. Again the dynamics of this environment are subject to constant change, forcing the agents negotiating on behalf of their users to adopt some form of learning in order to bargain effectively. In this model, an agent, when it receives an offer from its opponent, first updates its knowledge about its opponent and then evaluates this offer in the light of this new knowledge. We list below the main components of this model:

- Each element of the history set H is a sequence of actions $(a^k_{k=1,\dots,K})$ performed by the agents during the negotiation process.
- A function Q maps each member of H to a numerical value. This function defines an ordering of the agent's responses. Q is termed the player function.
- The set Ω contains information about the parameters of the environment which are subject to change. These include beliefs about the opponents' decision making process and reservation prices.
- For every $h \in H$, and each player i, a probability distribution, $P_{h,i}$, is defined which represents all the information that the player has about the negotiation process.
- For each player i, history h and action a an implementation cost $C_{i,a,h}$ is defined. This cost can be interpreted as the cost of communication between agents or the cost incurred due to delaying the agreement.

Given these components, we can now list some of the main assumptions of this learning model:

- When player i makes an offer at the $k + 1^{th}$ step in the negotiation process it is assumed that the player has information about all the actions taken by all the players up to that point in the negotiation.
- The set of probability distributions over Ω, $P_{H_{i,k-1},i}$ is known to i.

Under these assumptions, the main steps of the solution procedure are:

- The main concept used in this method is to update the information of the agent, i, using the prior distribution of histories, $P_{H_{i,k-1},i}$, and the incoming signal from the environment, $H_{i,k}$, to determine the posterior distribution $P_{H_{i,k},i}$, using Bayes rule of conditional probabilities:

$$P(H|e) = \frac{P(H_i)P(e|H_i)}{\sum_{k=1}^{n} P(e|H_k)P(H_k)} \tag{2.14}$$

- Select the best action based on recursive evaluation criteria:

$$V_{i,k,h} = max_{a_i \in A_i} - C_{i,a_i,h} + \int_X [V_{i,k+1,(h,a_i) \times P_{h,i}(X)}]dX \tag{2.15}$$

Here $C_{i,a,h}$ represents the implementation cost for each agent, i, history, h, and action, a, which has been deducted from the expected payoff.

In this learning model an agent's knowledge about its opponent is updated based on signals that it receives from the environment. But it is assumed that the pattern of the opponent's behaviour does not change. Therefore this model also develops only stationary strategies. Nevertheless this model motivates our work by showing how BL techniques can be used to model opponent behaviour and how the prior distribution can be updated by the incoming information. As can be seen, BL techniques form an important part of the general theory of machine learning and have been used in the literature for modelling agent behaviour in a negotiation context. Therefore, to satisfy our requirement 5 of reading patterns of change in the opponent's behaviour, we will explore the possibility of using these techniques, along with those from RL, to adapt to both a changing environment and adaptive opponents.

For the sake of completeness, we will briefly examine how adaptive techniques have also been used offline to develop negotiation strategies in multi-agent environments. Here, specifically, genetic algorithms (GAs) are used for developing negotiation strategies [Matos et al., 1998]. Specifically, GAs are search algorithms using which it is possible to generate sequences of 'ever improving populations' [Holland, 1992]. In this vein, Matos et al. [1998] encode the negotiation strategies from Section 2.2 (time-dependent, resource dependent, imitative) as *genes*. The strategies are then assigned fitness scores. Such a score for the agents determines how well a given agent performs amongst others in the same population. This score also determines the agent's probability of surviving to the next generation. In particular this work uses *Tournament Selection* [Blickle and Thiele, 1995] to evolve populations. The overall aim of this technique is to generate at each iteration a population that is more fit than its predecessor. The search algorithm terminates when the population is stable and when most of the individuals in the population have the same fitness scores.

The main challenge, therefore, is to assign fitness scores to strategies in such a manner that those strategies that are termed fit, also maximize the agents' utility functions. The agents using the GA approach develop negotiation strategies offline, but rather than exploring the space of all possible negotiation strategies, they search for the best strategy from among a combination of finite predefined alternatives. Thus this technique does not guarantee that the best negotiation strategy will actually be employed consistently through the negotiation process. Also the scoring process does not take into account possible changes in the environment that could render a strategy undesirable, for the current state of the system, that was awarded a high fitness score earlier in the negotiation process.

Using this technique, the agents can develop strategies that adapt to changes in the environment, but it is assumed that the agents know their own utility functions and also that this function does not evolve with time. Also this approach relies heavily on developing fitness scores. Again through multiple iterations the same fitness scores are used to determine the population of fit strategies. However, this is clearly not what we want to model because, according to our requirement of non-stationarity, we want a technique by which the fitness scores also evolve over time. Therefore while this model satisfies requirements 1 and 5, it fails to satisfy the requirements 2 and 3.

2.5 Summary

In this chapter we have presented a review of the existing work in adaptive models for negotiation in multi-agent systems. Our twin objectives are to develop a model that adapts to both environmental changes and to changes in the opponent's behaviour (see Chapter 1). To this end, we have examined RL and BL techniques in detail and have described the main models that used these techniques for adaptive negotiation. In general, although many of these models have discussed the need for adaptation in multi-agent systems they have developed only static algorithms. We have also seen the complementary nature of these two approaches and have decided that they both have features that are desirable in our solution. Specifically, we need to use RL methods because *a priori* model of the environment is not required for learning. Thus this method enables agents to learn in new and unfamiliar circumstances. We use BL methods because the agent, by repeatedly interacting with its opponent, can use its knowledge of the history of the process to build a model of its opponent's behaviour.

In more detail, this chapter has described the core of the body of work on negotiation models and we have seen how these models fail to fully meet our requirements. Nevertheless, we have reviewed the techniques that we believe are most suitable for coping with our specific requirements and presented a justification for selecting them. Specifically, we will use the NDF approach to generate offers and in order to easily change the behaviour of the agents (to satisfy requirement 4). In order for the agent to develop strategies in the absence of information in an unsupervised manner, we will adopt a RL approach to learning the optimal state-action mapping (thus satisfying requirements 1 and 2). Using this approach, agents will also be able to respond to changes in the environment and negotiation parameters (satisfying requirement 4). We will then use Bayesian learning techniques to model the behaviour of the agent so that it can adapt to changes in the opponent's behaviour (satisfying requirement 5).

In short, we will use both these learning techniques to negotiate effectively when the transition function and the reward signal change with time, thus satisfying our requirement of non-stationarity. Given this background, in Chapter 3 we will describe in detail how we use RL techniques to develop an adaptive negotiation model ands in Chapter 4 we will use BL techniques to adapt to changes in the opponent's strategies.

Chapter 3

Adapting to Environmental Dynamics using Reinforcement Learning

Given the requirements of our model detailed in Chapter 1, it is clear that the negotiating agents need to adopt some form of unsupervised machine learning in order to be able to bargain effectively. Here we, for the first time, cast the dynamic negotiation model as a Reinforcement Learning (RL) problem (as motivated in Chapter 2) and then present the adaptive algorithm that we have developed for addressing some of the requirements of our model. In doing so, we show that our algorithm adapts to changes in negotiation parameters and consistently outperforms other standard negotiation techniques.

In more detail in the first section we cast our negotiation problem as a RL problem. Specifically, we define the concept of non-stationary Markov Decision Processes, present the main ideas used in the development of the adaptive algorithm, explain the structure of our model and present the new adaptive negotiation algorithm that we have developed for dynamic negotiation. In the second section, we describe how the algorithm can be used in the mobile communications domain, since the dynamic conditions that exist here and in other multi-agent systems have specified the requirements of our model. After this, we present our empirical evaluation of the performance of our algorithm and finally, we present a discussion of the results and state our conclusions.

3.1 Dynamic Negotiation as a Reinforcement Learning Problem

Following our analysis of the literature, we are now ready to cast our negotiation problem as a RL problem. Keeping in mind that we essentially wish to model the negotiation process in various trading scenarios in distributed multi-agent systems, we make the following assumptions:

1. Two agents, designated as a buyer and seller, negotiate over the price of a service.
2. The two agents do not have complete information about their opponents' negotiation parameters and have to *learn* them from signals received from the environment (as per requirement 2).
3. The dynamics of the system are only probabilistically known (i.e., $T^a_{ss'}$, the transition probability function, is not exactly known, but it is specified by another probability function). This addresses our requirement that the dynamics of the system are unknown.
4. The negotiations are conducted over a network and depending on the quality of the network, defined by the bandwidth available for communication and the number of users, the negotiation preferences of the agents can change [Paurobally et al., 2003]. Thus a strategy that is optimal at a time instant in the negotiation process may not be optimal at a different time instant, even if all other states are the same. Thus, as specified in requirements 3 and 4, the negotiation strategies need to adapt to these changes in the environment (see Section 1.3 for more details).

5. The two agents have deadlines and they must complete the negotiation process before their time is up (as per requirement 6).

6. The two agents have an initial price and a reservation price.

7. The agents are rational and, as such, attempt to maximize their utility functions.

In our setting of incomplete information and variable parameters, the agents have to devise strategies for reaching an agreement. As argued previously, the agent has to decide on the best course of action given the current state of the system and we believe the RL framework is well suited for this type of decision making. Therefore we cast the negotiation as a MDP and, in this framework, we develop a value iteration algorithm for determining a strategy for dynamic negotiations. Hence, the specific problem that we wish to consider is set in a non-stationary environment where the dynamics of the system vary with time (as argued in Chapter 1). Thus the associated decision process is also non-stationary and we are in the realm of non-stationary MDPs. Given this, we first present in this section the corresponding notion of non-stationary MDPs and their associated state and action value functions.

3.1.1 Non-Stationary MDPs

In Section 2.3.1 we have defined and described the concept of MDPs and the process of obtaining an optimal policy by defining state and action value functions. In this section, therefore, we explicitly define these concepts for the non-stationary case. A non-stationary MDP for each time-step, n, is defined as [Garcia and Ndiaye, 1998]:

- a discrete state space S_n
- a set of discrete actions A_n
- a reward function $R_n : S_n \times A_n \rightarrow \Re$

- a probabilistic state transition function, $T_n : S_n \times A_n \rightarrow [0,1]$, $T_n(s_n, a, s_{n+1})$ is defined as the probability of making a transition from state s_n to state s_{n+1} using action a_n.

In a standard MDP, an agent tries to find a policy $\pi : S \rightarrow A$ that maps an action, a, to a state, s, and maximizes its expected sum of discounted rewards over an infinite period of time (as discussed in Chapter 2). However, in our negotiation context, the agents have finite deadlines and, therefore, we define the corresponding notion of maximizing expected rewards for a finite time horizon. In this case, the policy π can be decomposed into a set $\pi_1, \pi_2, ..., \pi_N$ where $\pi_n : S_n \rightarrow A_n$. As the first step towards determining the optimal policy, we introduce the notion of the value of a state. Formally, a *value* of a state $s \in S_n$, under a policy π_n, during the n^{th} time-step, is defined as:

$$V_n^{\pi}(s) = \sum_{t=n}^{N} E(R_t(s_t, \pi_t(s_t))|s_n = s) \tag{3.1}$$

where s_n is the state of the system at time-step n, R_n is the reward obtained at time step n, and $E(R_t(s_t, \pi_t(s_t))|s_n = s)$ is the expected value of the reward under policy π_n and state $s_n = s$. The optimal policy is denoted by π^* and the associated value function is given by:

$$V_n^*(s) = max_a[R_n(s,a) + \sum_{s' \in S_{n+1}} T_n(s', a, s) \times V_{n+1}^*(s')] \tag{3.2}$$

for all $s \in S_n$, $n \in 1, ..., N$ and $V_{N+1}^* = 0$. The optimal policy is specified by $\pi_n^*(s) = a$ where a is the action at which a maximum is attained in equation 3.2. Now, when the dynamics of the system are known, the optimal value function can be solved by standard dynamic programming techniques [Sutton and Barto, 1998]. However, in our negotiation problem, the probability of state transitions (changes in the dynamics of the system) are not known exactly. Rather they are themselves specified by another non-stationary probability function, P_n, called the estimate function. We define this function as:

$$P_n(s,\ a,\ s') : T_n(s,\ a,\ s') \rightarrow [0, 1] \tag{3.3}$$

Since the transition probabilities are not exactly known, it is not possible to compute the value functions based on these values. Instead, we use the expected values of the transition probabilities to compute the value functions and have developed a value iteration algorithm based on the average or expected $T_n(s,\ a,\ s')$ values given by:

$$E_n(T_n(s,\ a,\ s')) = \sum_{s'}(P_n(s,\ a,\ s')) \times (T_n(s,\ a,\ s')) \tag{3.4}$$

Having defined the concept of a non-stationary MDP and having introduced the notion of an estimate function, we are now ready to describe the details of our iteration method for determining an optimal policy.

3.1.2 Average Value Iteration

Here we describe the key notions used in developing an adaptive negotiation RL algorithm. Our algorithm is based on the value iteration method as this is one of the most effective ways of solving RL problems for situations in which we have incomplete knowledge of the system [Sutton and Barto, 1998]. Specifically, this method is used in several key techniques for solving standard reinforcement learning problems including dynamic programming and Monte Carlo methods. In order to use the value iteration method in our problem, we use estimated values of the probability transition function $T_n(s,\ a,\ s')$ because in our environment it is not reasonable to assume that the dynamics of the system that are specified by the transition probabilities are exactly known. As in other iteration algorithms (detailed in Chapter 2), here also the agents use state and action value functions for determining the optimal policy. The value function is iteratively estimated using the update rule given in equation 3.5 and an arbitrary policy initially. Using these values, the policy is improved upon by choosing that action from each state which maximises the action value function. Using this new policy, the value function is again estimated and this process continues until both the value functions and the policy converge (as explained in Chapter 2). We now formally describe our average value iteration method for finding an optimal policy.

We first redefine the state and action value functions in terms of the average values:

$$V_n^*(s) = max_a[R_n(s,a) + \sum_{s' \in S_{n+1}} E_n(T_n(s',a,s)) \times V_{n+1}^*(s')] \qquad (3.5)$$

$$Q_n^\pi(s,a) = \{ R_n(s,a) + \sum_{s' \in S_{n+1}} E_n(T_n(s,a,s')) \times V_{n+1}^\pi(s')\} \qquad (3.6)$$

The corresponding optimal Q-function is given by:

$$Q_n^*(s,a) = \{R_n(s,a) + \sum_{s' \in S_{n+1}} E_n(T_n(s,a,s')) \times V_{n+1}^*(s')\} \qquad (3.7)$$

for all $s \in S_n$ and $a \in A_n$.

Now from equations (3.5) and (3.7) we have:

$$V_n^*(s) = max_a[Q^*(s,a)] \tag{3.8}$$

Then once the Q-value for each state, s, is determined using the average value iteration algorithm, the agent will deterministically choose the action, a, that maximizes the Q-value. That is, it will assign $\pi^*(s,a) = 1$.

