A GAME THEORETIC FRAMEWORK FOR COMMUNICATION DECISIONS IN MULTIAGENT SYSTEMS

by

TUMMALAPALLI SUDHAMSH REDDY

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2012

To my family

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Manfred Huber and my co-supervising professor Dr Gergely Zaruba for their invaluable support and advise. This dissertation would have not been possible without their significant effort. I would also like to thank Mr. David Levine for the help and support he has given me for the past few years, first as the thesis advisor for my Masters of Science degree, and secondly as a committee member for my dissertation. I would also like to thank Dr. Farhad Kamanger for his helpful comments and for serving on my PhD committee.

I would like to thank my family for their constant support and encouragement. This work would have not been possible without their support.

November 16, 2012

ABSTRACT

A GAME THEORETIC FRAMEWORK FOR COMMUNICATION DECISIONS IN MULTIAGENT SYSTEMS

TUMMALAPALLI SUDHAMSH REDDY, Ph.D.

The University of Texas at Arlington, 2012

Supervising Professor: Manfred Huber

Communication is the process of transferring information between multiple entities. We study the communication process between different entities that are modeled as multiple agents. Agents are assumed to be rational and take actions to increase their utility. To study the interactions between these entities we use game theory, which is a mathematical tool that is used to model the interactions and decision process of these multiple players/agents. In the presence of multiple players their interactions are generally modelled as a stochastic game.

In most cases it is clear that communication can help in coordinating the actions between multiple agents such that they can achieve higher utility, but, what is not clear is how the agents can take decisions about when to communicate and more importantly what to communicate. In this dissertation, we focus on the question of what information the agents can communicate and how they take decisions on selecting information to communicate.

Here we assume that the communication medium, protocols and language are already present in this multiagent system. We address the question of information selection in the communication process.

In this thesis, we develop a formal framework for communication between different agents using game theory. Our major contributions are:

A classifications of multiagent systems and what information to communicate in these various cases.

Algorithms for inverse reinforcement learning in multiagent systems, which allow an agent to get a better understanding about the other agents.

A mathematical framework using which the agents can make two important decisions, when to communicate, and, more importantly what to communicate in different classes of multiagent systems.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

Communication is a fundamental aspect of all sentient organisms that are capable of reasoning, be they humans or animals communicating verbally, or unicellular organisms such as bacteria communicating using signalling [1]. Communication can be defined as the process of conveying information between sentient beings. While the process of communication differs between different organisms, the question of what information to communicate is central to all organisms.

There are many forms of communication which differ based on the abilities of the organism or entity that is communicating. In human beings, communication is generally done consciously by either verbally using specific language that all people in the group can reasonably understand, or visually using specific sign language such as the American Sign Language system, or by subconsciously using certain body language or facial expressions. The more formally defined the language and communication process is, the better it is for communicating information and understanding the communicated information.

In most cases there is a reason to communicate any information. We believe that the true value of communication is based on that reason and the impact it has on the long term benefit of the communicating entity[1].

[1]In this document we use the terms agent, players, entity to mean a rational agent which acts in the environment to maximize its long term benefit

To study the interactions between these entities we model them as rational agents [2] in a multiagent system (MAS) [3] and then use *game theory* [3] to model the interactions between these agents.

A rational agent is defined as an agent that always takes an action which maximizes its performance measure.

Game theory makes certain assumptions about the agents in the multiagent system. Mainly, it assumes that all agents are strictly rational and the agents act in the environment in a way such that they maximize their own rewards. This assumption makes all MAS fundamentally non-cooperative or selfish in nature. But in certain cases where the agents have to cooperate to accomplish some task, this behaviour of cooperation is enforced by the reward function of each individual agent. This behaviour is also known as cooperative game theory.

1.1 Objective

Agents often need to communicate to accomplish their tasks. This is true in both cooperative and non-cooperative systems. While it is clear in most cases that communication is needed, it is not clear what piece of information needs to be communicated. This problem is compounded by the fact that communication is not always free. Most of the research on communication in MAS has been focused toward developing communication languages [4], robust protocols [5], and cross platform communication standards [6]. Recently there has been some interest in what information to communicate [7]. While one approach is to use an information theoretic framework, we believe a game/utility theoretic framework would be more appropriate to use when deciding which information to communicate.

If one were to take a top down approach to communication, one can break the communication process into multiple components:

1. *How to communicate* : this addresses the discovery of communication services, communication protocols etc.

2. *When to communicate* : at which time step of the execution should the agents communicate? Should they communicate their entire strategy for all the states in the world at the beginning of the game? Or should they decide to send information only when it is useful?

3. *What to communicate* : this refers to the information selection process. Since there is a cost to sending messages, the agents must try to minimize the cost of communication to increase their utility over time. The agents must be able to compute which pieces of information are the most useful to communicate.

4. *Understanding communication*: this is concerned with common language development. Agents must be able to understand the information the other agents are sending.

There have been various other works on the different aspects of communication in cooperative systems. Here we are attempting to answer the question of what information to communicate but in a much broader multiagent framework. We make no assumptions about cooperation between agents. If the agents are cooperative then it would be reflected in their reward functions. The framework we are proposing also address the issue of incomplete information about both the world and the other agents in the world. We also present inverse reinforcement learning algorithms that can be used to estimate the reward functions of other agents in cases where the information about the other agents' reward function is not available.

The framework presented here is different from most previous work on using game theory for communication [8] [9] which is mainly focused on using game theory to model the communication process such that the agents (the hearer) understand the information that is communicated by other agents (the speaker). In our research

we assume that when an agent conveys some information to another agent then the receiving agent understands the information perfectly.

1.2 Methodology

We study the communication between intelligent agents. We model the MAS using the stochastic game framework [10] from game theory. In stochastic games each agent has its own individual reward function that is used to describe the objective of the agent. The agents select their actions to maximize their utility based on their reward functions. The world is divided into states based on some characteristic of the world. For example, in the case of a simple grid world example the states are created based on (x,y) coordinates of the agent's locations.

The agents probabilistically transition from one state to another based on the joint actions of all the agents. After each transition the agents get an individual reward for the state. The objective of each agent is to maximize these rewards. We use techniques from reinforcement learning to learn the policy for each agent, where the policy describes the actions that the agent takes at each state of the game.

We classify the multiagent systems based on the observability of the state. If the agents have a perfect observation of the current state of the system then the multiagent system is known as a fully observable multiagent system. If the agents only have a partial observability of the current state, then the multiagent system is known as a partially observable multiagent system.

In the case of both fully observable decentralized MAS, and partially observable decentralized MAS, each agent must compute its own policy. It is often the case that the agents learn different policies, which leads to miscoordination between agents when they execute their learned policies. We use communication as a mechanism to avoid this problem of miscoordination. Communication is also useful in the case of

partially observable decentralized MAS, where the agents can also communicate their observations or belief about the current state of the world.

We model the communication process using another framework from game theory known as extensive form games, where each communication action has a reward associated with it. By modelling the communication as a game the agents can select their communication actions based on their numerical values such that they maximize their long term benefit by reducing the short term cost of communication.

1.3 Contribution

Our main contributions of this thesis in order of their presentation are:

- We classify the multiagent systems based on certain key characteristic of the agents and the environmental such as the abilities of the individual agents, the observability of the world and the information that is available about the other agent. We then analyse what information to communicate in these various cases.
- We present a minor enhancement to the popular NashQ learning algorithm, which addresses the question of equilibrium selection during learning.
- When the information about the reward function of the other agents is not available then how do we estimate this information based on other information about the agents? Algorithms for inverse reinforcement learning in MAS are presented to resolve this question.
- We present a mathematical framework based on which the agents can make two important decisions: when to communicate and, more importantly, what to communicate in different classes of MAS.

- We model the communication process between agents as a game using the extensive form game framework, which allows the agents to numerically evaluate the value of communicating any piece of information during the execution process.
- We propose a learning algorithm that uses the communication game during the learning process to improve on the previous solution which use the communication game only during execution.

1.4 Outline

The rest of the thesis is organized as follows:

Chapter 2 introduces some background material about the field of MAS, communication in MAS and some formal models from game theory that are used in the rest of the thesis. In Chapter 3, we review some literature that addresses the question of planning and communication in MAS. We also present some work on inverse reinforcement learning in single agent systems.

In Chapter 4, we look at communication in MAS in detail. We present different classifications of the MAS and the information that should be communicated in these different classes of MAS. In Chapter 5, we present our enhancement to the Nash-Q learning algorithm. In Chapter 6, we present our inverse reinforcement learning algorithm in MAS [11].

We describe our communication game in Chapter 7, and present an algorithm to use this game in the execution phase of the MAS [12]. In chapter 8 we allow using the communication game during the learning phase instead of the execution phase.

In Chapter 9 we extend our communication game to the case of partially observable MAS. We present our conclusions and future work in Chapter 10.

CHAPTER 2

MULTIAGENT SYSTEMS

2.1 Introduction

A multiagent system (MAS) is any system where multiple actors/agents are acting in a way such that an agent's actions impact the other agents. Each agent in a multiagent system takes actions to maximize its utility. The field of MAS is very diverse with interests and contributions from fields such as artificial intelligence, distributed systems, economics, operations research and many others. The main advantage of MAS is that they allow us to model more general systems with a higher degree of precision than single agent systems. A lot of research has been done in the field of MAS which addresses issues such as generating optimal policies [13] for the agents, dynamic team formation[14], communications [15] [6], negotiations[16], distributed problem solving[17], agent-oriented software design [18] [19], Multiagent learning [13] [12], multi-robot coordination[20] [21], etc. A MAS can be either cooperative or non-cooperative.

The main objective of the agents in MAS is to learn how to act in the environment. Learning in MAS is hard and is made more complex by the presence of multiple decision makers, which raises issues such as conflict of interest between agents, a lack of coordination between agents, and that the agents need to consider how their current interactions can impact future interactions with other agents. Both repeat games and stochastic games are useful to model the impact of the current actions on future interactions. In this thesis we model our MAS using a game theoretic framework known as stochastic games that is explained in this chapter.

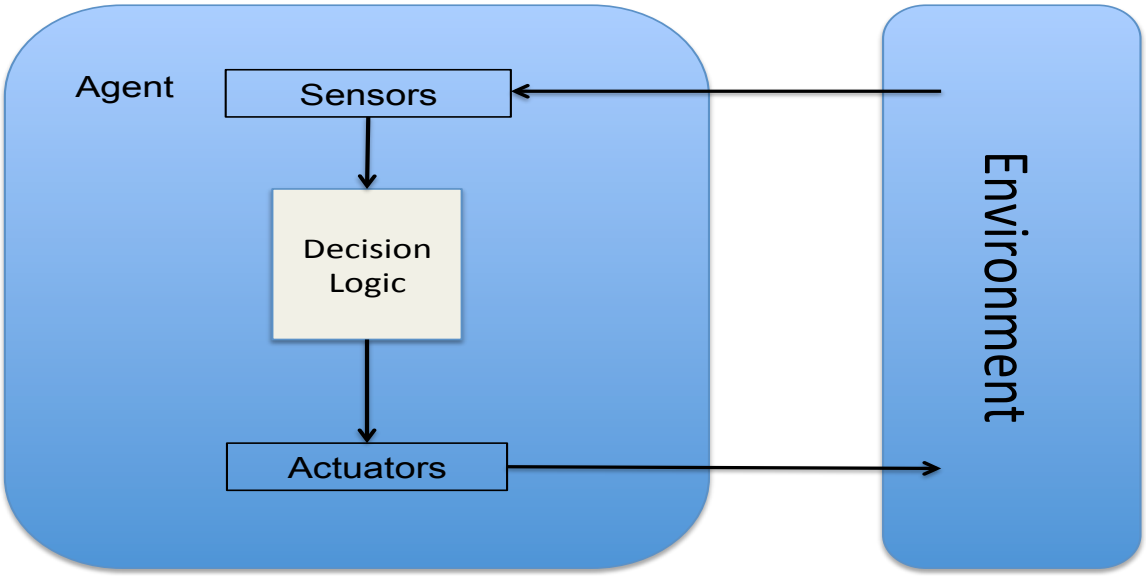

Figure 2.1. Agent interacts with environment.

In the stochastic game [10] framework, all aspects of the agents are modeled directly or indirectly. For example, the actions available to the agent are modeled directly, whereas the abilities of the agent to execute the actions perfectly is modeled as the probability of the agents transitioning from a state to the next state. The major focus of stochastic games is on the interactions between the agents.

In this chapter, we first look at single agent systems and two methods to model single agent systems. We then look at various methods to model multiagent system using game theory and briefly discuss communication in multiagent system which will be talked about in more detail in future chapters.

2.2 Single Agent System

According to Russell and Norvig (2010), *"An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators [2]."* This idea is illustrated in Figure 2.1 [2].

At each point in time, the environment can be in one of a finite number of states. Each state is represented by certain characteristics of the environment. Based on the above definition, the agent perceives these characteristics through its sensors and then takes an action through its actuators. These actions may cause the world to transition from one state to another. Each state also has a reward associated with it. The higher the numerical value of the reward the better it is for the agent. The goal of the agent is to take actions such that it maximizes its long term utility.

Most single agent systems are modeled as either a *Markov Decision Process (MDP)* or as a *Partially Observable Markov Decision Process (POMDP).*

2.2.1 Markov Decision Process

A MDP can be used to model a single agent system. While a MDP in itself is not of great importance to this work, some of the solutions that are used in regards to MDPs can be easily extended to the case of MAS. A MDP is defined as a tuple $\langle S, A, P, \gamma, R \rangle$, where

- S is a finite set of M states
- $A = \{a_1, ..., a_k\}$ is a set of k actions
- $P : S \times A \times S^{'} \rightarrow \mathbb{R}$ is the state transition probability function. $P(s'|s, a)$ denotes the probability of transitioning from state s to state s' by taking action a.
- $R : S \rightarrow \mathbb{R}$ is the reward function. $R(s)$ denotes the reward obtained by entering state s.
- $\gamma \in [0,1)$ is the discount factor.

A deterministic policy in a MDP is defined as a mapping $\pi : S \rightarrow A$, where $\pi(s) = a$ denotes that in state s, under policy π, we always execute action a. The

value function for a policy π is defined as the expected reward that the agent obtains by executing the policy starting from state s, discounted over time.

$$V^{\pi}(s) = E\left[R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + ... \mid \pi\right]$$

where the expectation is over the sequence of states $(s_1, s_2, ...)$ that are encountered, while starting from state $s_1 = s$ and executing the policy π.

The above equation is commonly represented using the Bellman equations which are represented as:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^{\pi}(s') \tag{2.1}$$

$$Q^{\pi}(s, a) = R(s) + \gamma \sum_{s'} P(s'|s, a) V^{\pi}(s') \tag{2.2}$$

Given a MDP, the agent tries to find an optimal policy (as is generally the case in traditional Reinforcement Learning). A MDP can have multiple policies; as such we use π^* to denote an optimal policy. The value function for the optimal policy is denoted as $V^{\pi^*}(s)$ and the optimal Q-function is denoted as $Q^{\pi^*}(s, a)$. The optimal policy should maximize the value for each state in the MDP. This is expressed in the Bellman optimality condition as:

$$\pi^*(s) \in \arg\max_a Q^{\pi^*}(s, a) \tag{2.3}$$

2.2.2 Partially Observable Markov Decision Process

A POMDP is a generalization of the MDP explained above. In a MDP the state of the world is fully observable whereas in a POMDP the agent only has partial observability regarding the state. Formally, a POMDP is defined as a tuple $\langle S, A, O, P, Z, \gamma, R \rangle$, where

- S is a finite set of M states
- $A = \{a_1, ..., a_k\}$ is a set of k actions

- O is a finite set of observations.
- $P : S \times A \times S^{'} \rightarrow \mathbb{R}$ is the state transition probability function. $P(s'|s,a)$ denotes the probability of transitioning from state s to state s' by taking action a.
- $Z : S \times A \times O \rightarrow \mathbb{R}$ is the observation probability function. $P(o|s,a)$ denotes the probability of observing an observation o while in state s and taking action a.
- $R : S \rightarrow \mathbb{R}$ is the reward function. $R(s)$ denotes the reward obtained by entering state s.
- $\gamma \in [0,1)$ is the discount factor.

Since, the actual state of the agent is not always available in a POMDP, the notion of belief state is used. A belief state b is a probability distribution over the set of actual states in the system. $b(s)$ denotes the probability assigned to state s by belief state b. The probability of the next state $s^{'}$ can be calculated from the current state probability by using the following equation

$$b(s^{'}) = \alpha Z(o|s^{'},a) \sum_{s} P(s^{'}|s,a)b(s) \tag{2.4}$$

where α is a normalizing constant that is calculated by the following equation,

$$\alpha = \frac{1}{\sum_{s' \in S} Z(o|s',a) \sum_{s \in S} P(s'|s,a)b(s)} \tag{2.5}$$

The optimal policy $\pi^{*}(b)$ is a mapping from belief states to actions. A POMDP is computationally more complex than a MDP.

The main difference between a MDP and POMDP is that in a POMDP the optimal action depends on the current belief state of the agent as compared to the actual state of the agent in a MDP.

2.3 Game Theory

Game theory is a mathematical framework for studying the interactions between independent and rational agents. Rationality in Game theory means that the agents act in the environment by balancing the costs against the rewards such that they maximize their personal rewards or utilities. This game theoretic framework lends itself to the field of MAS as it can be used to model the world and the agents' behaviour very easily. We first introduce normal form games which are the simplest form of games that can be used to model multiple agents. The normal form games are used to represent single shot games where the agents only take actions for one time step and then the game ends. We then introduce the concept of Nash equilibrium in normal form games, before we look into more complex games such as extensive form games (EFG) and stochastic games which can be used to model the agents interacting over multiple time steps.