3.1.3 The Structure of the Dynamic Negotiation Model

For the reasons outlined in Section 2.3, we base our negotiation model on MDPs. Here, we formally define the Markov negotiation set as composed of two Markov decision processes: $(S_n^1, A_n^1, P_n^1, R_n^1)$ and $(S_n^2, A_n^2, P_n^2, R_n^2)$ where S_n^a is the non-stationary discrete finite state space for agent a, A_n^a is the discrete finite action space for agent a, P_n^a is the estimate function for agent a, and R_n^a is the reward function for agent a, at time instant n. Having defined the concept of Markov negotiation, we now elaborate on the process of generating offers in our model.

In more detail, the agents use negotiation decision functions (NDFs) (see Section 2.2) to generate offers (since these have been developed specifically for negotiations in incomplete and time constrained environments). Formally, these are mathematical functions that generate offers between the initial offer and the RP of the agent. Since in our negotiation process we wish the agent to generate offers between these two values (initial and RP) in a systematic fashion which we can control, we have used these functions to generate offers. Again using these functions we can control the rate at which the agent's offers approach its RP depending on the currently available resources, the current reservation price and the other identified factors, by choosing the parameter ψ (see section 2.2 for more details on generating offers). The behaviour of an agent in a negotiation process is characterized by the speed with which it approaches its RP. By controlling the value ψ, we can make the agent exhibit a range of behaviour patterns that can broadly be classified as stubborn (approaches the RP slowly), linear (approaches the RP linearly) and conceding (approaches the RP relatively quickly). We use these three types of behaviour because they are sufficient to describe all possible ways in which the RP can be approached (as discussed in section 2.2). In this context, the key strategic decision then becomes which of them has to be adopted and at what time.

Using our algorithm (detailed in section 3.1.4), the agent will endeavour to appropriately map its negotiation actions to situations so that an agreement is reached before the deadline and before the resources available for negotiation are exhausted. Intuitively, this is achieved by rewarding the agent when it adopts a stubborn approach when there are adequate resources for the negotiation process and, correspondingly, rewarding the agent for adopting a conceding approach when the available resources are low. The selection of the appropriate action is based not simply on the immediate reward that the agent will obtain, but takes into consideration all possible future rewards during the entire course of the negotiation. Thus using our algorithm, the agent at each stage chooses an action that would earn it an optimal reward rather than having a set of strategies specified at the start of the negotiation encounter. We now move on to describing the adaptive negotiation algorithm itself.

3.1.4 The Adaptive Negotiation Algorithm

In this section we outline the steps of the average value based iteration algorithm based on observations of the state space, actions of the agents and the reward signals (see Algorithm 1). In more detail, the agent must determine the best policy to adopt for a given state of the system. In order to learn this mapping using our algorithm, the agent first initializes the value of the states $V^{\pi}(s) = 0$ for all states $s \in S$. It then uses, initially, an arbitrary policy π (random policy or equiprobable policy where all actions are equally likely for instance or a deterministic policy) to generate the next set of $V^{\pi}(s)$. Then, using this value function, it improves the current policy π, by choosing the action a that maximizes equation 3.8. This process continues until both the value function and the policy converge. The agent learns this policy π offline and then uses this learned policy in its negotiations with its opponent. The policy specifies the value of the parameter ψ. The agent uses this value of ψ in equation 2.1 and generates offers. Therefore the algorithm consists of two parts. First, the determination of the optimal policy, π^*, using the average value iteration method (function AVERAGE VALUE ITERATION) and then using this policy to determine an offer (function OFFER DETERMINATION). The process terminates when an agreement is reached or when either deadline is reached.

3.2 Application to Scenario 1

The negotiation scenarios described in Chapter 1 have been motivated by the likely conditions that will become prevalent in the mobile communication domains. To ground our model further into this context, this section will demonstrate exactly how our algorithm addresses the negotiation issues raised by scenario 1.

In Scenario 1, the agent representing the salesman needs to negotiate with one service provider for either a terrestrial or a satellite broadcast of his favourite programs. As explained in Chapter 1, the agent has to adapt to resource availability changes and also to changes in negotiation parameters like deadlines and reservation prices. Here we formally cast this scenario as RL model and examine how the agents can use our algorithm to develop effective negotiation strategies. Towards this end we define:

- State Space:$S = \{$Resource Availability (RA), Time Availability (TA), Reservation Price (RP) $\}$. To represent the fluctuations in resource availability we assume that it is described by two states: High and Low. Similarly, to represent the changes in the time available for negotiations we assume that Time Availability can be in two states: High and Low. Again in order to represent the fact that the reservation price can also change, we assume that it can have two values during

Algorithm 1 Adaptive Negotiation Algorithm

```
function AVERAGE VALUE ITERATION ($p^t_{\mathring{a}\to a}$, $s_n$, $P_n(T_n(s', a, s))$, $R_n(s,a)$) returns $\pi^*$

inputs:
    1. $p^n_{\mathring{a}\to a}$ (the offer of the opponent at time instant $n$).
    2. $s_n$ (the state of the system at time instant $n$).
    3. $P_n(T_n(s', a, s))$ (the current estimation function based on the partial knowledge
       of the system at time instant $n$).
    4. $R_n(s,a)$ for each $(s,a)$, $s \in S_n$ and $a \in A_n$ (the reward function at time
       instant $n$).
initialize $V^\pi_n(s) = 0 \; \forall \; s \in S$ for an arbitrary policy $\pi$.

for each $s \in S$ do
    $V^\pi_n(s) = max_a[R_n(s,a) + \sum_{s' \in S_{n+1}} E_n(T_n(s', a, s)) \times V^\pi_{n+1}(s')]$
    Set $\pi(s) = a$
return Policy $\pi^*$

function OFFER DETERMINATION ($a$) returns value of $\psi$.
inputs: action $a$ determined by $\pi^*$.
if $a = Boulware$ then
  Set parameter $0 < \psi < 1$
else $\{a = Conceder\}$
  Set parameter $1 < \psi < \infty$
else $\{a = Linear\}$
  Set parameter $\psi = 1$
end if
return $\psi$
Use parameter $\psi$ in the NDF $f_s(n)$ and use this $f_s(n)$ to generate offer $p(n)$.
```

the negotiation RP^1 and RP^2. Thus we have a total of $2 \times 2 \times 2 = 8$ states and $8 \times 8 = 64$ state transitions. In this fashion we capture the changes in the environment in the definition of the state space.

- Action Space: A=\{*switch between terrestrial broadcaster (TB) and satellite broadcaster (SB) (a_1), switch between SB and TB (a_2), stubborn with TB (a_3), stubborn with SB (a_4), conceding with TB (a_5), conceding with SB (a_6)*\}. Thus the agent has a total of six actions to choose from. Now the RL problem is to find a mapping between the states and the actions that maximise the reward of the agent.

- Rewards: The reward signal guides the agent in selecting an appropriate action from a state. We define the rewards as $R_n = +1$ when the agent chooses a_5 when $TA = Less$, $R_n = -1$ when the agent chooses a_3 when $T = More$, $R_n = +1$ when the agent chooses a_5 when $RA = Less$, $R_n = -1$ when the agent chooses a_3 when $RA = More$, $R_n = +1$ when the agent chooses a_1 when $RA = Less$ and $T = Less$, $R_n = -2$ when the agent chooses any other action from this state, and $R_n = 0$ for all other state-action pairs. This reward scheme has been chosen with the general idea of rewarding the agent for conceding when resources are low so that the negotiation can be completed before the deadlines are up (as per our requirement 6), to be stubborn when the resources are high so that the agent can strike a good bargain and to switch between the expensive terrestrial broadcaster and the cheaper satellite broadcaster when the funds for securing the service have been reduced.

- Estimation function: P_n is generated arbitrarily for this problem. We do this because this is the most general form of the estimation function and therefore can be used to represent the dynamics of a wide range of multi-agent systems. Clearly, if more specific information was used, the performance of the algorithm would improve.

Given this framework, we are ready to present our solution procedure for this scenario.

We first train the agent offline to determine the policy that maximises its expected reward using algorithm 1. The main steps involved in this are:

- The agent evaluates equation 3.8 for every $s \in S$ by using the update rule specified in algorithm 1. Here R_n is as specified earlier, P_n is generated arbitrarily and follows an equiprobable policy π initially (which specifies that all actions are equally likely from every state).

- Having obtained $V(s)$ for every $s \in S$, we now improve the policy π. This is done by deterministically choosing action a_s from each state s that maximises 3.8 and setting $\pi'(s) = a_s$ for every $s \in S$.
- Using this new policy $\pi'(s)$ the agent then again evaluates equation 3.8.
- This process continues until both $V(s)$ and the policy π converge to their optimal values.

Once the agent has learned the optimal policy mapping by maximising the expected reward in the RL problem, it is ready to apply this policy. Thus, using our algorithm the agent learns to choose actions that are best suited to the current situation. For instance, when the agent is in state $s = \{R = less, T = high, RP^1\}$ it will evaluate equation 3.8 and because of our reward scheme which gives a positive value for choosing a conceding strategy, the agent is encouraged to select this action from this state. Using RL techniques, each time the agent encounters this state it would choose the action that yields the maximum reward with higher probability, until it chooses this action every time it is in this state, deterministically. Thus our learning algorithm builds an intuition within the agent that guides its decision making process. Now, agents that are non-adaptive would simply determine one strategy to use depending on the initial negotiation parameters and the initial strategy profile of the opponents. This would be a naïve approach in a real-world scenario where parameters and the strategy profiles are subject to change. Our adaptive algorithm therefore gives the agents the intelligence to develop multiple strategies depending on the circumstances and thus makes them better negotiators in realistic conditions.

In Chapter 4 we show how BL techniques can be used in order to model and analyse the opponent's behaviour. Again, in Chapter 5 we examine the comparative benefits and drawbacks of using these two learning techniques to adapt to both environmental changes and to changes in the opponent's strategy. However for now our adaptive agent is only able to respond to changes in the environment. In the next section we present our empirical results.

3.3 Empirical Evaluation

In this section we describe the implementation details of our algorithm and present a detailed analysis of our empirical results. We begin by describing how the RL problem is solved for a specific negotiation scenario and then present our results using the adaptive negotiation algorithm in a wide range of negotiation encounters.

3.3.1 Solving the Negotiation

In this section we will illustrate this algorithm using an example negotiation encounter (based on that detailed in section 3.2). In this specific negotiation we simulate the following changes in the environment:

1. Deadline changes
2. Reservation price changes
3. Resource availability changes

We have chosen these factors because these play a critical role in the domains and scenarios outlined in Chapter 1. Although this is a simplified version of the scenarios presented in Chapter 1, it captures the important feature of variable environmental conditions which is the key characteristic of these environments and to which we wish our negotiation model to adapt. In order to explain how the environmental changes are represented in the state space definition and to list the action choices that are available to the agent from these states, we first need to describe the design of the state and action spaces in detail.

3.3.2 State and Action Spaces

The key endeavour in designing state spaces in an MDP is to represent most of the features in the environment in as few states as possible (since the computational expense of running the algorithm grows exponentially with the number of states [Sutton and Barto, 1998]). Hence we have to effect a trade-off between detailing the environment and restricting the size of the state space. Hence in the model described here we have assumed that the resource availability, deadlines and reservations prices can have only two values. We have then studied how the agents adapt to these changes. Thus our representation captures the changes in the environment, while ensuring that the state space is compact. Formally, the finite discrete state space of each agent in the negotiation process is defined by:

1. Resource Availability: This denotes the computational resources that are available for the negotiation. Here we assume this can take two values (high and low). These two states broadly capture the pattern of resource availability changes.

2. Deadlines: The agent has a finite discrete set of deadlines. This represents the fact that its deadline can change. For our example negotiation problem, we assume that state space consists of two deadlines: T^1 and T^2 time units.

3. Reservation Price: The agent has a discrete, finite set of reservation prices. This again means that the reservation prices of an agent can vary. In the example problem we will assume that the agent has two reservation prices: R^1 and R^2.

Now, depending on the state of the system and the input (offer) that the agent receives from its opponent, the agent chooses between a stubborn, a conceding strategy and a linear strategy. It also has the option of doing nothing (i.e., making no response) since the negotiation is a process of alternating offers, at alternate time-steps when it is the opponent's turn to make an offer the agent does nothing. Having detailed the state and action spaces, we now move on to describe the functions that govern the decision making process (namely the estimation and reward functions).

3.3.3 Estimation and Reward Functions

Intuitively, as argued for in section 3.3.1, the agent should be rewarded for choosing a stubborn strategy when the resources are high and when the agent has sufficient time to continue with the negotiation process. Similarly, it should be rewarded when it adopts a conceding approach when the resources for negotiation are low. The agent should also sense when the RP changes and accordingly choose a strategy. When the RP increases it means that the agent has more money at its disposal than at the beginning of the encounter and therefore the agent could adopt a conceding approach. While conversely, when the RP decreases the agent has less money than it had initially and therefore it needs to adopt a stubborn approach. Thus these functions should enable the agent to exploit the current conditions in order to bargain effectively. In our model specifically, the agent adapts to changes in resource availability, time constraints and budget considerations.