2.3.1 Normal Form Game

A finite, n-player Normal Form Game can be defined as tuple G = (N, A, u), where,

- N is the finite number of players in the game
- A = A_1 x A_2 x ...x A_n, is the finite set of actions available to the agents where, A_i is the finite set of actions available to player i. A vector $(a_1, ..., a_n) \in A$ is called an action profile and represents a joint action defining the individual action taken by each of the agents.
- u = $(u_1, ..., u_n)$, is the joint utility function of the agents where, $u_i : A_i \rightarrow \mathbb{R}$ is a function that maps an action profile to a real valued utility for a player i.

These normal form games can be easily represented in the form of matrices where the rows represent actions of one agent and the columns represent actions of

		Player 2	
		Action 1	Action 2
Player 1	Action 1	4,4	0,3
	Action 2	3,0	2,2

Figure 2.2. A simple bimatrix game.

the other agents. When there are only two agents or players then the games are known as bimatrix games. In the above game the rewards or utility for both agents following a joint action is given by the values in each block; for example agent 1 gets reward 4, and agent 2 gets reward 4, when agent 1 takes action 1 and agent 2 takes action 1. We look at some basic concepts such as strategies, mixed strategies, expected utilities of a mixed strategy and best responses before we describe the concept of Nash equilibrium.

In the game shown in Figure 2.2, we talk about the actions that are available to both agents, but not their set of strategies.

In game theory, a *strategy* [3] is generally defined as the action that an agent always takes. If all the agents follow a pure strategy then we call this vector of pure strategies a pure strategy profile

In the case of a *mixed strategy* [3], a strategy $st_i(a_i)$ is known as a mixed strategy for agent i if the agent picks its actions based on a probability distribution over a set of available actions. By st_i we denote the probability that an action a_i will be played under the mixed strategy st_i. We use ST_i to denote a set of mixed strategies for agent i.

The *support* [3] of a mixed strategy st_i for an agent i is the set of pure strategies a_i , such that, $st_i(a_i) > 0$. It can also be said that the mixed strategy is a probabilistic distribution over a set of pure strategies for the agent.

		Player 2	
		Action 1	Action 2
Player 1	Action 1	1,0	-1,-1
	Action 2	2,3	5,6

Figure 2.3. A simple bimatrix game with 2 players and 2 actions.

A pure strategy can be considered a special case of a mixed strategy where the support is a single action. A majority of the interesting games have mixed strategies. The expected utility u_i for player i of a mixed strategy profile $st = (st_1, ..., st_n)$ is defined as

$$u_i = \sum_{a \in A} u_i(a) \prod_{j=1}^{n} st_j(a_j)$$

We now look at the concepts of best response. We define $st_{-i} = (st_1, ..., st_{i-1}, st_{i+1}, ...st_n)$, where st_{-i} is a strategy profile containing the strategies of all agents other then agent i. Player i's best response to a strategy st_{-i} is a mixed strategy st_i^* such that, $u_i(st_i^*, st_{-i}) \geq u_i(st_i, st_{-i})$, for all strategies $st_i \in ST_i$.

Based on the above definition of a best response we can now define the Nash equilibrium as a strategy profile $st = (st_1, ..., st_n)$, where for all agents i, st_i is the best response to st_{-i}. The Nash equilibrium is considered to be strict or weak depending on whether the strategy profile st contains a unique best response for all the agents or not. In a vast majority of domains, a strict Nash equilibrium is seen very rarely. A weak Nash equilibrium is the more commonly seen solution in MAS.

In the example shown in Figure 2.3, we can see that the action profile, where both agents take action 2, represented as (2,2), is a Nash equilibrium and both the agents play that Nash equilibrium. Now let us look at a slightly different example.

In the example shown in Figure 2.4, there are two Nash equilibria, the action profile (1,1) and the action profile (2,2). In this case how do the agents select a

		Player 2	
		Action 1	Action 2
Player 1	Action 1	4,4	-1,-1
	Action 2	2,3	4,4

Figure 2.4. A simple bimatrix game with 2 players and 2 actions.

consistent Nash equilibrium? This problem is compounded by the fact that both agents must play at the same time. If one agent plays after the other, then the agent going second will simply tailor his response to match the response of the first agent to ensure they both get high reward. But in the case where both agents must take actions at the same time, we will not be able to guarantee that the resulting action profile would be a Nash equilibrium. While there are other equilibria that can be used to resolve this issue such, as a correlated equilibrium or the more recently proposed compromise equilibrium, these require slightly different assumptions. We will look at how to resolve this issue by using communication.

2.3.2 Extensive Form Games

Extensive form [3] is an alternate representation of games that makes an explicit assumption about the sequence in which agents take actions. There are two types of extensive form games, the special case of a perfect information extensive form game and the more general case of imperfect information extensive form games. Here we only look at the perfect information extensive form game as in a fully observable world, it is generally the case that we have perfect information about the world.

Extensive form games model games using a tree-like structure where each node represents the choice of an agent, the edges represent the possible actions taken, and the leaves represent the payoffs defined by the utility function of each agent.

The finite perfect information extensive form game can be defined as a tuple $G = \langle N, A, H, Z, \chi, \rho, \sigma, u \rangle$ [3]

- N is a finite set of n players.
- A is a set of actions.
- H is a set of nonterminal choice nodes.
- Z is a set of terminal nodes from H.
- $\chi : H \rightarrow 2^A$ is the action function which assigns to each nonterminal choice node a set of actions.
- $\rho : H \rightarrow N$ is the player function which assigns to each nonterminal node a player n $\in$ N, who chooses an action at that node.
- $\sigma : H \times A \rightarrow H \cup Z$ is the transition function which maps a choice node and an action to a new choice or terminal node such that for all $h1, h2 \in H$ and $a1, a2 \in A$, if $\sigma(h1, a1) = \sigma(h2, a2)$ then $h1 = h2$ and $a1 = a2$.
- $u = (u_1, ..., u_n)$, where $u_i : Z \rightarrow \mathbb{R}$ is a real-valued utility function for player i in the terminal nodes Z.

In the EFG, G, described above, the pure strategies of player i are defined as the Cartesian product $\Pi_{h \in H, \rho(h)=i} \chi(h)$.

Theorem 2.1: *[3] Every (finite) perfect-information game in extensive form has a pure-strategy Nash equilibrium.*

Given an EFG G,the subgame of G rooted at node h is the restriction of G to h and the descendants of h. The set of subgames of G consists of all of the subgames of G rooted at some node in G [3].

The subgame-perfect equilibria (SPE) [3] of a game G are all strategy profiles st such that for any subgame G' of G, the restriction of st to G' is a Nash equilibrium of G'

2.3.3 Stochastic Games

A stochastic game is a dynamic game with one or more players, and with probabilistic transitions between states. It was introduced in 1953, by Lloyd Shapley [Shapley, 1953]. We now look at the General Sum Stochastic Game(GSSG), which is used to model a fully observable world, and the Partially Observable Stochastic Game (POSG) that is used to model a world with incomplete information about the state of the world.

2.3.3.1 General Sum Stochastic Games

A General-sum discounted stochastic game is defined as a tuple $\langle S, N, A, P, R, \gamma \rangle$

- N is a finite set of n players.
- S is a finite set of m states
- A_i is a finite set of agent i's actions, and $(a_1, ..., a_n)^T \in \times_i A_i$ is a vector of the joint action of the agents.
- P: $S \times A \times S \to \mathbb{R}$ is the state transition probability function. $P(s'|s, \vec{a})$ denotes the probability of transitioning from state s to state s' taking joint action $\vec{a}$.
- R_i: S $\to \mathbb{R}$ is the reward function for agent i. $R_i(s)$ denotes the reward of agent i for entering state s.
- $\gamma \in$ [0,1) is the discount factor.

In this definition we assume that S and A_i are finite. A policy is defined as $\pi : S \times A \to \mathbb{R}$, where A is the joint action set of all agents.

It should be noted that General Sum Stochastic Games (GSSG) generalize both the Markov decision process and repeated games. In the above definition, if we reduce the number of agents in the problem to only one agent, then we get a standard future discounted MDP. The future discounted reward is the more commonly used method to aggregate the utility of an agent. Alternately, we can use the average reward case. In our research we use the future discounted reward model, as it is generally more intuitive. In a GSSG, each state is a game, and can be modeled as a normal form game (NFG). Solving a NFG means finding its Nash equilibrium. Unlike a NFG mentioned in the previous sections, in the case of GSSGs the agent must solve for a Nash equilibrium based on the utilities and not the immediate rewards for the state. The utilities are the sum of discounted future rewards. It has been proven that in a GSSG, a Nash equilibrium always exists.

Planning in GSSGs involves computing the policy for each agent, where the policy is defined as the probability over the actions that an agent can take in a given state. In GSSGs the agents can make decisions based on either the current state or the history of state transitions. Two concepts of interest here are those of Markov strategy [3] and Markov perfect equilibrium [3]. A Markov strategy is one, where the agent selects its action only based on the current state and the reward that the agent can get in the next state; in other words, the history of state transitions has no influence on how the agent selects its current action. This limits the amount of memory that is needed for planning. A Markov perfect equilibrium is defined as a strategy profile that contains only Markov strategies and is a Nash equilibrium in every state irrespective of the starting state. A Markov Perfect equilibrium is a refinement of a Nash equilibrium.