Given this set up, there are a total of 8 (2 Deadlines $\times$ 2 Reservation Prices $\times$ 2 states of Resource Availability) states and 64 possible state transitions. There is an estimation function $P(s', a, s,)$ (see equation 3.3) and a reward function $R^a_{ss'}$ (see equation 3.1) associated with every state transition and action. We have shown the correspondence between the state-action pairs and their estimation and reward functions in table 3.1 (these are defined for other state-action pairs in a similar manner). In our problem we have used an arbitrary probability distribution to represent the estimation function and the reward function. These distributions also change in our problem and this reflects the non-stationarity in the domain.

s	$s\prime$	a	$P(s', a, s,)$	$R^a_{ss'}$
(low,T^1,R^1)	(low,T^1,R^1)	C	$P^C_{ss'}$	R^C
(low,T^1,R^1)	(high,T^1,R^1)	C	$P^C_{ss'}$	R^C
(low,T^2,R^1)	(high,T^2,R^1)	B	$P^B_{ss'}$	R^B
(low,T^2,R^1)	(high,T^2,R^2)	B	$P^B_{ss'}$	R^B
(low,T^2,R^1)	(high,T^2,R^2)	C	$P^C_{ss'}$	R^C
(high,T^2,R^1)	(high,T^2,R^2)	B	$P^B_{ss'}$	R^B
(low,T^2,R^1)	(low,T^2,R^2)	B	$P^B_{ss'}$	R^B
(high,T^2,R^1)	(high,T^1,R^2)	C	$P^C_{ss'}$	R^C
(high,T^2,R^1)	(high,T^2,R^2)	B	$P^B_{ss'}$	R^B
(high,T^2,R^2)	(low,T^1,R^2)	C	$P^C_{ss'}$	R^C
(high,T^2,R^2)	(low,T^2,R^2)	B	$P^B_{ss'}$	R^B
(high,T^2,R^2)	(high,T^1,R^2)	B	$P^B_{ss'}$	R^B
(high,T^2,R^2)	(high,T^1,R^2)	B	$P^B_{ss'}$	R^B
(high,T^2,R^1)	(high,T^1,R^2)	B	$P^B_{ss'}$	R^B
(high,T^2,R^1)	(high,T^1,R^2)	C	$P^C_{ss'}$	R^C
(low,T^2,R^1)	(low,T^2,R^2)	C	$P^C_{ss'}$	R^C
(low,T^2,R^1)	(high,T^2,R^2)	C	$P^C_{ss'}$	R^C
(low,T^2,R^1)	(high,T^1,R^2)	C	$P^C_{ss'}$	R^C
(low,T^1,R^1)	(high,T^2,R^2)	C	$P^C_{ss'}$	R^C
(low,T^1,R^1)	(high,T^2,R^2)	B	$P^B_{ss'}$	R^B
(low,T^1,R^1)	(high,T^2,R^2)	B	$P^B_{ss'}$	R^B
(low,T^1,R^1)	(high,T^2,R^2)	B	$P^B_{ss'}$	R^B
(low,T^2,R^2)	(high,T^2,R^2)	B	$P^B_{ss'}$	R^B
(low,T^2,R^2)	(high,T^2,R^2)	C	$P^C_{ss'}$	R^C
(low,T^2,R^2)	(high,T^2,R^2)	C	$P^C_{ss'}$	R^C
(low,T^2,R^2)	(high,T^2,R^2)	C	$P^C_{ss'}$	R^C

TABLE 3.1: Reward Scheme.

With the estimation function defined and the reward scheme in place, we now move on to the details of the evaluation of our algorithm.

To do this, for a set of negotiation parameters (T^a, RP^a, IP^a) we allowed a buyer and a seller to negotiate and measured the value at which an agreement was reached and the time taken to get there. We repeated the experiment for a number of different negotiation parameter sets. Using the adaptive algorithm detailed in section 3.1.4, the agent learned an optimal mapping between state and action pairs offline and then used this policy online against its opponent during the actual negotiations. During the actual run, we used parameter sets different from those used in the training of the agent. Specifically, we found that in the actual negotiation encounters the agent learned to choose an appropriate value of ψ and hence an appropriate strategy given the current state of the system.

To provide a benchmark for our algorithm, we compared it against an agent that uses NDFs to determine the negotiation strategy, but that does not adapt its strategy in response to changing resource availability, deadlines or reservation prices during the course of the encounter and also compared it with another agent that uses an evolutionary algorithm to read changes in the environment. In the NDF approach, the agent adopts either a Boulware or a Conceder strategy throughout the negotiation process, irrespective of changes in the simulated environment. In the heuristic method, the agent uses the tournament selection technique to pick a strategy in response to changes in the environment. Here we picked an arbitrary number of strategies from the total population. Now to each strategy we assigned a probability p as being the best, $p * (p - 1)$ as being second best and so on. Now we allowed the strategies to play a 'tournament' where their probailities are exchanged or get 'mixed'. We then applied the strategy which had the highest probability of being the best.

We defined the performance metrics as the number of times an agreement was reached within the deadline T under adaptive conditions, compared to a non-adaptive algorithm (based on [Fatima et al., 2004]) and a heuristic algorithm (based on [Matos et al., 1998]). Although this was chosen as the metric, the reward structure used in the RL process also reflected the profit that the agent could earn from the negotiation. Therefore the strategy was chosen as a trade-off between profit share and number of agreements reached. In the first instance, we trained the agents using a specific reward scheme. During the training the agents learn an optimal policy that maximises the total reward. Multiple negotiation runs were then used until the agent learnt the optimal policy to adopt from each state of the system. Now during the actual negotiations the agent applies this policy to adapt to the negotiations. Here it is important to note that it is not necessary to use the same negotiation parameters as in the training, we only need to use a reward scheme that rewards and penalises the agent in accordance with the training reward scheme. During the training we ran extensive experiments with a number of different combinations of T^a, RP^a, RA and reward schemes. These parameters are generated in the program by random functions. The agent checks the value of each of the three randomly generated parameters before making an offer and sets its own value of each of them to either high or low. Thus, based on these values, it determines the state of the system. Then using the reward scheme it determines an optimal policy according to algorithm 1. We computed the actual optimal policy by solving the standard Markov Decision Processes problem in which the agents have complete information about the underlying environment using MATLAB 7.1.

In more detail, Figure 3.1 shows the number of iterations it took for the trained agent to converge in each negotiation encounter 200 negotiations. For each of these negotiation encounters we used a different reward scheme. Table 3.2 gives the mean (μ_1) and standard deviation (σ_1) of the distribution of the number of iterations to converge to the optimal solution. As can be seen, it takes an average of 23 iterations to converge to the optimal solution. Here, it must be stressed that the agent does not learn over 23 negotiation encounters, but rather that, within a negotiation, our recursive algorithm takes 23 iterations to converge to the optimal solution. Further, 98% of the population is within 2σ distance from the mean.

Figure 3.2 shows the average time taken for each iteration over the 200 negotiations and table 3.3 shows the mean (μ_2) and standard deviation (σ_2) for this distribution. Given that each iteration takes 51 seconds to be completed, the total time to find an optimal solution is $23 \times 51 = 1173$ seconds (or 19.5 minutes). This short time to convergence is a further strength of our algorithm and makes it especially suited for multi-agent systems, like digital marketplaces, where negotiation deadlines are typically short.

Finally, figure 3.3 shows our comparison results over these 200 negotiations. We have compared the number of agreements that were reached in the different negotiations using our algorithm, the heuristic algorithm and the non-adaptive algorithm. In conditions where the negotiation parameters change, using our adaptive algorithm nearly 90% of the negotiations end in agreements. This is a direct of result of judging the conditions correctly and developing appropriate strategies. The heuristic algorithm, on the other hand, develops strategies for a fixed set of conditions and does not adapt very well to new situations. Therefore the number of agreements it reaches is much lower (56%) than the number obtained using our adaptive RL algorithm. Nevertheless it still outperforms the non-adaptive algorithm which simply applies pre-determined strategies without taking into consideration any of the changes in the negotiation process.

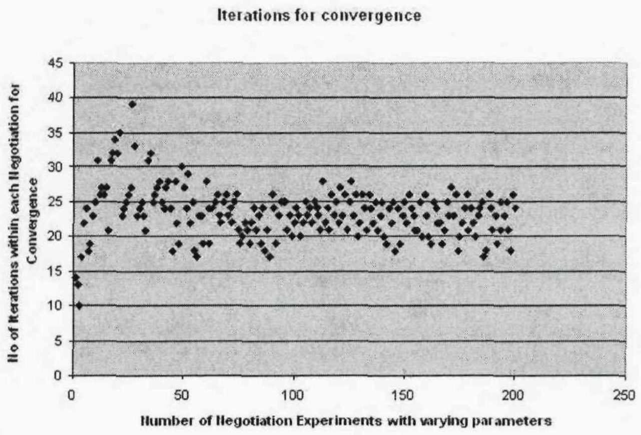

FIGURE 3.1: Iterations to convergence to optimal solution

Negotiations	Mean	Std Dev
200	23.39	3.73

TABLE 3.2: Iterations for convergence to optimal solution

Negotiations	Mean	Std Dev
200	51.125	7.55

TABLE 3.3: Time for each algorithm iteration

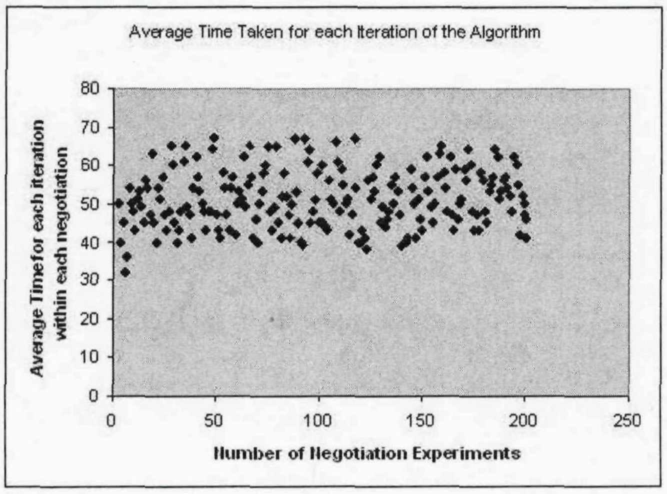

FIGURE 3.2: Average time for each algorithm iteration

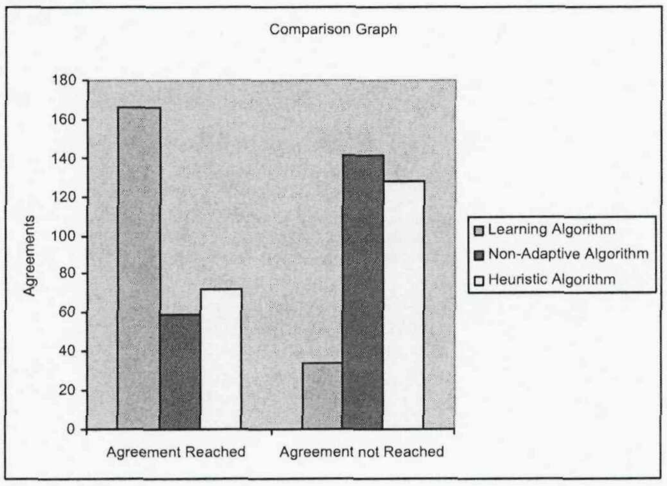

FIGURE 3.3: Comparison of the number of agreements reached using different algorithms.

3.4 Summary

In this chapter we have modelled a bilateral negotiation problem in the mobile communications environment as a RL problem and have developed a novel average value iteration algorithm for determining an optimal policy. Using our algorithm, the agent autonomously adapts to dynamic changes in the environment, specifically reservation prices, deadlines and resource availability. The strategies are not specified at the start of the negotiation process, but rather the agents using our algorithm learn to map appropriate actions to the states that they encounter as they progress with the negotiation.

This satisfies our requirements 1 and 4. Also in our model the reward scheme is given by a probability function and hence is non-deterministic. This corresponds to the negotiation preferences of the agents changing during the negotiation encounter and hence addresses our requirement 3. The algorithm also functions in the absence of complete information about the environment which addresses our requirement 2. Also the algorithm, by appropriately specifying rewards, ensures that the negotiation finishes before the deadlines and before the resources are exhausted (thus satisfying requirement 6). Requirement 5, that of adapting to changes in the opponent's behaviour, will be addressed in Chapter 4.

To empirically evaluate the performance of our algorithm we have compared the number of negotiations that end in agreements using our adaptive agent against other standard negotiation techniques. We have used this metric to compare the algorithms because the main aim of the negotiation process is to reach an agreement within the deadline. It is also important to note here that none of the other algorithms are attempting to adapt to changes in the environment. Therefore it is meaningless to compare other metrics like the time taken by the algorithms to complete the negotiation. As can be seen from figure 3.3 using the standard algorithm only 49% of the negotiations end in an agreement and only 56% of the negotiations using heuristic methods end in an agreement.

In summary, we have developed an algorithm for negotiations that satisfies 4 of our requirements. We also shown that it outperforms other algorithms in the literature. The most important contribution of this part of the work is that our algorithm adapts to non-stationary changes in the environment. Ours is the first algorithm that has been able to respond to changes simultaneously in different negotiation parameters and preferences. We have also shown how our algorithm can be applied to the scenario 1 introduced in Chapter 1. From our results we would argue that RL is a very promising methodology to deal with decision making in uncertain, multi-agent environments.

Chapter 4

Adapting to Changes in the Opponent's Strategy using Bayesian Learning

In the previous chapter we discussed how RL techniques can be used in agent negotiation to adapt to changes in the environment. However as stated in our requirements in Chapter 1, in addition to responding to changes in the environment, we also wish the agents to respond to changes in the opponent's strategies. Thus in Chapter 3 we discussed how RL techniques could be used for such adaptive agent behaviour in negotiations in dynamic environments. Now, in order to determine machine learning techniques that can be used to adapt to changes in the opponent's behaviour in the negotiation process we turn our attention to Bayesian learning techniques.

As before, we consider negotiation between a pair of agents over a single issue (price). We use the non-stationary Markov chain framework to model the negotiation process and prove, for the first time, an important estimation property for these processes (namely that the future distribution of the states can be obtained given their initial distribution and the probabilities of state change during the process). Within this framework, at each stage in the negotiation process, the agent uses Bayesian updating to learn the strategy that its opponent is most likely to use and, based on this, determines what it should adopt to maximise its payoff at that stage of the negotiation process. In so doing, we analytically prove that in repeated negotiations our algorithm converges to the actual optimal strategy at every stage of the negotiation process. We verify this experimentally and examine the number of negotiations needed for convergence and the average time for each negotiation. We also show that the share in the profits that the adaptive agent earns is 83% compared to only 54% of the share in the profits earned by its non-adaptive counterpart.

The rest of the chapter is organized as follows: Section 4.1 describes how to cast the negotiation as a bayesian learning problem, Section 4.2 gives our empirical results and finally, Section 4.3 concludes.

4.1 Dynamic Negotiation as a Bayesian Learning Problem

In this section we describe a new negotiation model that uses Bayesian learning techniques to adapt to changes in the opponent's behaviour. To do this we first describe the concept of mixed strategy profiles in the context of classical game theory which we will use as a framework to describe the strategies that our negotiating agents will use.

Now, in classical game theory, a *Strategic-Form* game has three elements (see section 2.2):

1. the set of players $i \in I$, where $I = 1, 2, ..., n$
2. the *pure-strategy space* S_i, for each player i, and
3. *payoff functions* u_i that give player $i's$ utility $u_i(s)$ for each *profile*, $s = (s_1, s_2, ..., s_n)$, of strategies

Therefore in game theory a *strategy* is perceived as an action choice of a player that has a *utility* associated with it. Often the objective in games is to determine a strategy s_i that will maximise player $i's$ payoff given the strategy set, s_{-i}, that the other players use. Also, notice that the u_i that player i receives depends on the strategies of all the players in the game and is not related to an isolated strategy that i may use. Thus in our problem also the payoff that the agent earns will depend on its response to its opponent's play. Therefore the agent learns the mixed-strategy profile of its opponent and evolve a strategy in response to that strategy that earns it maximum payoff (see section 2.2). In this sense, our negotiation problem can be considered as a two-player strategic game.