In GSSGs we will refer to a strategy profile as a policy, π, which is defined as $\pi : S \rightarrow P(\vec{A})$, where $\vec{A}$ is the joint action set of all agents. We use the notation

π_i and π_{-i} to denote the strategy of agent i and the strategy of all agents excluding agent i, respectively. The utility function and the corresponding Q function for an agent i in the stochastic game can be expressed as,

$$V_i^{\pi}(s) = \sum_{\vec{a}} \left(\prod_i P_{\pi_i}(a_i|s) \right) \big(R_i(s) + \gamma \sum_{s'} P(s'|s,\vec{a}) V_i^{\pi}(s') \big) \quad \forall \vec{a} \in A \ , \ a_i \in A_i \ , \ s \in S$$

$$Q_i^{\pi}(s,\vec{a}) = R_i(s) + \gamma \sum_{s'} P(s'|s,\vec{a}) V_i^{\pi}(s') \tag{2.6}$$

$$Q_i^{\pi}(s,\pi^*) = \sum_{\vec{a}} Q_i^{\pi}(s,\vec{a}) \prod_i P_{\pi_i^*}(a_i|s) \tag{2.7}$$

In Equation 2.6 we describe the Q-function of agent i for a state and action pair $(s,\vec{a})$, while in Eq 2.7 we describe the Q-function of agent i for a state s and policy π. The policy $\pi(s)$ represents the probabilities of all the joint actions in state s. In the case of pure strategies, $P(a|s) = 1$ since the support only has one action a in a state s.

We now mention two important results from Game Theory that explain how agents select their actions in a GSSG.

Theorem 2.2: *Every n-player, general-sum, discounted-reward stochastic game has a Markov perfect equilibrium [3].*

This is an important result as this shows that in the case of stochastic games in Multiagent systems, there always exists a policy which is also a Nash equilibrium in all states.

Theorem 2.3: *Given a General Sum Stochastic Game,* $M = \langle N, S, A, P, R, \gamma \rangle$*, and a policy* $\pi : S \times A \to \mathbb{R}$*,* π^* *is an optimal policy for agent* i *if and only if, for all* $s \in S$*,*

$$Q_i(s, \pi^*) \geq Q_i(s, a_i \pi^*_{-i}) \qquad \forall a_i \in A_i, \forall i \in N \tag{2.8}$$

There are many different methods to solve the problem of finding a Nash equilibrium in NFG, one of the most commonly used is the Lemke-Howson method. In a GSSG, the agents must be able to learn the optimal policy for all the agents. There are many methods to solve this problem, one of which is the Nash-Q learning algorithm. We will explain the Nash-Q learning process in the next chapter.

2.3.3.2 Partially Observable Stochastic Games (POSG)

POSGs are a very powerful way to model multiagent systems. They can handle different types of uncertainties that GSSGs cannot, such as partial observations, noisy observations, random events, and uncertain outcomes [22]. The ability of the POSG framework to handle these different scenarios makes it a very powerful way to model MAS. While POSG is a good method to model these different kinds of situations, it also has a very big drawback; mainly that solving any POSG becomes intractable very fast. There have been some methods which give approximate solutions for POSGs.

There are multiple definitions of POSGs that differ slightly from each other. We use the following definition: A POSG is defined as a tuple $\langle S, N, A, Z, P, O, R, \gamma \rangle$

- N is a finite set of n players.
- S is a finite set of m states.
- A_i is a set of actions of agent i, and $(a_1, ..., a_n)^T \in \times_i A_i$ is a vector of the joint action of the agents.

- Z_i is the finite set of observations for agent i, and $(z_1, ..., z_n)^T \in \times_i z_i$ is a vector of the joint observations of the agents.
- P: $S \times \vec{A} \times S \rightarrow \mathbb{R}$ is the state transition probability function. $P(s'|s, \vec{a})$ denotes the probability of transitioning from state s to state s' taking joint action $\vec{a}$.
- O: $S \times \vec{A} \times \vec{Z} \rightarrow \mathbb{R}$ is the observation probability function. $O(\vec{z}|s, \vec{a})$ denotes the probability of observing $\vec{z}$ by taking joint action $\vec{a}$ in state s resulting in state s'.
- R_i: S $\rightarrow \mathbb{R}$ is the reward function for agent i. $R_i(s)$ denotes the reward of agent i for entering state s.
- $\gamma \in$ [0,1) is the discount factor.

A POSG can have a finite or infinite sequence of stages. The number of stages are known as the horizon of the game. One way to solve finite horizon POSGs is to convert it into an EFG and then compute the Nash equilibrium [22]. In an EFG the POSG is converted into an acyclic directed graph where the node also contains the history of the game. We consider infinite-horizon POSGs only. A single agent POSG is a POMDP.

2.4 Communication in MAS

Communication is a very critical part of a MAS, as it allows agents to coordinate their actions to accomplish their goal. The issue of communication languages or communication systems has been well researched and implementations of different standards of communication protocols are easily available [23]. For the purpose of this research we assume that the communication protocols and methods are always available to the agent, although the agent will incur some cost for using the communication systems.

The main issue we address in this research is what information to communicate. To be able to make this decision the agent must be able to decide what information, if communicated to the other agents, might increase its own utility. As such, it is important for the agent to understand the world in which the agents are acting. We assume that all the agents in the world have a good understanding of the world, such that if we were to provide all the information about one agent to the other agent and if the other agent has the same actions available to it, then the other agent can successfully perform the task of the first agent.

Once an agent has enough understanding about the world and the other agents in the world, the next question the agents must consider is if there is any benefit of communicating with the other agents. We will look into this in more detail in later chapters. Once both these questions have been answered, the agent must decide what information to communicate before the agent can communicate with other agents.

CHAPTER 3

RELATED WORK

3.1 Introduction

In the previous chapter we looked at some frameworks to model the single agent and multiagent systems (MAS). In this chapter we first look at some methods to solving the problem of generating the policies for the agents in these frameworks. We also look at some of the work done on communication in MAS and describe Inverse Reinforcement Learning (IRL) for single agent systems.

Since we only consider the case of stochastic games, we do not discuss learning in repeat games. Repeat games are similar to stochastic games in the sense that the game is played over time. But, they differ from stochastic games in the sense that it is generally the same single stage game that is played over and over again. Learning in repeat games is often much easier than in stochastic games and algorithms such as fictitious play [3] and rational learning [3] are used to compute the strategy for the agents.

3.2 Reinforcement Learning in Multiagent Systems

Multiagent learning has been heavily influenced by the field of reinforcement learning, mainly because reinforcement learning does not require complete information about the model of the environment. The agents can learn by taking actions and observing the reward obtained at each step. One of the most famous reinforcement learning method that has been widely used in single agent systems is the Q-learning method [24]. Traditionally, Q-learning is applied to learn optimal policies in a Markov

decision process. In a multiagent setting Q-learning can be extended to allow the agents to learn policies which are optimal in the sense of an equilibrium solution concept.

There have been multiple multiagent Q-learning algorithms that have been proposed, the most popular of which is the Nash Q-learning method. In the Nash Q-learning method [25], the authors, propose to find a Nash equilibrium in each time step and update the Q-values of the agents based on the next state resulting from taking an action that is part of the Nash equilibrium. The Correlated Q-learning [26] method is similar to the Nash Q-learning method, but, instead of the Nash equilibrium, the authors use a correlated equilibrium to update the Q-values of the agents.

Some other Q-learning methods that have been proposed over the years are friend-or-foe-Q learning [27] and minimax-Q learning [28]. Minimax Q-learning is a fusion of Q-learning and a linear programming solution to the zero sum game. In friend-or-foe Q-learning each agent maintains a single Q-table for itself and the agents must know beforehand whether the other agents are friends or foes. If they are friends then each agent looks for a coordinated equilibrium and if the other agents are foes, then each agent looks for an adversarial equilibrium. Neither of these two solutions listed above solve for the more general case where agents are neither completely cooperative or completely adversarial. Of all the Q-learning based methods that have been proposed so far, only the NashQ algorithm works for the case of games that have more then two agents.

For a more extensive review of reinforcement learning in MAS, please look at [29].

3.2.1 NASH Q-Learning

The Nash-Q algorithm is an extension of Q-learning to a non-cooperative multiagent context. The multiagent system is modeled as a general sum stochastic game. Each agent maintains Q-functions over joint actions and performs updates based on assuming Nash equilibrium behaviour over the current Q-values.

In [25] the authors use the Lemke-Howson algorithm that returns either a single Nash equilibrium or multiple equilibria in a fixed order. The Lemke-Howson algorithm used is only applicable for two player games and cannot be extended to multiagent systems with more then two agents. There are other methods to calculate Nash equilibria in general sum games.

A major drawback of the Nash Q-learning method is that if multiple Nash equilibria exist, the agents must in a decentralized way pick a Nash equilibrium to update their Q-values. This is only possible if there is either a fixed ordering of Nash equilibria, which will break the decentralized property, or if each game has a global optimal point or saddle point which is also a Nash equilibrium and this point is used to update the Q-values of the agents.