We are now ready to describe the learning component and outline the solution procedure we have developed. The main objective of our agent is to learn the mixed-strategy profile of its opponent and determine a strategy in response to this profile that maximises its payoff at each stage of the negotiation process. This learning problem is complicated by the fact that the agent has no information about its opponent and that the strategy that the opponent uses may well change during the course of the negotiation. Now, to model this process of change in the strategies of the opponent, we use a *non-stationary Markov chain*. Formally, a *non-stationary Markov chain* is a Markov process (see Section 2.3.1) whose one-step transition probability function, **P(t)** $= Pr\{X_{n+1} = x|X_n = x_n\}$, varies with time [Kulkarni, 1996]. If we define the state space, S, associated with this non-stationary Markov chain to be the space of all possible strategies that the opponent can employ, and the corresponding time dependent transition probability function, $\mathbf{P^n(t)}$, to represent the probability that the strategy of the opponent changes at each step n of the game and that this probability function itself is a function of time, then this framework gives us a powerful tool to describe and analyse the non-stationary negotiation process that we are trying to model. Therefore, we adopt this mathematical formulation in this work.

Now, if $\mathbf{P^n(t)}$ were specified as a function of time, then we could obtain the strategy profile of the opponent at each stage of the negotiation process using standard stochastic process analysis [Karlin and Taylor, 1974] and then obtain a strategy that maximises our own payoff using maximization algorithms [Filar and Vrieze, 1996]. But since this function is unknown in our problem, the agent has to learn it from interactions with the opponent and the environment. Therefore, in our learning problem we are trying to learn the form of $\mathbf{P^n(t)}$. Now, in Bayesian learning, as explained in section 2.3.2, the probability of a hypothesis being true is continuously updated by signals that are received from the environment and, as such, is well suited to modelling uncertainty in the environment. In our problem, in order to learn the function, $\mathbf{P^n(t)}$, we propose that the learning agent initially has a finite number of hypotheses [1] of the possible distributions of $\mathbf{P^n(t)}$ which it updates using Bayesian inference rules. This means that in successive negotiations, by updating the different hypotheses, the agent comes closer to estimating the true value of $\mathbf{P^n(t)}$ and, therefore, to estimating the true optimal strategy in response to its opponent's play.

Formally, we consider two agents, say buyer X and seller Y, negotiating over price. We assume that the buyer is learning to respond to the strategy of the seller. Now, we assume that there is a payoff function, $u^n_{s_x}(t)$, associated with X, which depends on the strategy s that X uses in response to Y at each step, n, of the negotiation. $X's$ objective in the negotiation is to find a strategy profile that maximises $u^n_{s_x}(t)$ which we assume is known to X. Therefore, X must learn the strategy profile of Y in order to determine an optimal response strategy. To describe how X learns this strategy within the Markov chain, we need to first define some of its properties.

In stationary Markov chains, the probability of moving from state i to state j in n time steps is represented as P^n_{ij} and is given by [Karlin and Taylor, 1974]:

$$P^n_{ij} = \sum_{k=0}^{m} P^r_{ik} \times P^s_{kj} \tag{4.1}$$

[1]This assumption reduces the search space, while still allowing us to represent the uncertainty in the problem.

where m is the total number of states, k is some intermediate state and $r + s = n$. Intuitively, this means that the probability of moving from one state to another in a certain number of steps, n, is equivalent to the probability of moving from the first state to an intermediate state, k, in r steps together with the probability of moving from k to the final state in the remaining number of steps. Now, from matrix algebra, we recognise equation 4.1 as the formula for matrix multiplication, so the n-step transition matrix, represented by P^n, is equal to $P^{(n)}$ or that the entries P_{ij}^n in P^n are equal to the entries in the matrix $P^{(n)}$, which is the n^{th} power of the one-step transition matrix P. It follows that if the probability of the process initially being in state j is p_j, (that is, $Pr\{ X_0 = j \}$), then the probability of the process being in state k, at time n is:

$$p_k^n = \sum_{j=0}^{m} p_j P_{jk}^n \tag{4.2}$$

This equation is the classic Chapman-Kolmogorov equation for Markov Chains [Papoulis, 1984]. Thus, if we know the initial distribution of the opponent's strategies, we can calculate the probability that the opponent uses a certain strategy after n time steps, because the negotiation process is modelled as a Markov chain. This is one of the chief benefits of using the Markov chain framework to model the negotiation process. However, our process is non-stationary and, therefore, we need to use an equivalent result for the non-stationary case. Now, the main difference between the two processes is that in the non-stationary case, $\mathbf{P^n(t)}$ is a function of time and therefore at each step of the process we have a different transition probability matrix. Here, we propose that to obtain p_k^n as a function of time, we need to multiply n different transition matrices. We now formally state and prove this result.

Theorem 4.1. *In non-stationary Markov chains, the probability of moving from state i to state j in n time steps, during time instant t, is represented as $P_{ij}^n(t)$ [2] and is given by:*

[2]Here n represents the number of time steps and t represents the fact that the transition probability $P_{ij}^n(t)$ is a function of time.

$$P_{ij}^n(t) = \sum_{k=0}^m P_{ik}^r(u) \times P_{kj}^s(v)$$

where m is the total number of states, k is some intermediate state, $r + s = n$ and u and v are the time instants at which the transitions occur.

Proof. We prove the result when $n = 2$. The event of moving from state i to state j can happen in the mutually exclusive ways; of going to some intermediate state $k \in \{0, 1, 2..., m\}$ in the first transition and then moving from state k to state j in the next transition. Now, because of the Markovian assumption that the transition probability is independent of the history of the process, the probability of the second transition is simply $P_{kj}(v)$ and, by definition, the probability of the first is $P_{ik}(u)$. Therefore, by the law of total probability: $P_{ij}^2(t) = \sum_{k=0}^m P_{ik}^1(u) \times P_{kj}^1(v)$. In the general case, by breaking up the first r steps and then the next s steps into a series of single step transitions and again by using the law of total probability for each transition the proof is obtained. □

Thus, in the non-stationary case also, given the probability that initially the process was in, say state j (that is, $p_j^0(0) = Pr\{X_0(0) = j\}$), then the probability that it is in state k after n time steps and at time t is represented as $p_k^n(t) = Pr\{\ X_n(t) = k\}$ and is given by:

$$p_k^n(t) = \sum_{j=0}^m p_j^0(0) \times Q_{jk}^n(t) \tag{4.3}$$

where $Q_{jk}^n(t) = P^0(0) \times P^1(1)... \times P^{n-1}(t-1)$.

Therefore, we now have a means of obtaining the probability distribution of the process and, thereby, the probability distribution of strategies at any stage in the negotiation process given an initial distribution of strategies and the transition probability matrices.

Now, we come to the main issue of learning the transition probability matrices. As already stated, we propose to do this using Bayesian inference rules (see section 2.3.2). However, to do this we must assume that the learning agent, in our case the buyer X, has some knowledge about the negotiation process. Specifically, in order to update its hypothesis about the strategy distribution of Y from the offers that it receives, it has to know the relationship between the offer generation process and the strategy selection process. To this end, let us assume that S, the set of all possible strategies that Y can use, and as such constitutes the state space of our Markov chain, is given by $S = \{s_0^o, s_1^o, ..., s_m^o\}$. Therefore, Y switches between the strategies in S according to the transition probability matrix $\mathbf{P^n(t)}$, which varies with time. We let $O_n(t)$ represent a sequence of offers made by Y and $O_n^p(t)$ represent the event that the offer at the n^{th} step of the process at time t is p. We also let $H_n(t)$ represent a sequence of finite sets of hypotheses about $\mathbf{P^n(t)}$ during the negotiation process. Therefore $H_n(t) = \{H_n^1(t), ..., H_n^k(t)\}$, where k is some finite positive integer. We assume that the hypothesis representing the true value of the transition probability function also belongs to $H_n(t)$. Then the objective of our learning algorithm is to update each of these hypotheses $\{H_n^i(t) \in H_n(t)\}$ at every step n of the negotiation. The steps of the algorithm are detailed in Algorithm 2.

In more detail, applying Bayes rule (see equation 2.13) we have for each hypothesis, at step n, of the process that:

$$Pr\{H_n^i(t)|O_n^p(t)\} = \frac{Pr\{H_n^i(t)\} \times Pr\{O_n^p(t)|H_n^i(t)\}}{\sum_k^{i=0}[Pr\{H_n^i(t)\}] \times [Pr\{O_n^p(t)|H_n^i(t)\}]} \tag{4.4}$$

We call $Pr\{H_n^i(t)|O_n^p(t)\}$ the likelihood function L. Thus each $H_n^i(t)$ is updated in the light of the incoming offer $O_n^p(t)$. Now, B uses the hypothesis $H_n^{new}(t) = \sum_k^{i=0}\{Pr\{H_n^i(t)|O_n^p(t)\} \times H_n^i(t)\}$ to find the strategy used by the opponent. Therefore the learning agent weights the different hypotheses by the probabilities of their occurring in order to form a new hypothesis about $\mathbf{P^n(t)}$. Because of this construction of the new hypothesis we can show that, as $t \longrightarrow T$ where T is sufficiently large, $H_n^{new}(t)$ approaches the true value of $\mathbf{P^n(t)}$ (see Theorem 4.2 for details). Then, according to $u_{s_x}^n(t)$, the agent determines the strategy $s_{max}^n(t)$ that maximises $u_{s_x}^n(t)$ at each step of the negotiation process. We denote this maximum value of the payoff function by $u_{s_{max}}^n(t)$. This completes the solution procedure for determining the best response strategy to the opponent's play and consequently the maximum payoff at each step of the negotiation process.

Algorithm 2 The Adaptive Negotiation Algorithm

```
1.  for $(t = 0, 1, 2, ..., T)$
2.        initialize $H_n^i(t) \in H_n(t)$ as an arbitrary distribution.
3.     for $(n = 0, 1, 2, ..., t_{terminal})$
4.        input opponent offer $p \in Domain\{O_n(t)\}$
5.        $Pr\{H_n^i(t+1)\} \leftarrow Pr\{H_n^i(t)|O_n^p(t)\}$ using equation 4.4
6.        assign $H_n^{new}(t) = \sum_{i=0}^{k}\{Pr\{H_n^i(t)|O_n^p(t)\} \times H_n^i(t)\}$
7.        assign $[P^n(t)] = H_n^{new}(t)$
8.        compute $[(s_1^o, s_2^o, ..., s_m^o)]^n(t)$ using equation 4.3
9.        compute $s_{max}^n(t) = \max [(s_1, s_2, ..., s_m)]^n(t) \times u_x^n(t) \times [(s_1^o, s_2^o, ..., s_m^o)]^n(t)]^T$ s.t $\sum_{i=0}^{m} s_i = 1$ and $s_i \geq 0\ \forall i$
10.       compute $u_{s_{max}}^n(t)$
11.    next $n$
12. next $t$
```

Having detailed the steps of the algorithm we now show how it can be applied in the context of Scenario 2. In subsection 4.1.1 we describe how this algorithm can be applied to scenario 2. Then in subsection 4.1.2 we describe the conditions under which this algorithm converges. In subsection 4.1.3 we detail how the strategy space can be represented in a concise manner. Then in subsection 4.1.4 we describe how such a strategy space can be used to achieve a better estimation of the opponent's offer generating model. Finally in subsection 4.1.5 we describe an adaptive algorithm for negotiations based on this estimation method.

4.1.1 Application to Scenario 2

Here we consider specifically, how this algorithm can be applied to the situation described in Scenario 2 (see Chapter 1). We do this in order to detail the steps involved and how they can be instantiated in a specific situation. As already discussed, this regional manager needs to negotiate with an unknown network operator for some services. Now, in order for her to get a good deal she must model her network operator's behaviour and adapt her negotiation strategies to changes in the operator's negotiation pattern. To illustrate how she could achieve this using our algorithm, we make the following assumptions:

1. The strategy space S of the operator consists of two strategies, $S = \{s_1, s_2\}$ (this can easily be expanded to many strategies).

2. At each time step n of the negotiation, the manager has a set of three hypotheses about the possible value of $\mathbf{P^n(t)}$ which here governs the manager's offer generating pattern.

We now describe the solution procedure in our problem.

- Here, as specified in Step 3 of algorithm 2, the manager initializes, $H_0^i(0) \in H_0(0)$.

$$H_0^1(0) = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}$$

$$H_0^2(0) = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix}$$

$$H_0^3(0) = \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix}$$

Now since the manager is unaware of the true value of $\mathbf{P^n(t)}$, it assigns arbitrary probabilities to each of these hypotheses about $\mathbf{P^n(t)}$. Specifically let,

- $\{Pr(H_0^1(0)) = p_{00}^1\}$,$\{Pr(H_0^2(0)) = p_{00}^2\}$, $\{Pr(H_0^3(0)) = p_{00}^3\}$ and the offer of the operator $O_0(0) = O_{00}$.

- Then according to Step 4 in algorithm 2, her agent observes the offer of the opponent and makes this assumption: $Pr\{O_0(0)|H_0^1(0)\} = q_{00}^1$, $Pr\{O_0(0)|H_0^2(0)\} = q_{00}^2$ and $Pr\{O_0(0)|H_0^3(0)\} = q_{00}^3$ (here we assume arbitrary values for $Pr\{O|H\}$, but in section 4.1.4 we formalize this relationship).

- Using Step 5, her agent updates its knowledge about the operator. In other words, using the algorithm it updates the probabilities as $Pr\{H_0^1(0)|O_0(0)\} = u_{00}^1$, $Pr\{H_0^2(0)|O_0(0)\} = u_{00}^2$ and $Pr\{H_0^3(0)|O_0(0)\} = u_{00}^3$.

- Based on these updated probabilities, the agent then determines a new hypothesis for the operator's behaviour. Formally, from Step 6, it determines $H_0^{new}(0) = (u_{00}^1) \times H_0^1(0) + (u_{00}^2) \times H_0^2(0) + (u_{00}^3) \times H_0^3(0)$.

- Now Step 7 determines the strategy profile of the operator. Here we assume that the operator's initial profile is known to the agent and is represented as $[s_1^o, s_2^o]$

and the payoff function of the agent is:

$$u_x^0(0) = \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \end{bmatrix}$$

- From Step 8, to determine its strategy profile, $[s_1, s_2]$ that maximises its payoff, the agent solves the linear program:

  max $[s_1, s_2] \times u_x^0(0) \times [s_1^o, s_2^o]^T$ *s.t* $s_1 + s_2 = 1$ *and* $s_1, s_2 \geq 0$.

  The strategy profile thus obtained is denoted as $s_{max}^0(0)$ and the maximum payoff function is $u_{max}^n(0)$.

- The agent repeats steps $(1-7)$ for every time step n of the negotiation process and obtains the sequences $\{s_{max}^0(0), s_{max}^1(0), ...\}$ and $\{u_{max}^0(0), u_{max}^1(0), u_{max}^2(0), ...\}$

- The agent then repeatedly negotiates with the network operator negotiation during time instants $t = 0, 1, 2, 3, ...$ and obtains the sequences $\{s_{max}^0(0), s_{max}^0(1), s_{max}^0(2), ...\}$, $\{s_{max}^1(0), s_{max}^1(1), s_{max}^1(2), ...\}$, $\{s_{max}^3(0), s_{max}^3(1), ...\}$ and the corresponding payoff sequences.

Here, it is important to note that using this algorithm, the agent learns across successive negotiations and not within a single negotiation encounter. Therefore, the agent builds a model of operator's negotiation pattern over repeated negotiations and thus learns how it varies its strategies over time. Therefore this algorithm enables agents to develop a model of the operator's behaviour in this type of scenario. Now we move onto to examining the convergence properties of our algorithm.

4.1.2 Proof of Convergence

We claim that in repeated negotiations using our algorithm, the agent will learn to use the optimal strategy and earn maximum payoff at each stage of the negotiation process.

Now, we seek to prove this fact. In order to do so, however we need the following lemma.

Lemma 1: After a sufficiently large time T, the real probability distribution over the future rational play of a game is ϵ-close to what player i *believes* the distribution is [Kalai and Lehrer, 1993].

Here, ϵ-close implies that we can approach arbitrarily close to the actual distribution and rational play means that at each stage of the negotiation the players take the action that maximises their pay-offs. Having stated Lemma 1, we are now ready to prove our main result.

Theorem 4.2. *In the non-stationary negotiation process, the sequence of the n^{th} step strategies $\{s^n_{max}(0), ..., s^n_{max}(t), s^n_{max}(t+1), ...,\}$ and the corresponding sequence of n^{th} step payoff functions, $\{u^n_{max}(0), ..., u^n_{max}(t), u^n_{max}(t+1), ...,\}$, after a sufficiently large time T, are ϵ-close to the true optimal strategy and the corresponding maximum payoff function at the n^{th} step of the negotiation process.*

Proof. According to Lemma 1, in a systematic belief update process, the learner eventually comes arbitrarily close to the true distribution after a sufficiently large time T. Since the process by which our learning agent estimates the strategy of the opponent is constructed as a Bayesian belief process, the sequence of updated probabilities, $\{Pr\{H^0_n(t)|O^p_n(t)\}, ..., Pr\{H^i_n(t)|O^p_n(t)\}, ..., Pr\{H^k_n(t)|O^p_n(t)\}\}$ comes arbitrarily close to $\{0, ..., 1, ..., 0\}$ for some $i \in \{0, 1, 2, ..., k\}$ as $t \to T$. This implies that $H^i_n(t)$ is the true hypothesis. Therefore, $H^{new}_n(t) = \sum_k^{i=0}\{Pr\{H^i_n(t)|O^p_n(t)\} \times H^i_n(t)\}$, by construction, and, therefore, $H^{new}_n(t) \to H^i_n(t)$ as $t \to T$. Since the opponent determines its strategy at each step using $H^{new}_n(t)$ and since our agent determines its optimal strategy and the maximum payoff at each step in response to this updated opponent strategy, the result is proved. □

Thus analytically we have proved that our algorithm converges. Now, we move on describing formally the representation of $\mathbf{P^n(t)}$ in terms of matrices and analyse how to obtain a good representation of the space of possible opponent offers with the minimum number of matrices. By such an analysis we can constrain the search space of the adaptive algorithm and make it efficient even in environments where very little information is available about the opponent.

4.1.3 Representation of the Strategy Space

In order to do achieve such a representation, we define the opponent's Offer Generating Function as OGF. Now in order to estimate the opponent's OGF, we propose that the learning agent first models the range of this function formally as a *vector space* [Halmos, 1974]. Each element in this space can then be represented as a linear combination of independent basis vectors [3]. Then we construct a special set of matrices that are formed by combinations of these basis vectors $\mathbf{M} = M_1, M_2, ..., M_n$, each of which is a possible form of $\mathbf{P^n(t)}$. Thus by appropriately selecting different linear combinations of these matrices, we cover the entire space of possible distributions of $\mathbf{P^n(t)}$. A specific linear combination of matrices, therefore, specifies one possible distribution of $\mathbf{P^n(t)}$ and the agent needs to find the specific linear combination which is the closest approximation to the opponent's OGF. The degree of closeness of the two distributions ($\mathbf{P^n(t)}$ and the OGF) is determined by the similarity of the offers generated by them. This method of estimating a distribution is described as the Likelihood Estimation method. Therefore our agent, in successive negotiations, tries to build a model of $\mathbf{P^n(t)}$, using learning, that would generate the same offers as the opponent would using its own OGF. This knowledge enables the agent to strategically respond to its opponent's play.

In more detail, we consider two agents, say buyer X and seller Y, negotiating over

[3]This approximation reduces the search space, while still allowing us to represent the uncertainty in the problem.

price. The objective of the learning is to enable X (for instance) to estimate the OGF of Y and respond appropriately. Therefore the agent needs to use the incoming offers to form a model of the opponent's behaviour. To this end, we describe the procedure we have derived for this in detail in the next section.

4.1.4 Likelihood Estimation using Markov Chains

We describe the negotiation process from the point of view of the buyer. So X models the offer generating process of the seller Y. Now, X knows that Y may change its strategy during the course of the negotiation process. So, X models this process of change as a Markov chain, by defining the state space S as the set of strategies given by $S = \{s_0^o, s_1^o, ..., s_m^o\}$ and by representing the strategy change process by $\mathbf{P^n(t)}$. Therefore, Y switches between the strategies in S according to the transition probability matrix $\mathbf{P^n(t)}$, which varies with time. Given this, X has to determine a representation for $\mathbf{P^n(t)}$ that closely approximates the strategy profile of Y. To do this, initially, X randomly selects a sample probability distribution, $\alpha^{\mathbf{n}}(\mathbf{t})$ (n and t as usual represent the fact that this distribution changes at each step of the process and over time), over a finite set of matrices $\mathbf{M}$. $\alpha^{\mathbf{n}}(\mathbf{t})$ is the prior for our learning model. The objective is to update this prior and obtain the posterior distribution by observing the offers of Y. X calculates the probable strategy profile of Y using $\alpha^{\mathbf{n}}(\mathbf{t})$ as:

$$\mathbf{P^n(t)} = \sum_i \alpha_i \times \mathbf{M_i} \tag{4.5}$$

where $\alpha_i \in \alpha^{\mathbf{n}}(\mathbf{t})$ and $\mathbf{M_i} \in \mathbf{M}$. Once it has this profile, which is a row vector, in terms of the $\alpha_i' s$, X randomly chooses one strategy from the profile. We denote this strategy determined by the $\alpha_i' s$ as $f(\alpha_i)$. X now generates an offer, A_n, by using $\psi = f(\alpha_i)$ in equation 2.1. Therefore, A_n, is a function of the $\alpha_i' s$. Now we want this offer to match O_n, the offer of Y, as closely as possible. To this end, if we assume that the basis matrices, $\mathbf{M_i}$, are predetermined to represent the entire set of hypotheses about the opponent's model, we must then find the appropriate distribution of α_i that yields an A_n which best approximates the OGF of Y. Using Bayesian inference, this is done in the following manner.

In general, if M is the model that we wish to learn from the data D, then from section 2.3.2 we have:

$$P(M|D) = \frac{P(D|M) \times P(M)}{P(D)} \tag{4.6}$$

where $P(M|D)$ represents the posterior, $P(D|M)$ represents the likelihood and $P(M)$ the prior. Here, M is the opponent's model specified in terms of the $\alpha_i's$ and D represents the offers that the agent receives. For ease of mathematical analysis, we rewrite the above equation by dropping the denominator as:

$$P(M|D) \propto P(D|M) \times P(M) \tag{4.7}$$

Now we want the updated distribution to be a good estimate of the model. This is achieved by maximising the posterior (that is, $Max\{P(M|D)\}$). Again, for convenient analysis, we rewrite the above equation in terms of their logarithmic functions. The logarithmic function is monotonic so we can write $arg(Max\{P(M|D)\})$ as $arg(Max\{log(P(M|D))\}$. Applying the logarithmic function to both sides of equation 4.7, we have:

$$Max\{log(P(M|D)\} = kMax\{logP(D|M) \times P(M)\} \tag{4.8}$$

where k is some constant of proportionality. This equivalent to (removing k):

$$\begin{aligned} Min - log\{(P(M|D))\} &= Min\{-log\{P(D|M) \times P(M)\}\} \\ &= Min\{-logP(D|M) - logP(M)\} \end{aligned} \tag{4.9}$$

Then, if we assume that the $\alpha_i' s$ follow a Gaussian distribution initially, we can rewrite the RHS of the above equation in terms of A_n and O_n as [Tikhonov, 1963]:

$$-logP(D|M) - logP(M) = \frac{1}{z} * exp\{-|O_n - A_n|\} * exp\{\sum \frac{-\alpha_i^2}{2}\}\} \quad (4.10)$$

where z is some constant. This implies that:

$$-logP(M|D) = \frac{1}{z_p} * exp\{-|O_n - A_n|\} * exp\{\sum \frac{-\alpha_i^2}{2}\}\} \quad (4.11)$$

where $z_p = k \times z$. Therefore in order to maximise the posterior $P(M|D)$ we need to minimise $-P(M|D)$, which from the above equation is obtained as:

$$min\left\{(|O_n - A_n|) - \{\lambda\} \times \sum_i \alpha_i^2\right\} \quad (4.12)$$

Here, the term $|O_n - A_n|$ is called the empirical risk, $\sum_i \alpha_i^2$ is called the prior regularisation term and λ is a constant. From equation 4.12 we see that when $\lambda \rightarrow \infty$ the regularisation term dominates and, therefore, the solution to the problem approaches the prior distribution (that is, the unlearned initial distribution of the opponent's model) and when $\lambda \rightarrow 0$ the solution tends to the posterior distribution (that is, the true learned distribution of the opponent's model). Therefore λ controls the learning process in equation 4.12. Now, since the agent is learning within a Markov chain framework, the $\alpha_i' s$ have to satisfy an additional constraint called the partition of unity constraint. This is simply the condition that the $\alpha_i' s$ must sum to 1. This constraint also ensures that the $\alpha_i' s$ represent a true probability distribution. Incorporating this constraint in equation 4.12 we have the optimisation problem:

$$min\left\{|O_n - A_n| - \{\lambda\} \sum_i \alpha_i^2\right\}, s.t \quad \Sigma\alpha_i = 1 \quad (4.13)$$

In general Bayesian inference methods, the regularisation term is used to represent any prior information we have about the strategy distribution. For instance in equation 4.12, by using this form of the prior, for instance, we are saying that we know that some of the $\alpha_i's$ are zero and that the others are close to 1 (because the maximum value of $\sum_i \alpha_i^2$ can only be 1 and this can be achieved only if a few $\alpha_i's$ are close to 1). Therefore, by correctly choosing this term, we can make the solution move closer to the actual OGF of Y and it also gives us the flexibility of representing many different types of $Y's$ behaviour. Given this flexibility, this method of learning therefore becomes a powerful tool for studying a wide range of adaptive negotiation processes. Therefore this method has been developed as a generalisation of algorithm 2 to formalize the relationship between the prior distribution of the opponent's OGF and the actual observations of the opponent's offers. Therefore although we develop an algorithm based on the likelihood estimation method and prove its convergence, we do not implement it in this work. We now formally describe the algorithm based on this method.

4.1.5 Adaptive Negotiation using the Likelihood Estimation Method

We now present our algorithm for adaptive negotiations and describe its properties. To start with, we break up the entire negotiation process into a series of discrete steps, each of which corresponds to the event of receiving an offer from the opponent. To build the model of $Y's$ OGF, X, at the first time step n, of the process observes $Y's$ offer O_n. X then calculates $\mathbf{P^n(t)}$ using *a sample* $\alpha^{\mathbf{n}}(\mathbf{t})$ and randomly picks $f(\alpha_i)$. X then computes A_n using $f(\alpha_i)$ and equation 2.1 (see algorithm 3). X then solves equation 4.12 to determine the optimised $\alpha_i's$. It uses these $\alpha_i's$ to determine the updated $\mathbf{P^n(t)}$. It then moves on to the next step and repeats the above steps till the deadline T of Y is reached. It repeats this encounter with Y and reoptimises $\alpha_i's$ using new data. By using the updated $\alpha^{\mathbf{n}}(\mathbf{t})$ at each stage within a negotiation encounter, it learns within the encounter and by using new offers to reoptimise the $\alpha_i's$ by repeatedly negotiating with the opponent it learns across encounters as well. This dual learning makes the algorithm very effective in learning the opponent's OGF both quickly and correctly.

Moving on to discuss some of the properties of the negotiation process, we claim that the process is non-stationary because Y can change its strategy during the course of the negotiations. Also the OGF of Y can change during the negotiation because the parameters T and RP can change. This algorithm assumes that T and RP, though changing, are known at each stage of the process. However, the parameter ψ in equation 2.1 both changes and is unknown. Another important assumption of the model is that in repeated negotiations we assume that Y uses the same OGF to generate offers [4]. However, as the process of generating offers is probabilistic, we obtain new offers from the OGF and therefore more data for our model.

Now, using this algorithm we claim that the learning agent X eventually learns the true OGF of Y. In order to prove our claim we need to first state the Weak Law of Large Numbers from probability theory [Feller, 1968]:

The Weak Law of Large Numbers 1. Let $X_1, X_2, .., X_n$ be a sequence of n independently and identically distributed random variables. Each X_i has the same mean μ and standard deviation σ. Now define a new variable:

$$X = \frac{X_1 + X_2 + ... + X_n}{n}$$

Then as $n \to \infty$, the sample mean $< x >$ of X equals the population mean μ of each of the variables X_i.

Intuitively this theorem states that the mean of the means of *randomly* selected samples from a population, converge to the actual mean of the larger population as the number of samples increases. The convergence of our algorithm can be shown as a corollary to this theorem.

[4]When this is not the case the learning is less effective because it takes longer to learn more variations in the pattern of offer generation. However it is reasonable to assume that there is some common pattern of behaviour across negotiations.

Corollary 1. In the non-stationary negotiation process, if $\alpha^{\mathrm{n}}(\mathrm{t})$ represents the true distribution of strategy profiles of the opponent Y, and if in each negotiation encounter we use a randomly selected sample from $\alpha^{\mathrm{n}}(\mathrm{t})$ for learning in algorithm (3), then over repeated negotiations the mean of the sample distributions converges to the mean of $\alpha^{\mathrm{n}}(\mathrm{t})$ and therefore converges to the true OGF of its opponent.