As mentioned above the authors used the Lemke-Howson algorithm to get the Nash Equilibria in a fixed order. Based on this order they proposed 3 ways for equilibrium selection. The proposed first Nash and second Nash are simply selecting the first and second Nash equilibrium that the Lemke-Howson algorithm returns. The third method they proposed is the best Nash, or simply selecting the Nash equilibrium which gives an individual agents its highest payoff value (irrespective of the value it returns for the other agents).

The first two methods are not practical for any multiagent system with more then two agents as Lemke-Howson does not work for more than two agents and also because enforcing a fixed order or a fixed choice of algorithm for all agents is breaking

the decentralized property of the multiagent system. Using the third method is also often not feasible as in the absence of a global optimal point or saddle point for all the agents, the agents will play different equilibria that will result in non-equilibrium actions.

3.3 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) [30] can be defined as a method to determine an agent's reward function, if an agent's policy and a model of the environment are given. Formally, if the environment and the agent's behaviour are modeled as a Markov Decision Process (MDP), then the objective of IRL is to find the reward function of the MDP and thus to identify the goal or objective of the agent. IRL is of particular interest in applications such as human and animal learning, where the reward function is either not known or can't be easily expressed.

IRL has been successfully used in applications such as apprenticeship learning, where an agent observes an expert's behavior over time and then estimates a reward function which can be used to generate a policy for the observer agent to perform the task of the expert with either the same efficiency (or in some cases with improved efficiency) as the expert [31] [32].

We now look at IRL in single agent Markovian environments. Most of the algorithms proposed to date address this case. We briefly summarize the work done by Ng and Russell [2000]. In this problem we are given a MDP\R, which is an MDP model without the reward function. A comparison of traditional reinforcement learning and inverse reinforcement learning is shown in Figure 3.1

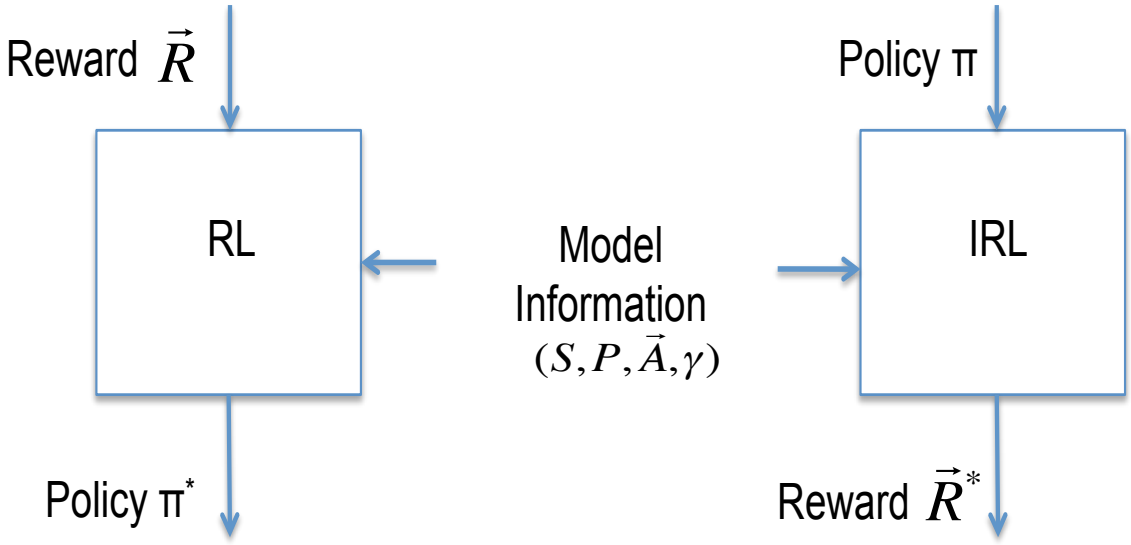

Figure 3.1. IRL in Single agent systems.

3.3.1 IRL for MDP\R from Policies

The IRL problem in a single agent, completely observable Markovian environment can be stated as: Given an MDP\R $\langle S, A, T, \gamma \rangle$ and a policy π, find a reward function that makes π an optimal policy for the MDP. A necessary and sufficient condition for the reward function R of the MDP to make π optimal is,

$$Q^{\pi}(s, \pi(s)) \geq Q^{\pi}(s, a), \forall s, \forall a \tag{3.1}$$

which states that deviating from the optimal policy does not result in higher value. The condition can be expressed in terms of reward as [30],

$$(P_{\pi} - P_a)(I - \gamma P_{\pi})^{-1} R_{\pi(s)} \succeq 0 \tag{3.2}$$

where P_a denotes the $|S| \times |S|$ matrix where the element (s, s') is the transition probability $P(s'|s, a)$ and $R_{\pi(s)}$ denotes the $|S|$ vector with the s-th element being the reward $R(s, \pi(s))$. The $\succeq$ operator is a row wise greater then or equal operator.

Given the optimal policy π, the reward function can be found by solving the following linear program [30];

maximize

$$\sum_{i=1}^{M} min_{a \in a_2,...,a_k}(P_\pi(i) - P_a(i))(I - \gamma P_\pi)^{-1}R - \lambda \parallel R \parallel_1$$

subject to:

$(P_\pi - P_a)(I - \gamma P_\pi)^{-1}R \succeq 0$

$|R_i| \leq R_{max}, i = 1, ..., M$

where λ is an adjustable penalty coefficient for having too many non-zero values in the reward function.

3.3.2 IRL for MDP\R from Observations

When the optimal policy π^* is not given, but some trajectories of the optimal policy are given, the IRL problem can be solved in an iterative form starting with a random policy π_1. The true reward function R should satisfy the condition $V^{\pi^*}(s) \geq V^{\pi_1}(s)$, since π^* is the optimal policy with respect to R. To make this problem more tractable, Ng and Russell [30] assume that instead of expressing the reward function as having a single value per state, the reward function can be represented in a more compact format by assuming a linear expression

$$R(s) = \alpha_1\phi_1(s) + \alpha_2\phi_2(s) + \cdots + \alpha_d\phi_d(s) \tag{3.3}$$

where $\phi_1, \phi_2, ..., \phi_d$ are fixed known bounded basis functions mapping from S to $\mathbb{R}$, and α_i are the unknown parameters that need to be estimated. The goal here is to reduce the complexity of the number of parameters in the reward function from $|S|$ to the number of basis functions. For a candidate reward function $\hat{R} = \hat{\alpha}^T\phi$, the

value of the optimal policy is estimated by the average empirical return from the K trajectories.

$$\hat{V}^{\pi^*}(s_0) = \frac{1}{K} \sum_{k=1}^{K} \sum_{t=0}^{T-1} \gamma^t \hat{R}(s_t^k)$$

where $(s_0^k, s_1^k, ..., s_{t-1}^k)$ are the T-step state sequences in the k-th trajectory of the optimal policy. The value of other policies with different candidate reward function $\hat{R}$ can be found either using the sample trajectories or using the Bellman equations. This algorithm computes a new policy π_i^* based on the reward value obtained from the reward function found in the current iteration i.

The algorithm iteratively tries to find a better Reward function, by solving the following optimization for l-iterations:

maximize

$$\sum_{i=1}^{l} p(\hat{V}^{\pi^*}(s_0)) - \hat{V}^{\pi_i^*}(s_0))$$

subject to:

$$|\alpha_i| \leq 1, i = 1, ..., d$$

where, p(x) = x if x $\geq$ 0, and p(x) = 2x if x $<$ 0. This is to ensure that any violations of the constraints are penalized. The algorithm stops when it encounters a $\hat{R}$ which minimizes the difference between the value of the optimal policy and a policy generated by the estimated $\hat{R}$ to below an ϵ-threshold.

3.4 Communication in MAS

This research is focused on making communication decisions in multiagent systems, mainly on deciding when to communicate and what information to communicate. It has been proven that presence of a communication mechanism does not reduce the computational complexity of cooperative multiagent systems [33] unless it is free and unlimited. It has been established that in the presence of free and

unlimited communication the cooperative multiagent system modeled as DEC-MDPs or DEC-POMDPs can be solved similar to single agent MDP or POMDP models [7]. In the case of non-cooperative systems modelled as GSSGs or POSGs even in the presence of free and unlimited communication the problem complexity does not reduce.

In practice, however, communication is not always free and the agents incur a cost every time they decide to communicate [33] [34] [35]. The issue of when to communication has been studied in the context of cooperative systems in [36] [37] [33] [34] [35]. In [38] [39] [40], the agents must decide on when they want to communicate and then all agents must communicate their information so as to unify their world view.

Another aspect of communication that needs to be considered is whether a communication action replaces a domain-level action in the timestep in which it takes place (OR-communication) [41], or can communication and domain-level actions take place in the same timestep(AND-communication) [41]. While most of the approaches mentioned earlier assume synchronous communication, in [37], the authors have considered the case where there is a delay in communication.