Proof. We start with a sample of the true distribution $\alpha^{\mathrm{n}}(\mathrm{t})$ with which we generate the strategy profiles of the opponent Y. Then let each sample of this distribution be denoted by $\alpha_{\mathrm{i}}(\mathrm{t})$. Now, in repeated encounters during the learning process, we use different samples from the same distribution $\alpha^{\mathrm{n}}(\mathrm{t})$ for estimating the strategy profile of Y. Let the mean for each $\alpha_{\mathrm{i}}(\mathrm{t})$ be denoted by $\mu_i(t)$ and the sample mean be denoted by $\mu_{sample}(t)$. Also let the mean of $\alpha^{\mathrm{n}}(\mathrm{t})$ be denoted by μ. Then, by the Weak Law of Large Numbers, $\mu_{sample}(t) \rightarrow \mu$ as the $t \rightarrow \infty$. This implies that the model converges to the true $\alpha^{\mathrm{n}}(\mathrm{t})$ and to the true OGF of its opponent. □

Having described a formal method of estimating the opponent's strategy profile, we now proceed to give our empirical analysis in the next section.

Algorithm 3 The Adaptive Negotiation Algorithm using the Likelihood Estimation

```
1. for ($t = 0, 1, 2, ..., T$)
2.   for ($n = 0, 1, 2, ..., deadline$)
3.     input opponent offer $p \in Domain\{O_n(t)\}$
4.     initialize $\alpha^{\mathrm{n}}(\mathrm{t})$
5.     compute $\mathrm{P}^{\mathrm{n}}(\mathrm{t})$
6.     compute strategy set $S_n(t)$ from equation(4.3)
7.     select random strategy $f(\alpha_i)$ from $S_n(t)$
8.     compute $A_n(t)$ from $f(\alpha_i)$ and equation (2.1)
9.     compute optimised $\alpha_i^o s$ using equation (4.12)
10.    substitute $\alpha_i^o s$ for $\alpha_i s$
11.    compute $\mathrm{P}^{\mathrm{n}}(\mathrm{t})$ from equation (4.5)
12.  next $n$
13. next $t$
```

4.2 Empirical Evaluation

The aim of the algorithms detailed in this chapter is to find the correct representation of the opponent's strategy profile. Here the opponent is non-stationary in that its strategies are changing with time but it is not explicitly modelling and adapting to our agent's strategic play. Specifically, using algorithm 2 we do this by initially assuming a probability distribution over the set of possible matrices that represent the opponent's strategy profile and then by constantly updating this distribution until we are left with only one matrix (that is, all other matrices are multiplied by a zero probability). For this we start with an initial set of matrices and a random arbitrary distribution over this set of matrices. We consider a large number of possible representations of the opponent's behaviour and, therefore, a correspondingly large number of matrices to increase the probability that the true representation or the actual matrix that specifies the opponent's strategy profile is a part of this set. This is a crucial assumption of our algorithm, however it is not limiting because it is reasonable to assume that our agent has sufficient knowledge of its opponent and the environment to include the true strategy profile as one among many possibilities. Now we can measure the number of iterations in which the agent learns the true strategy profile of its opponent. To this end in our experiments we have varied:

- The number of matrices that are used to represent the strategy profile of the opponent.
- The initial probability distribution over the matrices. In BL terms this corresponds to the prior distribution (see section 2.3.2).
- The conditional probability distribution used to calculate the posterior distribution (see section 2.3.2).
- The utility function of the agent.

Also, here, we do not train the agent but directly deploy it in the negotiations. It however learns the true model of the opponent's behaviour over repeated negotiations. So we measure the number of negotiations it takes for the agent to learn the true model of the its opponent's behaviour. We have also compared the share in the profits (which is calculated as a percentage of the total utility it could have gained if the agent had known the strategy profile of its opponent at each stage of the negotiation process), that the agent earns using our algorithm with the profit-share earned by an agent using a standard BL algorithm from the literature (specifically it uses the model of Zeng and Sycara — see section 2.4).

In more detail, figure 4.1 shows the number of negotiations in which the agent learned the true model of the opponent's behaviour in over 200 experiments. Figure 4.2 shows the average time each negotiation took. Table 4.1 shows the mean μ_1 and standard deviation σ_1 of the number of negotiations. On an average it can be seen that it takes an average of 60 encounters for the agent to learn the true model of opponent's behaviour. Here $\sigma_1 = 6$ encounters and 98% of this population is within 2σ distance from the mean. We would expect, that during the lifetime of a complex multi-agent system, the agents interact with each other hundreds of times. In this context an average of 60 encounters to learn the true model of opponent behaviour is, we believe, a good performance. Table 4.2 shows the mean μ_2 and standard deviation σ_2 for the time taken for each negotiation. Here, 97.5% of the population is within 2σ distance from the mean. It takes on average 23 seconds to conclude each negotiation encounter. This is again a very good performance. Naturally the time to conclude a negotiation can increase depending on the deadlines of the agents, however we believe that typically in the types of domain that we are interested in the time of interaction would be short.

Having examined the time taken to learn the opponent's model we now turn our attention to the profits earned by the adaptive agent using algorithm 2. Figure 4.3 shows the adaptive agent's share of the profits compared to that of a non-adaptive agent and table 4.3 shows the mean and standard deviation of the profits. As can be seen, using our algorithm the agent achieves nearly 83% of the profits while negotiating with an adaptive opponent as compared to the stationary one that earns only 54%. This a direct result of using a non-stationary algorithm that captures the change in the opponent's behaviour pattern as a function of time. Our algorithm therefore develops a more accurate model of the opponent's behaviour than the stationary algorithm and hence gains a larger share of the profits. This is a clear advantage of using our algorithm and corresponds well with our expectations.

Now, in order to show how much more effective than random search our algorithm is we present the following analysis. If we assume that there are k hypotheses for $\mathbf{P^n(t)}$, then in the random case the probability of finding the true hypothesis is always $1/k$ (i.e., this probability does not improve with time). However, the convergence of our algorithm is guaranteed by theorem 2 and therefore our estimation of the true $\mathbf{P^n(t)}$ improves with each iteration and it will eventually converge. Taking this, and the fact that the algorithm converges on an average within 1380 seconds (see tables 4.1 and 4.2) by varying different parameters, we claim that our algorithm is k times more effective than random estimation. Therefore, even in the case when there are only 2 hypotheses at each step n for $\mathbf{P^n(t)}$ our algorithm is 200% more effective than random estimation. Obviously, as we increase the number of hypotheses, which allows for a more general representation of $\mathbf{P^n(t)}$ and therefore of the uncertainty in the problem, the effectiveness of our algorithm over random estimation increases proportionally.

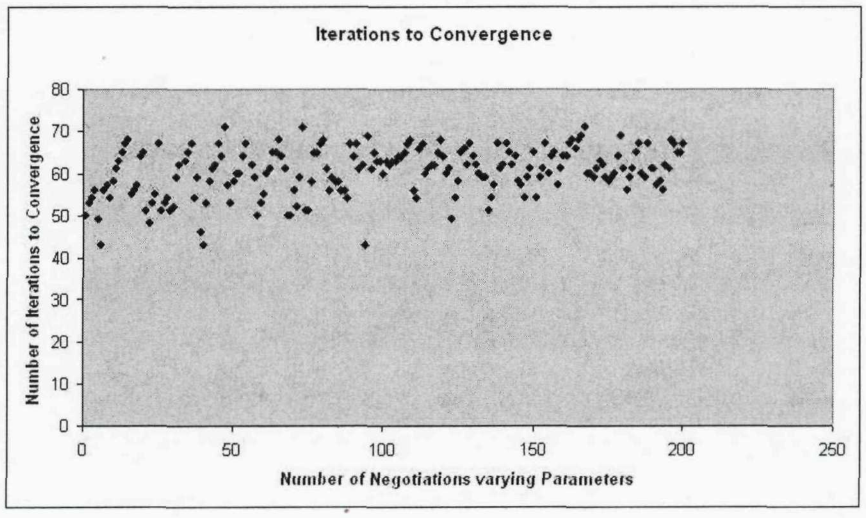

FIGURE 4.1: Convergence to the true opponent model over repeated negotiations.

Negotiations	Mean	Std Dev
200	60.32	5.69

TABLE 4.1: Time taken for convergence to the true opponent model.

Negotiations	Mean	Std Dev
200	23.73	2.12

TABLE 4.2: Time for Negotiation.

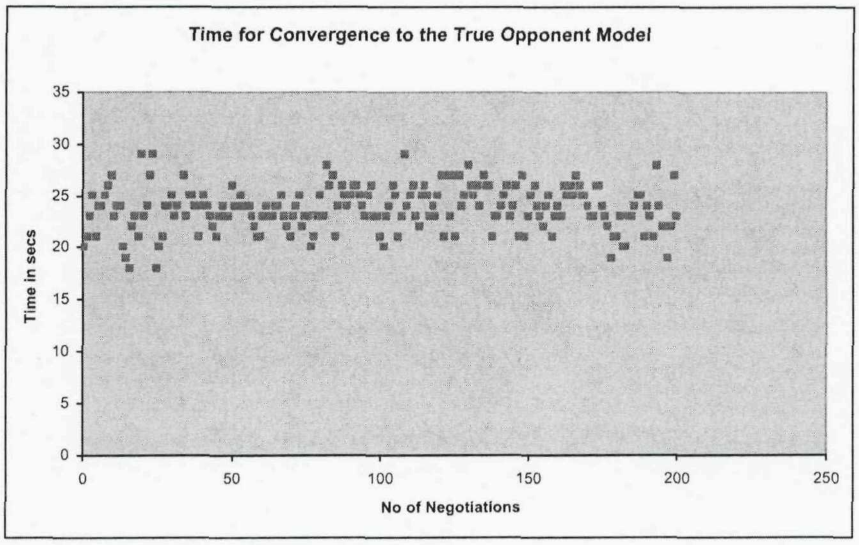

FIGURE 4.2: Time taken for each negotiation.

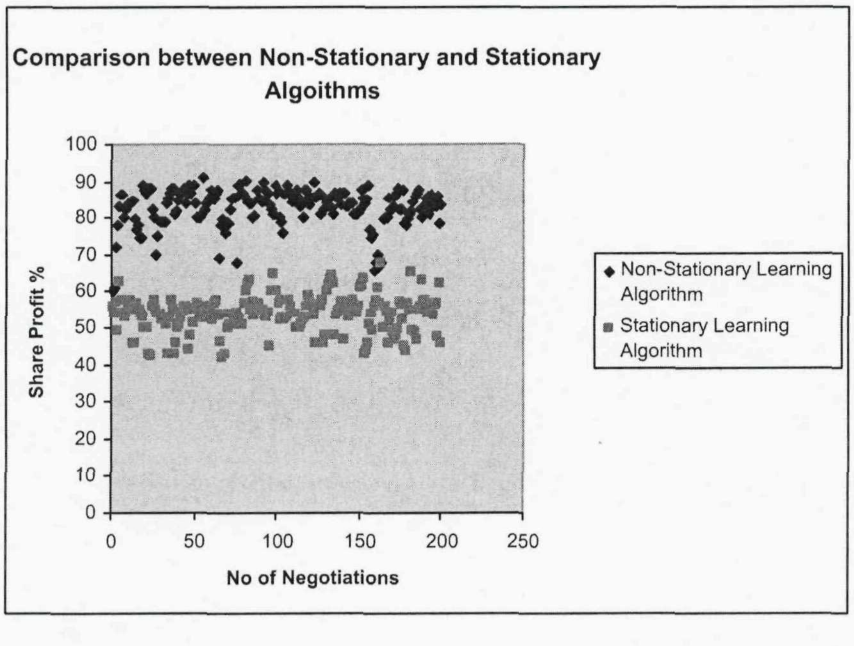

FIGURE 4.3: Comparison between non-stationary and stationary Algorithms: Share in profits.

4.3 Summary

In this chapter we describe a new framework that we have developed, using Markov chains, for studying negotiation in non-stationary environments. This is a rigorous theoretical model for learning in negotiations in non-stationary environments which can be used to study decision making in many stochastic systems. Within this framework, we

Negotiations	Mean	Std Dev
200	82.97	5.23

TABLE 4.3: Share in the Profits.

have derived, for the first time, an important result for non-stationary Markov chains that computes the distribution of the random variable, which defines it, at any future step of the process given its initial probability distribution and the transition probability matrices at each step of the process. Then, using this framework, we have developed an algorithm to learn a strategy in response to a non-stationary opponent's play and proved that it converges to the optimal strategy in repeated negotiations. By so doing, we have satisfied requirement 6. Unlike previous work in this area, our algorithm does not assume knowledge of the opponent's strategy profile and, as such, is a powerful tool to analyse negotiations in real world environments where such uncertainty is common. Our algorithm is also explicitly designed to deal with cases in which the strategy profile is not only unknown, but changes with time during the course of the negotiation process itself. The only assumptions that we make are:

- The agent has sufficient knowledge of its opponent to consider its true strategy profile amongst many possibilities at each step of the negotiation process.
- The agent knows its own utility function, although this may change during the course of the encounter.

This significantly extends the state of the art in the field of automated negotiations in non-stationary, real world environments. For such cases, we have proved, analytically, that our algorithm converges. Our empirical results prove that the algorithm converges within reasonable timeframes to the true hypothesis under varying conditions including changes in the utility function (thus satisfying our requirement 3). Also, in our algorithm, the states of the Markov chain represent the strategies that are available to the opponent and the actual uncertainty in the problem is represented by the number of hypotheses. Therefore we need not increase the number of states to represent greater uncertainty in the domain. The algorithm is also vastly more accurate than random estimation.

We have also presented a formal model for the likelihood function in the Bayesian belief update process, using the structure of the Markov chain, which helps us to reduce the computation effort involved in updating the agent's knowledge even in complicated real world scenarios. Thus algorithm 3 to scales well because it uses a linear combination of a small set of matrices to cover the entire strategy space of the opponent. As such, we only need as many vectors to span the entire space as there are degrees of freedom in the optimisation problem described by equation 4.13. Therefore, even if there are a large number of strategic choices for the opponent, because of the constraints imposed on the problem by the Markov chain framework (the rows of the matrices should sum to 1), using these vectors, this would translate to a small number of matrices. By doing this, we have converted the problem of determining an unknown opponent's strategy from an infinite space of possibilities to a low dimensional optimisation problem. This is a significant step towards solving more general and complicated negotiation problems in dynamic scenarios.

Now using algorithm 2 the agent can respond well to the opponent, however this algorithm is not explicitly designed to adapt to changes in the environment. Given this, and in order to satisfy all our requirements, we need to use both the RL algorithm of Chapter 3 and the BL algorithms of this chapter in conjunction. In the next chapter we examine in detail how this can be achieved.

Chapter 5

Comparison between Reinforcement and Bayesian Learning Methods

So far in this work so far we have described how RL and BL techniques can be used for automated negotiation in dynamic environments and against adaptive opponents respectively. Specifically, we have proposed an algorithm based on RL techniques that can be used for adapting to changes like deadlines and resource availability, whereas the BL algorithm is best suited for modelling opponent behaviour. Against this background, in this chapter we seek to compare these two learning methods by using them together in an archetypal dynamic negotiation test setting. In more detail, we have simulated a dynamic negotiation environment where the negotiation parameters change and have simultaneously allowed the agent to interact with its strategic opponent. Our aim in so doing is to determine the conditions under which the algorithms perform well and also to examine their limitations. Thus our analysis in this chapter enables the agents to choose the correct algorithm for different dynamic negotiations scenarios.

In more detail, the chapter is structured as follows: in section 5.1 we describe the test environment, in section 5.2 we analyse the performance of the methods and in section 5.3 we conclude.

5.1 Test Environment

In this section we describe in detail the test environment that we have used to evaluate our algorithms. In the negotiation environment we have two agents negotiating with one another on the price of an item. To represent the environmental changes we assume:

- The bandwidth or network resources available (RA) for the negotiations is subject to change.
- The time available for the negotiation, represented by the deadlines (T^a) of the agents, are subject to change.
- The financial resources available for the bargaining, represented by the reserve price (RP^a) of the agents, is subject to change.

We also assume that the other agent in the negotiation process is adapting to these environmental conditions. Because the two agents are negotiating in the same environment, we assume that the RA is common to both the agents although the other parameters can vary. Therefore we now have a total of 5 parameters RA, T^a, RP^a, T^b and RP^b. We also need to determine how the agent's opponent changes its strategies during the course of the negotiation process. As before, our agent does not know its opponent's parameters and does not try explicitly to learn them. Instead, it learns from repeated interactions to build a profile of its opponent's play. Therefore in this setting the agent is simultaneously adapting both to changes in the environment and to changes in its opponent's behaviour.

To place this in context, we explicitly relate this setting back to those of the previous two chapters. In Chapter 3, we experimented by simulating the same environmental changes as those described above. However, we did not assume that the opponent was also adapting to these changes. Thus the agent was only adapting to the dynamism in the environment and not to an adaptive opponent. Again, in Chapter 4, the opponent was changing its strategy, however this was not in response to the changes that were occurring in the environment. Therefore we did not change these parameters while observing the agent's behaviour. To sum up, then, in chapter 3, the agent specifically used the reinforcement learning algorithm (algorithm 1), to adapt to dynamic changes without considering the play of its opponent and in chapter 4 the agent adapted its strategies in response to its opponent. However, these changes were not explicitly modelled as being in response to environmental changes. Therefore, in these cases the agent's negotiation parameters did not change, although its utility function changed during the encounter. Thus, the agent used the Bayesian learning algorithm (algorithm 2) to model its opponent's behaviour and develop best response strategies. In this chapter, however, the agent adapts to both the dynamism in the environment and to its adaptive opponent.

In more detail, the value of RA is generated by a random function before every offer is made. The agents check this value and accordingly set their internal value of RA to $(high, low)$. For instance, the RA can be represented as a sinusoidal function which ensures that resource availability fluctuates during the course of the encounters[1]. Similarly, we also have a random function which tells the agent its current T value. However this random function operated between a specific range of T values. We believe this is reasonable because the deadline is not expected to change in a completely arbitrary manner. Based on the current T value, the agent then calculates how much time it has left to complete the negotiation process and accordingly sets the state to $(high, low)$. Finally, we have a random function that generates the RP for the agent. Like the deadlines, the random function chooses this value from a specific range of values. The agent then checks the amount of money it has at its disposal and sets its RP value to $(high, low)$.

As the negotiation progresses, the agent moves between these states. These state transitions are governed by P_n^a and each action is rewarded according to R_n^a (see chapter 3). These conditions are similar to those described in chapter 3 however now, *both* the

[1]We could have used any periodic function. In particular, the results are independent of the type of RA, we only need a function whose values are fluctuating.

agents adapt to these changes. In contrast, in chapter 4, we assumed that the opponent changes its strategies, however in that case the change was not due to changes in the environment. In this case, the agents have to adapt to these conditions and reach an agreement. Also they have to obtain a good share of the profits as represented by the utility function. As before, the utility function is specified at each stage of the negotiation process and represents the utility that the agent would gain in adopting a particular strategy profile in response to that of its opponent's strategy profile.

We ran 200 simulations by varying these conditions to obtain an accurate performance analysis of our algorithm. In particular, we wish to examine the performance of our algorithms when changes in the environment contribute to changes in the negotiation parameters and also when the opponent's strategies are changing. Thus for algorithm 1, the input parameters are RA, T and RP. Algorithm 1 then recommends a value of ψ to use based on solving the RL problem. Now the agent is also given a set of possible strategy profiles of opponent, its own utility function and the initial prior distribution over the strategy profile of the opponent. Then using algorithm 2, the agent computes the strategy profile that maximises its utility function. Therefore algorithm 2 recommends a strategy profile for the agent in response to the opponent's strategy profile. Now in order to compare the performance of the two algorithms we run both of them together in these conditions and examine their performance.

5.2 Performance Analysis

In this section we compare the performance of two agents, one using algorithm 1 and the other using algorithm 2. Specifically, we compare the number of agreements reached using these two algorithms and the utility share they obtain. Based on these two metrics we make recommendations for using these techniques appropriately and also make suggestions for integrating the reward schemes of both techniques in order to develop a single integrated algorithm that would adapt to both environmental conditions and adaptive opponents. First, however, we consider the number of agreements that are achieved using our RL and BL algorithms (see figure 5.1).

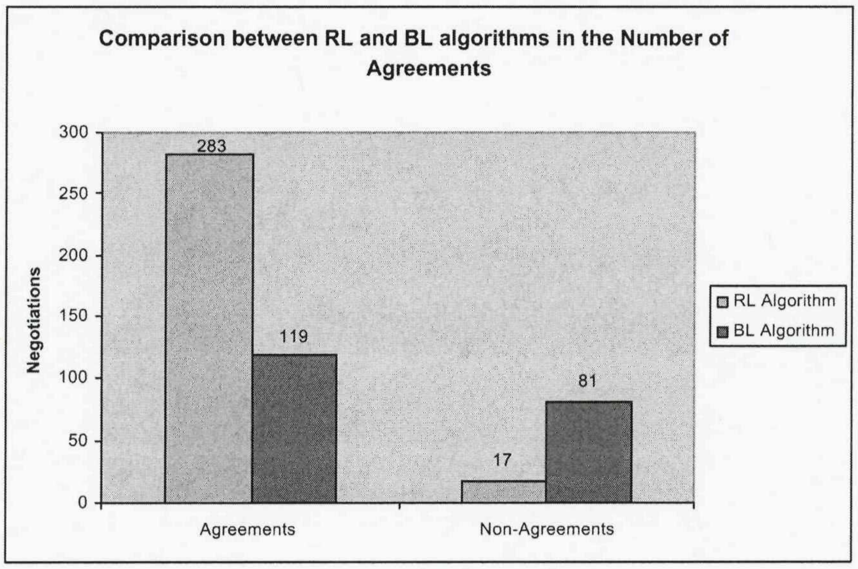

FIGURE 5.1: Comparison between RL and BL algorithms in terms of the number of Agreements.

For these experiments we varied the resource availability RA, the reservation price RP and the deadline T of the agent. Both the agents adapt to these conditions using algorithm 1. As stated previously, this algorithm recommends a strategy based on changes in resource, finance and time availability and hence the agents are able to adapt their strategies to reach an agreement before these resources are exhausted. This results in a high percentage of the negotiations ending in an agreement using algorithm 1. Algorithm 2 recommends a strategy profile based on the model it forms of its opponent. In more detail, the strategy profile that algorithm 2 recommends depends on the current strategy profile of its opponent and the current utility function. This utility function is designed to reward the agent for choosing a strategy that is a best response to the opponent's strategy and not a response to the environmental conditions. Therefore using algorithm 1 the percentage of negotiations that end in agreement is high (90.5%) (see table 5.1). Whereas using algorithm 2 this percentage is only 59.5%. However this figure still compares favourably with other non-adaptive algorithms (see chapter 3). So when the changes in the negotiation parameters are very dynamic we recommend using the RL algorithm. This is because using the Bayesian algorithm, the agent is modelling changes in its opponent's behaviour. Now, when this change is due to environmental conditions, the agent is implicitly adapting to them. Therefore algorithm 2 is able to react better than other non-adaptive algorithms to these changes.

Next, we compare the share in the utility that is obtained by the agents using the two algorithms (see figure 5.2). This is computed as a percentage of the total utility that could have been earned if the agent knew the opponent's strategy profile at each stage of the negotiations. For the agent using the BL algorithm, the strategy itself is chosen to maximise the utility function. For the agent using RL, the share is calculated based on the same utility function, however it uses a strategy that is recommended as a response to changes in the environment.

In terms of results, the BL algorithm enables the agents to earn a very high profit share of 88.7% and using the RL algorithm the agent earns a profit share of 66.3% (see table 5.2). This because using algorithm 2 the agent is actually developing a strategy, based on its model of opponent behaviour, that gives it maximum utility. Therefore it is expected that the agent obtains a larger share of the profits than its opponent using this algorithm. However even this figure compares favourably with the share of profits obtained by the stationary algorithm we examined in chapter 4. This is because the agent is still adaptive using algorithm 1 and recognises changes in the environment correctly. This guarantees the agent a reasonable payoff during the negotiations. Therefore when we wish the agent to bargain aggressively and gain a larger share of the profits we would recommend using the BL algorithm.

Algorithm	Negotiations	% of Agreements Reached
RL	300	90
BL	300	59.5

TABLE 5.1: Comparison Table between RL and BL algorithms in terms of agreements Reached.

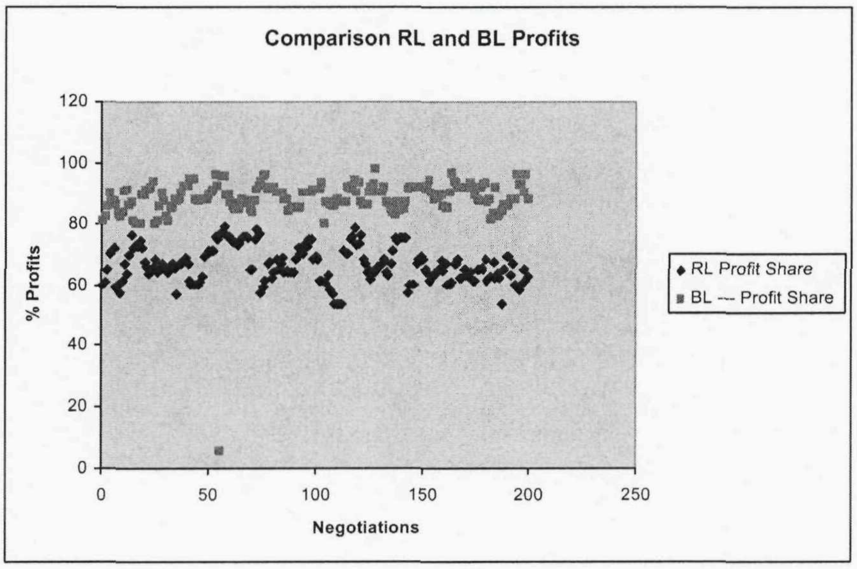

FIGURE 5.2: Comparison between RL and BL algorithms in terms of profits earned.

Algorithm	Negotiations	Mean	Std Dev
RL	200	66.3	5.7
BL	200	88.7	7.03

TABLE 5.2: Comparison Table between RL and BL algorithms in terms of Profits earned.

5.3 Summary

Having tested the our algorithms under different conditions, we now detail their main strengths and also discuss some of their limitations. To recap, we have tested the algorithms when both environmental conditions change and when the opponent is also adapting to changes in the environment. We have compared the number of agreements that have been reached using both the algorithms and the percentage of profits earned by the algorithms. We have chosen these two metrics because they represent the key success factors in many negotiation encounters [Rosenschein and Zlotkin, 1994]. In particular, we are actually measuring how the agent adapts to changes in its negotiation parameters when these can change dynamically and how the agent models the strategies of a non-stationary opponent.

We have seen that the agent using the RL algorithm achieves a very high percentage of successful agreements (90%). Thus by using the RL algorithm the agent is able recognise sudden changes in deadlines, resource availability and reserve prices. This makes the agent a smart negotiator in many types of dynamic multi-agent scenarios. However in these circumstances the agent using the BL algorithm performs less well. Specifically, only 60% of the negotiations end in an agreement in such cases. Nevertheless, it should be borne in mind that this still compares favourably with other non-adaptive and heuristic algorithms (where only 49% and 56% respectively of the negotiations end in agreement). Thus when the negotiation parameters are subject to frequent change, the RL algorithm outperforms the other algorithms.

Turning now to the share in the profits obtained using the algorithms and considering only those negotiations that end in agreement. The agent using the BL algorithm, by making an accurate model of its opponent's behaviour, garners a large share (88%) of the profits. However the agent using the RL algorithm gains only 66.3% of the profits. This difference occurs because the strategy that this algorithm recommends is mainly geared towards adapting to changes in the environment and reaching an agreement under changing circumstances. However this algorithm also performs better than the non-adaptive agent described in chapter 4 which gains only 54% of the profits. Thus in order to bargain aggressively and gain a large share of the profits available, we recommend that the agent should use the BL algorithm.

Now, if we assume that the opponent is also using the same algorithm to adapt to changes in our agent's behaviour or in other words if our agent is negotiating with a version of itself then it is our expectation that the profits will be evenly divided between the two. If the problem is set up as a competitive game then the 50% share will represent a unique Nash equilibrium solution. This however needs to be substantiated by future theoretical and empirical analysis.

To summarise, our algorithms are successful in adapting to the different dynamic conditions that prevail in many multi-agent systems.

Chapter 6

Conclusions

In this chapter we present our conclusions and outline the areas of further investigation that follow on from these studies.

In this thesis we have developed a learning based mechanism for automated, adaptive negotiation. Specifically, we have considered bilateral agents negotiating in multi-agent settings. The agents have to adapt to changes in their environment which contribute to changes in their negotiation parameters like deadlines and reserve prices (see Chapter 3) and they also have to adapt to changes in their opponent's strategy (see Chapter 4). Given this background, the work in thesis has examined in detail how these twin objectives can be achieved. After a thorough analysis of the literature in negotiation theory and multi-agent systems, we have identified Reinforcement Learning and Bayesian Learning as suitable techniques for learning in the dynamic and uncertain environments that we are interested in.