In their paper [7], Roth et al. have looked at the problem of what to communicate in a partially observable cooperative multiagent system. In their system all agents share a common team reward and individual agents are not allowed to be selfish. In their work they use a heuristic based approach to calculate the value of information that they believe will help the team get higher reward. Beynier and Mouaddib [42] propose an inference graph based method to model the communication between agents in a Dec-MDP (a cooperative MAS). There have been various other works on the different aspects of communication in cooperative systems.

Most of the work mentioned above considers the case of cooperative MAS. Burkov and Chaib-draa [43] presented an algorithm that allows for decentralized planning in stochastic games with communication. They modelled communication at each time step, as a normal form game, where the Nash Equilibrium on the communication gives a unique Nash equilibrium of the stochastic game. Similarly, Murray and Gordon [44] have presented an approach to decentralized planning where the agents communicate with each other to find a subgame perfect equilibrium in a stochastic game. In both these approaches, the authors do not consider any cost for communication between the agents. They model the MAS as a stochastic game with *cheap talk*. In practice, however, there is often a cost to communicate information, and the agents must balance these costs with their individual rewards to effectively plan their communication process.

In [45], an algorithm has been proposed to include communication during execution of the POSG. In this approach the author has proposed to cluster their observation histories and then communicate the index of the cluster. Using this approach the agents minimize the length of their messages.

The work presented by Burkov and Chaib-draa [43] and Roth et al [7] are the closest to the work presented here in this dissertation.

CHAPTER 4

COMMUNICATION IN MULTIAGENT SYSTEMS

In multiagent systems, agents operate in some environment where they may or may not be able to accomplish their goals by themselves. The agents might have to depend on other agents to accomplish their goals (in both cooperative and non-cooperative environments). In these cases agents might need to communicate with each other to accomplish their task. Communication is also an important component of the negotiation process between agents, in dynamic team forming, or in most other tasks. Some of the messages exchanged during negotiation between agents are: one of the agents proposes a course of action, retracts the proposed action, the agents accepts the proposed action, rejects the proposed action, or proposes a different course of action

The focus of our current research is to answer the question of what information to send and to some degree, when to send the information. These two issues are dependent on each other and hence cannot be easily decoupled.

4.1 Introduction

In this chapter we look at different types of Multiagent systems and define three ways in which we can differentiate between them. For ease of understanding we introduce a simple two player gridworld-based example and see how the different classifications impact the communication between the two agents.

The game shown in Figure 4.1 has 2 players in the locations shown by A1 and A2. The agents' goal locations are shown by G1 and G2, and the obstacles or pits in

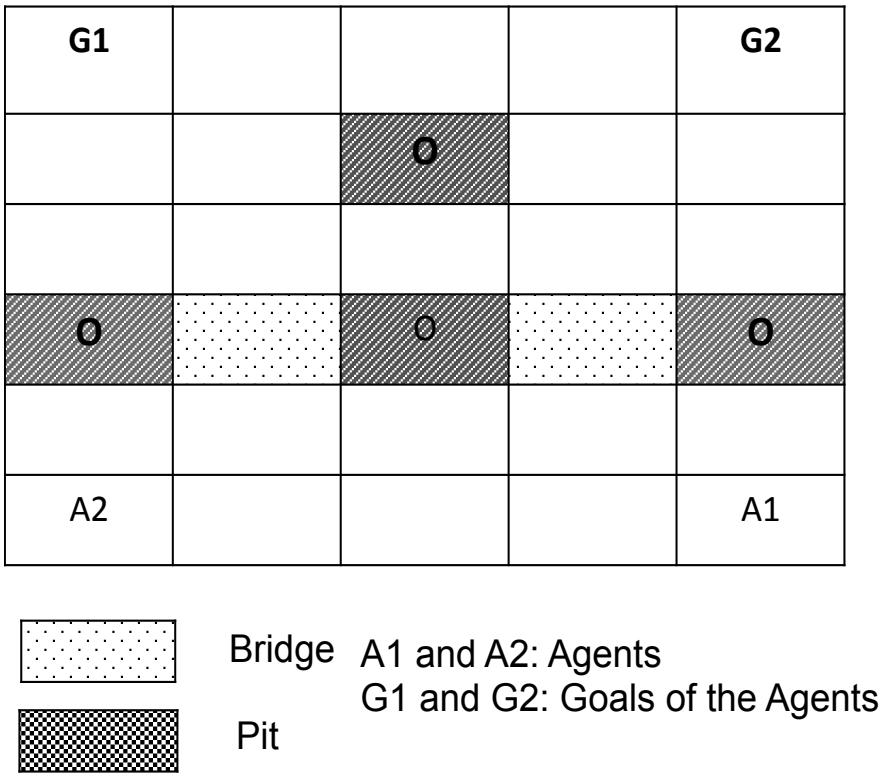

Figure 4.1. Gridworld game.

the world are shown by O. The areas marked as bridges can be crossed by the agents with some positive reward. The reward structure of the agents is as follows:

- A1 reaches G1: 100
- A2 reaches G2: 200
- A1 and A2 are in the same grid location: -20
- A1 or A2 enters the pit or obstacle location: -100
- A1 or A2 enters the bridge location alone: -50
- A1 and A2 enter the bridge location together: 50

The reward values were chosen arbitrarily and can be changed. The gridworld is divided into different states, where the state is defined as $\langle (x_1, y_1), (x_2, y_2) \rangle$. $(x_1.y_1)$ are used to denote the (x,y) coordinate of A1 and (x_2, y_2) are used to denote the

(x,y) coordinate of A2 in the grid. The agents, taking any one of the four movement actions going north, south, east and west transition probabilistically from one state to another based on a transition function.

This is a simple game and can be easily expanded to include more agents, obstacles and goal locations. The objective of the agents is to maximize their future discounted reward-based utility while minimizing the costs. The game ends when either of the agents enters their goal location.

We now look into the different classes of multiagent systems.

4.2 Classification of Multiagent Systems

We classify the multiagent systems based on the characteristic of the world, the abilities of the agents, and the knowledge the agents have about each other. This is shown in the Figure 4.2. The field is MAS is primarily divided into the Fully Observable MAS and Partially Observable MAS as indicated in the figure. The next classification is based on the knowledge the agent has regarding the reward function of the other agents. The final classification is based on the action set of the agents.

4.2.1 Observability

As shown in Figure 4.3, a simple and accurate way to classify multiagent systems based on the world is to classify them based on the accuracy of the observation regarding the world. In the case where the agents can view the state of the world accurately, the world is a fully observable world. A fully observable multiagent system can be modeled easily using the General Sum Stochastic Game (GSSG) discussed in Chapter 2. We cannot use the normal form game framework as most of these multiagent systems have multiple time steps with each time step being modeled as a different Normal Form Game(NFG). While we can use the extensive form game

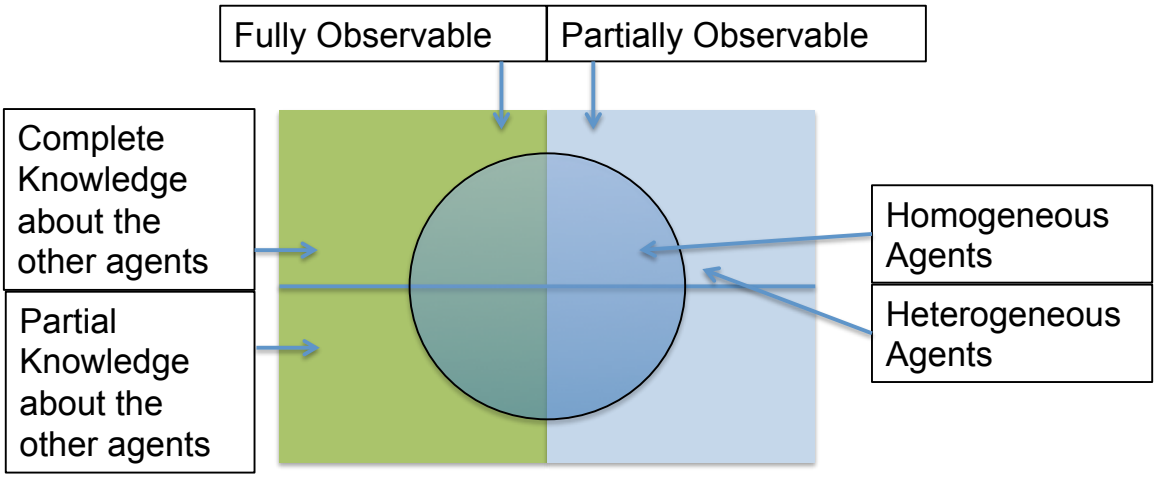

Figure 4.2. Classification of the multiagent systems.

framework to model these multiagent systems, we run into the problem of these games being of infinite horizon. Also, since we need to model the extensive form games as a game tree before the game is played, the amount of memory required to store the entire game can be very high. Figure 4.1 shows a grid world example of a fully observable multiagent system. In this example each agent has a perfect understanding of the current state of the world, which means each agent is aware of the location of all agents in the world. The agent also has a perfect knowledge of the other agents actions, and the transition probability of the world.