In more detail, for adapting to environmental conditions the technique we employ is based on Markov Decision Processes approach where the agent progresses from state to state by choosing appropriate actions. Thus the agent learns an appropriate mapping between states and actions through a reward scheme that differentiates between good and bad choices. To achieve this, we have set up our negotiation problem as a Markov negotiation problem and have developed a novel value iteration based algorithm to determine an optimal policy mapping between states and actions. Using this algorithm, the agent autonomously learns to respond to changes in the environment.

For our specific problem, we have changed the deadline, reservation prices and resource availability and have evaluated how the agent responds to these changes. We have also compared our algorithm to other standard techniques for solving negotiation problems and determined that it enables the agent to make more agreements. It is also important to note that our agent is adapting to non-stationary changes in the environment; that is, the underlying probability distribution that governs the transitions between the states is a function of time. This is a significant extension of the state of the art in negotiation theory and enables us to deal with negotiation encounters in many dynamic multi-agent systems. In short in this line of work we have developed an efficient algorithm by which the agents adapt strategies autonomously when they have only probabilistic knowledge of the dynamics of the environment. We have also shown theoretically that the algorithm converges to the optimal solution in a sufficiently large number of iterations.

Next, in order to adapt to changes in the opponent's behaviour, we have developed an algorithm based on Bayesian Learning methods. This algorithm models the opponent's behaviour using a probability distribution and updates this distribution using the information that it gains by negotiating repeatedly with its opponent. Thus by using Bayesian techniques the agent refines its model of the opponent's strategy profile and learns the true model of opponent behaviour. As it models the behaviour it also develops a response to the opponent's strategy. This response converges to the optimal as the agent's model converges to the true opponent behaviour model. Again, our agent deals with an adaptive, non-stationary opponent meaning our algorithm represents an extension of the state of the art in both Bayesian Learning algorithms and the theory of automated negotiations. We have empirically evaluated this algorithm against stationary adaptive algorithms based on BL techniques and have shown that the utility earned is higher than that earned using the benchmark algorithms. Therefore we claim that we have developed an efficient algorithm for responding to non-stationary opponents which can be deployed in many multi-agent systems.

Finally, we have compared the performance of the two learning algorithms in situations in which both the negotiation parameters and the strategies of the opponents change. Based on our experiments, we recommend that the reinforcement learning technique be used for conditions in which the environment is very dynamic and the Bayesian learning technique be used for when the opponent is aggressive and changes its strategies frequently during the course of the encounter.

To sum up, by using the learning algorithms we have developed our agent is able to negotiate autonomously and make decisions on its own. To function effectively, it only requires a reward scheme for the reinforcement algorithm and some information about the interaction between the opponent and the environment for the Bayesian algorithm. This therefore satisfies our requirement 1. The agent also functions at all times in the absence of complete information about the environment and its opponent meaning that requirement 2 is satisfied. In particular, algorithm 1 is specifically designed to deal with changes in the negotiation parameters of the agent. Therefore we satisfy requirement 3 of our model. Algorithm 2, on the other hand explicitly models opponent behaviour and is designed to cope with changes in its behaviour. This satisfies requirement 5. By doing this, we are also implicitly adapting to changes in the opponent's parameters as the opponent's final strategy is a result of its adaptation to the environmental conditions and changes in its own parameters. Therefore we satisfy requirement 4. Our reinforcement learning algorithm is also designed to strive for agreements during negotiations and we have shown, empirically, that it is successful in achieving this (see chapter 3). This satisfies requirement 6.

Against this background, we legitimately claim to have achieved all the research objectives laid out in chapter 1. We have built an effective, adaptive, automated negotiation model for conditions that prevail in many multi-agent systems and in so doing we have extended the state of art in the areas of:

- Machine learning
- Automated negotiations
- Interactions in multi-agent systems.

6.1 Future Research Directions

The fundamental research that has been undertaken in this thesis can be used as a point of departure for further research in a number of areas. Essentially this work has provided researchers with a new method of looking at interactions between agents. We have built separate algorithms, a reinforcement algorithm that deals mainly with changes the agent's own parameters and a Bayesian learning algorithm that essentially deals with changes in its opponent's behaviour.

From this, a number of future research directions naturally follow:

- Developing an integrated reward scheme for algorithms 1 and 2 in order to develop a single algorithm for dealing simultaneously with changes in the negotiation parameters and the opponent's strategy. For this, a method needs to be devised by which the numerical reward signal of the RL process can be aligned with the utility function used in algorithm 2. With this in place, a single algorithm would then enable the agent to adapt to the different changes described in this work.
- Extending these algorithms to deal with multilateral negotiations (that is negotiations involving more than one other party). Such negotiations are important when, for example the salesman has to negotiate with more than one service provider for the broadcast of his programs to get a good deal in scenario 1 or when the retail

manager has more than one network operator in her area who can provide her with services in scenario 2. To deal with this extension, we could build from the Markov chain framework and adopt the idea of using basis vectors to reduce the state space representation used in this thesis so that it can account for a model of its opponent's behaviour. However, any algorithm designed for multilateral settings would need to use a different protocol than the alternating offers one we use in this work because this only permits two parties to participate.

- Extending this work to deal with multiple issues, (that is, negotiation about issues other than price). Such negotiations are important when, for instance, the salesman can bargain for both the price and the quality of transmission in scenario 1 or when the retail manager has a choice of several services and therefore needs to bargain both on the cost of the services and the number of services that can be provided to her in scenario 2. This, we believe, would be a relatively straightforward extension of the ideas developed in this work. Specifically, the utility function would have to be modified to reflect the benefit gained from bargaining across different issues. But the main structure of the algorithms can be retained; This means keeping the changes in the environment defined as states of the non-stationary Markov chain and exploiting their properties to determine future states and the Bayesian method of learning from experience.

- Benchmarking the algorithms we have developed against human negotiations (as Das et al. [2001] did for the continuous double auction format). Such studies, would enable the relative benefits of automated and human negotiations to be compared in order to determine their relative strengths and weaknesses. Such evaluations are important since wide scale adoption of these technologies is not going to happen until compelling evidence of their superior performance is available.

Bibliography

Bernardo, J. and Smith, A. (1994). *Bayesian Theory.* John Wiley and Sons.

Binmore, K. and Samuelson, L. (2001). Evolution and mixed strategies. *Games and Economic Behaviour, 34(2), 200-226.*

Binmore, K. and Vulcan, N. (1997). Applying game theory to automated negotiation. *DICMAS Workshop on Economics, Game Theory and the Internet(1997), Rutgers University, NJ.*

Blickle, T. and Thiele, L. (1995). A comparison of selection schemes used in genetic algorithms. *Technical Report 11, TIK.*

Bui, H., Venkatesh, S., and Kieronska, D. (1999). Learning other agent's prefernces in multi-agent negotiation using bayesian classifier. *International Journal of Cooperative Information Systems 8(4), 275-293.*

Chavez, A., Dreilinger, D., Guttman, R., and Maes, P. (1997). A real-life experiment in creating a agent marketplace. *Proceedings of the Second International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, Blackpool, UK, 159-178.*

Christopher, M. (2005). *Logistics and Supply Chain Management: Creating Value-Added Networks.* Financial Times Prentice Hall.

Claus, C. and Boutilier, B. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. *Fifteenth National Conference on Artificial Intelligence(AAAI), Menlo Park, CA, USA, 746-752.*

Das, R., Hanson, J., Kephart, J., and Tesauro, G. (2001). Agent-human interactions in the continuous double auction. *Proc. of Seventeenth International Joint Conference on Artificial Intelligence, Seattle, WA, USA, 1169-1176.*

Emery-Montemerlo, R., Gordon, G., Schneider, J., and Thrun, S. (2004). Approximate solutions for partially observable stochastic games with common payoffs. *Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), Washington DC, USA, 136-143.*

Faratin, P., Sierra, C., and Jennings, N. (1998). Negotiation decision functions for autonomous agents. *International Journal for Robotics and Autonomous Systems 24 (3-4) 159-182.*

Fatima, S., Woolridge, M., and Jennings, N. (1997). An agenda-based framework for multi-issue negotiation. *Artificial Intelligence Journal 152(1):1-45.*

Fatima, S., Woolridge, M., and Jennings, N. (2002). Multi-issue negotiation under time constraints. *Proc. of First International Conference on Autonomous Agents, Bologna, Italy, 143-150.*

Fatima, S., Woolridge, M., and Jennings, N. (2004). Bargaining with incomplete information. *Annals of Mathematics and Artificial Intelligence, 44 (3) 207-232.*

Feller, W. (1968). *An Introduction to Probability Theory and its Applications:v.1.* John Wiley and Sons.

Filar, J. and Vrieze, K. (1996). *Competitive Markov Decision Processes.* Springer.

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

Garcia, F. and Ndiaye, S. (1998). A learning rate analysis of reinforcement learning algorithms in finite horizon. *Proceedings of the 11th International Conference on Machine Learning (ML-98), Madison, Wisconsin, USA, 215-223.*

Gilks, W., Richardson, S., and Spiegelhalter, D. (1995). *Markov chain Monte Carlo in Practice.* Chapman and Hall.

Halmos, P. (1974). *Finite Dimensional Vector Spaces.* Springer.

He, M., Jennings, N., and Leung, H. (2003). On agent-mediated electronic commerce. *IEEE Trans on Knowledge and Data Engineering, 15(4): 985-1003.*

Holland, J. (1992). *Adaptation in Natural and Artificial Systems.* MIT Press.

Howard, R. (1960). *Dynamic Programming and Markov Processes.* The MIT Press.

Hu, J. and Wellman, M. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. *Proceedings of the 11th International Conference on Machine Learning, Madison, Wisconsin, USA, 242-250.*

Jennings, N., Corera, J., and Laresgoiti, I. (1995). Developing industrial multi-agent systems. *(Invited Paper) First International Conference on Multi-Agent Systems(ICMAS-95), San Francisco,CA , 423-430.*

Jennings, N., Faratin, P., Lomuscio, A., Parsons, S., Sierra, C., and Woolridge, M. (2001). Automated negotiation:prospects, methods and challenges. *International Journal of Group Decision and Negotiation, 10(2): 199-215.*

Kaelbling, L., Littman, M., and Moore, A. (1996). Reinforcement learning: A survey. *Journal of Aritificial Intelligence, 4: 237-285.*

Kaelbling, L., Littmann, M., and Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence, 101(1-2), 99-134.*

Kalai, E. and Lehrer, E. (1993). Rational learning leads to nash equilibrium. *Econometrica, 61(5),1019-1045.*

Kannai, Y. (1977). Concavifiability and constructions of concave utility functions. *Journal of Mathematical Economics 4,1-56.*

Karlin, S. and Taylor, H. (1974). *First Course in Stochastic Processes.* Academic Press.

Keeney, R. and Raiffa, H. (1976). *Decisions with Multiple Objectives: Prefernces and Value Tradeoffs.* New York:John Wiley.

Kraus, S. (2001). *Strategic Negotiation in Multi-Agent Environments.* MIT Press.

Kraus, S. and Subrahmanian, V. (1995). Multiagent reasoning with probability, time, and beliefs. *International Journal of Intelligent Systems 10(5): 459-499.*

Kulkarni, V. (1996). *Modelling and Analysis of Stochastic Systems.* Chapman Hall/CRC.

Littman, M. (1994). Markov games as a framework for multi-agent reinforcement learning. *Proceedings of the 11th International Conference on Machine Learning, San Francisco, CA, USA, 157-163.*

Luck, M., McBurney, P., and Preist, C. (2003). *Agent Technology: Enabling Next Generation Computing (A Roadmap for Agent Based Computing).* AgentLink.

Matos, N., Sierra, C., and Jennings, N. (1998). Determining successful negotiation strategies: An evolutionary approach. *Proc. 3rd International Conference on Multi-Agent Systems (ICMAS-98), Paris, France, 182-189.*

Milgrom, P. (1982). A theory of auctions and competitive bidding. *Econometrica, 50(5), 1089-1122.*

MVCE, P. (2004). Deliverable pde 2.1, http://www.mobilevce.com/.

Narayanan, V. and Jennings, N. (2005). An adaptive bilateral negotiation model for e-commerce settings. *Proc. 7th Int. IEEE Conference on E-Commerce Technology, Munich, Germany, 34-39.*

Nash, J. (1950). The bargaining problem. *Econometrica, 18(2), 155-162.*

Nash, J. (1951). Two-person cooperative games. *The Annals of Mathematics, 54(2), 286-295.*

Nash, J. (1953). Two-person cooperative games. *Econometrica, 21(1), 128-140.*

Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behaviour.* Princeton University Press.

Oliver, J. (1996). A machine-learning approach to automated negotiation and prospects for electronic commerce. *Journal of Management Information Systems, 13(3), 83-112.*

Papoulis, A. (1984). *Probability, Random Variables, and Stocahstic Processes.* New York: McGRaw-Hill.

Paurobally, S., Turner, P., and Jennings, N. (2003). Towards automating negotiation for m-services. *Proceedings 5th International Workshop on Agent-Mediated E-Commerce, Melbourne, Australia, 124-131.*

Puterman, M. (1994). *Markov Decision Processes: Discrete stochastic Dynamic Programming.* New York:John Wiley.

Rosenschein, J. and Zlotkin, G. (1994). *Rules of Encounter.* MIT Press.

Rubinstein, A. (1982). Perfect bargaining in a bargaining model. *Econometrica,50:97-110.*

Shoham, Y., Powers, R., and Grenager, T. (2007). If multi-agent learning is the answer what is the question? *Artificial Intelligence 171(7), 365-377.*

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

Sycara, K. (1988). Resolving goal conflicts via negotiation. *Proc. 7th National Conference on Artifical Intelligence, St.Paul, Minnesota, USA, 245-250.*

Tikhonov, A. (1963). Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl., 4, 1035-1038.*

Watkins, C. and Dayan, P. (2004). Q-learning. *Machine Learning, 8:279-292.*

Weinberg, M. and Rosenschein, J. (2004). Best-response multiagent learning in non-stationary environments. *The Third International Joint Conference on Autonomous Agents and Mutli-Agent Systems, New York, USA, 506-513.*

Wurman, P., Walsh, W., and Wellman, M. (1998). Flexible double auctions for electronic commerce: Theory and implementation. *Decision Support Systems, 24, 17-27.*

Zeng, D. and Sycara, K. (1998). Bayesian learning in negotiation. *International Journal of Human-Computer Studies, 48, 125-141.*

Zlotkin, G. and Rosenschein, J. (1989). Negotiation and task sharing among autonomous agents in cooperative domains. *Proc. 11th International Joint Conference on Artifical Intelligence, Detroit, Michigan, 912-917.*