In the case where the agents cannot view the state of the world accurately, the world is a partially observable world where agents can see only a small part of the world state and must make a probabilistic assumption about the current state. This partially observable multiagent system can be modeled using the Partially Observable Stochastic Game discussed in Chapter 2. While small partially observable multiagent systems can be modeled using the extensive form game framework, we run into the

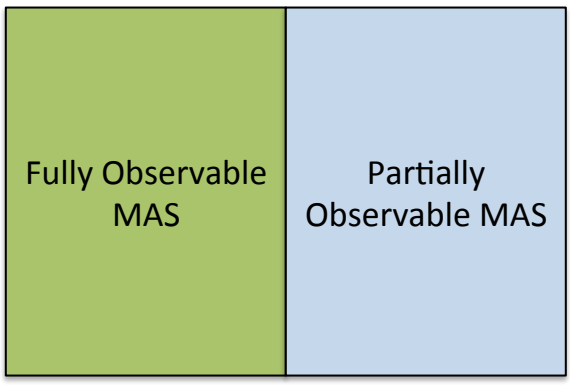

Figure 4.3. Classification of the multiagent systems based on observability of the world.

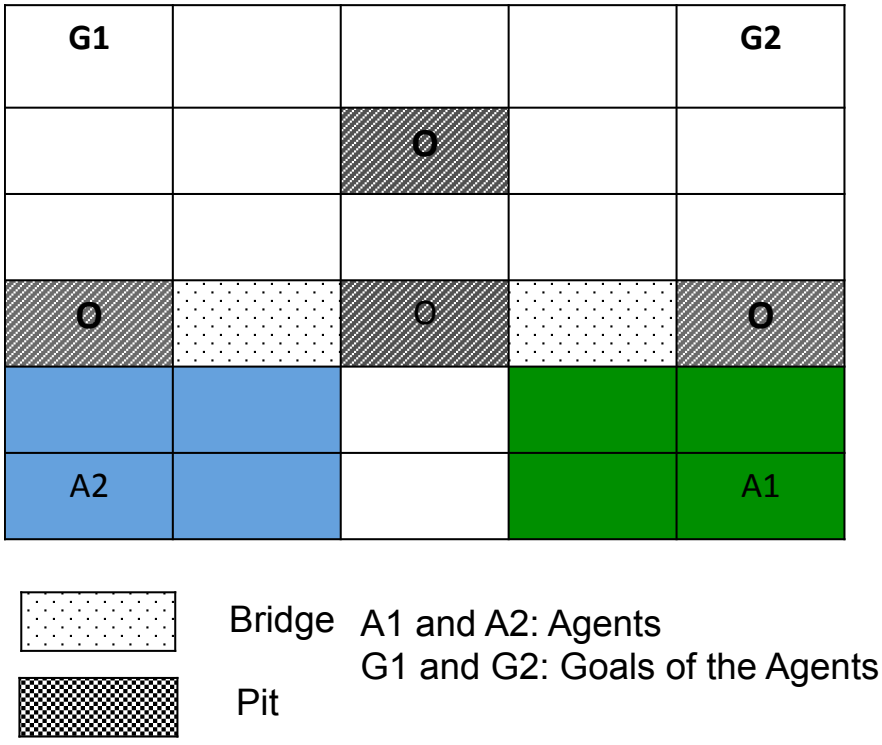

Figure 4.4. Partially observable gridworld.

problem of high memory requirement for large games. The GSSG and NFG framework cannot model imperfect information about the world.

Lets look at the grid world example shown in Figure 4.4. In this example, each agent can only observe its immediate surroundings, i.e. the agents can view the status of the blocks surrounding their current position. Based on the example shown in Figure 4.4, agent A1 can only view the area marked in green and agent A2 can only view the area marked in blue. The agents don't know the location of other agents or the actions other agents take, unless they can observe the other agent.

4.2.2 Information About Other Agents

In traditional modelling of multiagent systems it is always assumed that we have complete information about the other agents in the world. We know what actions are available to the other agents, what the other agents' reward function is, and if we do not know what the other agents' reward functions are then we can observe the agents' actions and their reward during the execution of the learning process.

In a fully observable multiagent systems with incomplete information about the other agents, planning becomes very difficult and the situation degrades into a partially observable stochastic game which, as mentioned before, becomes intractable very fast.

We present an inverse reinforcement learning algorithm to estimate the reward function of the other agents based on either their policy information or by observing their trajectories. The inverse reinforcement learning algorithm (IRL) is presented in Chapter 6. Based on our estimation of the reward function we can still use the General Sum Stochastic Game framework to solve the fully observable world case presented earlier, which is generally far more tractable than the Partially Observable Stochastic Game framework(POSG). In the case of the partially observable world we

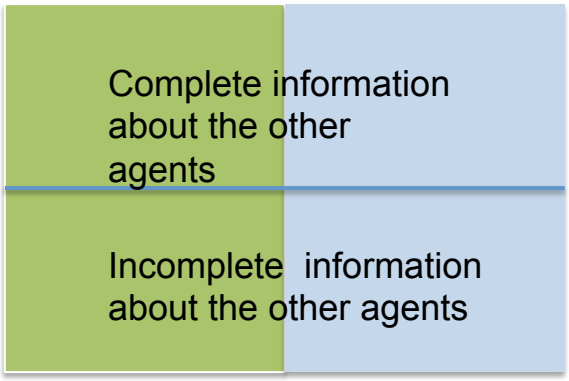

Figure 4.5. Classification of the multiagent systems based on information about other agents.

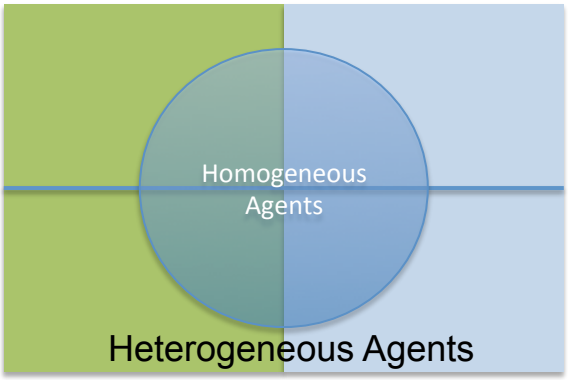

Figure 4.6. Classification of the multiagent systems based on similarities between agents.

assume that the information is known. This is due to the limitation of not having an IRL algorithm for POSGs that can be used by the agents in a decentralized manner.

This classification is shown in Figure 4.5.

4.2.3 Agent Type

When some agents have similar capabilities in terms of their action sets and similar reward functions, then the agents are said to be homogeneous agents. When the agents have different capabilities or different reward functions, the agents are said to be heterogeneous agents. This classification is of interest in symmetric games

where either agent can execute not only their part of the equilibrium but also the other agents' strategies.

This classification is shown in Figure 4.6. In symmetric games the agents will need to communicate which agent executes which part of the equilibrium.

4.3 Modeling of Multiagent Systems and What to Communicate

In a cooperative multiagent system, where the agents share a common reward, we have the following cases:

In a fully observable world modeled as a MMDP or a Dec-MDP, the agents can communicate joint actions. The issue of incomplete information about the other agents does not occur as all agents share the same reward. The agents can have similar capabilities which does gives rise to symmetric games but that is handled by communicating the joint actions with a specification of which agent executes which part of the joint action. In a fully observable world the agents have a perfect information about the state of the world and other agents, therefore communicating any information other then the joint actions to take is not meaningful.

In a partially observable world modeled as Dec-POMDP, the agents can communicate observations, belief state and joint actions. The issue of incomplete information about the other agents does not occur as all agents share the same reward. The agents can have similar capabilities which does gives rise to symmetric games but that is handled by communicating the joint actions with a specification of which agent executes which part of the joint action. In a partially observable world, the agents do not have a perfect information about the current state of the world, therefore, the agents can communicate the observations that they get from the environment and also their belief about the current state of the world. Once the agents have a common under-

Fully Observable MAS	Full Knowledge of other agents	Partial Knowledge of other agents
Modeled as General Sum Stochastic Game (GSSG)	GSSG	GSSG\R

Partially Observable MAS	Full Knowledge of other agents	Partial Knowledge of other agents
Modeled as Partially Observable Stochastic Game (POSG)	POSG	POSG\R

Figure 4.7. Models of the Different Classes of Non-Cooperative MAS.

standing of the world, the agents can then communicate their joint actions as in the case of the fully observable world.

In a Non-Cooperative multiagent system, where the agents are selfish and try to maximize their personal reward, we have the following cases, shown in Figure 4.7.

A fully observable world with complete information about other agents is modeled as a GSSG. This is the case of the example shown in Figure 4.1. The agents have complete information about the current state of the world and the other agents' reward functions. In this case the agents only need to communicate which Nash equilibrium (NE) to play. In each state the agents need to make a communication decision. They must first decide whether to communicate with the other agents based on the difference in utility of the equilibrium they would play if they did not communicate and the highest pay-off equilibrium they could play if they were to communicate.

A fully observable world with incomplete information about other agents is modeled as a GSSG\R (this is a GSSG with a missing reward function). In this case, all the agents can communicate their reward function information to the other agents so that they can then plan efficiently. Once this information is available, this

becomes a GSSG and the agents can communicate which NE to play. When the reward function is missing, the agents can also choose to use an inverse reinforcement learning algorithm based on either the policies or the trajectories of the policy of the other agents to compute the reward function and convert the GSSG\R into a GSSG. The agents can then communicate which NE they would play similar to the case of the a fully observable world with complete information.

A partially observable world with complete information about other agents is modeled as a POSG. This is the case shown in Figure 4.4. In this case, similar to the previous case of a fully observable MAS the agents can communicates which NE to play. Since the agents do not have a perfect understanding of the current state of the world, the agents can communicate their observations or belief state. This becomes a bit tricky, as the agents must decide if there is any value in communicating their observations. The agents might also choose to communicate their belief state rather then their observations. The agents use the belief state to compute the equilibrium and can base their decisions of whether to communicate the observations or belief state based on the difference in the value of the equilibrium they could achieve before communicating their observations and the equilibrium they can achieve with communication.

A Partially Observable world with incomplete information about other agents is modeled as POSG\R. The agents communicate the same information as in the case of a partially observable MAS. But since the agents dont know about the reward function of the other agents, they might want to communicate the reward function information as well. A decentralized IRL algorithm for POSGs does not exist yet.

In a fully observable MAS or a partially observable MAS, irrespective of the completeness of the information, if the agents are homogeneous then in the case of symmetric games the agents can communicate the Nash equilibrium including infor-

mation about which agent plays a particular part of the Nash equilibrium. If the agents are heterogeneous then the agent should still communicate the Nash equilibrium but if the action sets are completely different then they do not need to specify which part of the equilibrium is played by a particular agent.

Multiple approaches to the problem of what to communicate in the case of cooperative systems have been previously proposed. We will mainly focus on non-cooperative system, but our solutions can be used for cooperative systems as well.

CHAPTER 5

EQUILIBRIUM SELECTION IN NASHQ

We have introduced the NashQ learning algorithm [25] in Chapter 3. The NashQ algorithm has been shown to converge under a very limited set of conditions. In this chapter we discuss some advances we have made to the algorithm to increase the likelihood of convergence when these conditions are not fulfilled.

5.1 Introduction

In Chapter 3 we mention some shortcomings of the NashQ algorithm, mainly that it suffers from the problem of not picking a Nash equilibrium consistently over multiple iterations. Also, the three approaches that have been proposed by Hu and Wellman, to select amongst Nash Equilibria are not widely applicable. We present a more stable approach to selecting an equilibrium in our version of NashQ based on risk dominance. We also extend our approach to the case of Nash-SARSA [46]

5.2 Payoff and Risk Dominance

In their work on the theory of equilibrium selection, Harsanyi and Selten(1998) [47] introduced the concepts of payoff dominance and risk dominance as two different selection criteria for selecting a Nash equilibrium when multiple Nash equilibria are present. They then argued that the payoff dominance should be the first criterion to be applied. Harsanyi(1995) [48] later retracted this argument and proposed to take risk dominance as the relevant selection criterion.

	left	right
left	A,a	C,b
right	B,c	D,d

Figure 5.1. Bimatrix game format.

	left	right
left	4,4	0,3
right	3,0	2,2

Figure 5.2. A simple bimatrix game.

Both payoff dominance and risk dominance are refinements of a Nash equilibrium, which are used mainly in equilibrium selection. A Nash equilibrium is considered payoff dominant if it is Pareto superior with respect to all other Nash equilibria. When multiple equilibria are present then all players would agree on the payoff dominant equilibrium since it offers to each player at least as much payoff as the other Nash equilibria. The problem with payoff dominant equilibrium solution is that it might not always exist.

A Nash equilibrium is considered risk dominant compared to another if its product of deviation losses is larger than for the other Nash equilibrium. Deviation loss is simply defined as an agent's loss in value should it choose to deviate from the action selected by the Nash equilibrium. The risk factor is simply the product of the deviation losses for all agents involved in the Nash equilibrium. In the bimatrix game structure shown in Figure 5.1 there are two equilibria, (right,right) and (left,left). The deviation loss of the equilibria can be expressed as,

$$D_{rr} = (D - C)(d - c) \tag{5.1}$$

$$D_{ll} = (A - B)(a - b) \tag{5.2}$$

In Figure 5.2, we present a chicken game between 2 players. In this game there are two pure Nash equilibria, namely (left,left) and (right,right). We can see that (right,right) is a pure strategy profile that payoff dominates all other strategies. The strategy (left,left) risk dominates the strategy (right,right), as the product of the deviation losses of the row agent and the column agent from the strategy (left,left) is: $(2-0) \times (2-0) = 4$, whereas, the deviation loss of the strategy (right,right) is: $(4-3) \times (4-3) = 1$.

The simple intuition behind the concept of preferring the risk dominant equilibrium is that, since the cost of deviating from the risk dominant equilibrium is higher, no agent would choose to deviate from this equilibrium and hence this is the most stable equilibrium available to the agents. In the cases where there is uncertainty about the other agents payoffs, selecting the risk dominant Nash equilibrium is a better choice as it provides the highest margin of error in the payoffs.

In the absence of some form of coordination between agents, the risk dominant strategy will be selected to play. Recent results from evolutionary game theory [49] [50] have shown that in large games often the agents fail to play the payoff dominant equilibrium and instead play the risk dominant equilibrium, even if it was payoff dominated by other equilibria.

5.3 Risk Dominance Method for Equilibrium Selection

As mentioned in the previous section, in the absence of coordination between agents, the risk dominant equilibrium is one that the agents will play with the belief that the other agents will also select the same equilibrium. Here we define the deviation loss in a general N-player m-action game. From Equation 2.7 in Chapter 2, we can define the quality of a state s and the optimal policy, π^*, for an agent i as:

$$Q_i^{\pi}(s, \pi^*) = \sum_{\vec{a}} Q_i^{\pi}(s, \vec{a}) \prod_i P_{\pi_i^*}(a_i|s)$$

where $\pi^*(s)$ is a probability distribution over the support of the Nash equilibrium for state s. In the case of more than two actions we calculate the deviation loss by selecting the action that is not in the support of the Nash equilibrium for agent i and gives the highest value in the state.

$$Q_i^{\pi}(s, \pi^*_{-i}, a_i) = \sum_{a_{-i}} P_{\pi^*_{-i}}(a_{-i}|s) Q_i^{\pi}(s, a_{-i}a_i) \forall a_i \in A_i, \forall a_{-i} \in \pi^*_{-i}$$

$$Devloss_i = \min_{a_i \in A_i/SA_i} (Q_i^{\pi}(s, \pi^*) - Q_i^{\pi}(s, \pi^*_{-i}a_i)) \tag{5.3}$$

Here SA_i are the actions in the support of the Nash Equilibrium in state s, and A_i/SA_i is the set of actions for agent i not in the support. The risk factor, R_f, of an equilibrium is the product of the deviation losses of all agents in the game.

$$R_f(\pi^*) = \prod_{i \in N} Devloss_i \tag{5.4}$$

The algorithm for the process is described below.

Algorithm 5.1: Accept a List of NEs, Payoff Matrix, Num of Players N

---

numNEs = len(NEs)
for $k = 0$ to $numNEs$ **do**
  for $i = 0$ to N **do**
    Compute $Devloss_i(k)$ using Equation 5.3
    Compute $R_f(k)$ using Equation 5.4.
    Compute safety value $Q(s, \pi_{k_i} a_{-i})$ for all a_{-i}
  end for

end for

$j = \{argmax_k(R_f(\pi_{k_i}))\}$

if $len(j) > 1$ **then**

  $SafetyNE = \{argmax_{k \in j} Q(s, \pi_{k_i} a_{-i})\}$ for agent i

  if $len(SafetyNE) > 1$ **then**

    for $l = 0$ to $len(SafetyNE)$ **do**

      Compute $Q_i^{\pi}(s, \pi^*)$ of $SafetyNE[l]$

    end for

    $MaxUtilityNE = \left\{argmax_{k \in SNE} Q_i^{\pi_{k_i}}(s, \pi^*)\right\}$

    if $len(MaxUtilityNE) > 1$ **then**

      for $l = 0$ to $len(MaxUtilityNE)$ **do**

        Compute $Q_{-i}^{\pi}(s, \pi^*)$ of $MaxUtilityNE[l]$

      end for

      $MaxUtilOtherAgentsNE = \left\{argmax_{k \in MaxUtilityNE} Q_{-i}^{\pi}(s, \pi^*)\right\}$

      $SelectNE = MaxUtilOtherAgentsNE$

    else

      $SelectNE = MaxUtilityNE$

    end if

  else

    $SelectNE = SafetyNE$

  end if

else

  $SelectNE = j$

end if

return $SelectNE$

The risk dominant equilibrium is the one with the highest R_f. These equations only hold for Markov strategy Nash equilibria, which are behavioural strategies where the history does not impact the action selection and the Nash equilibrium can be executed in a decentralized way. The Equilibrium selection process used here is described in Algorithm 1.

We use multiple constraints to select the equilibrium. The first step is to reduce the set of equilibria based on the risk dominance described above, after which the safety value of the equilibria for the agent is calculated and only the ones with the highest safety value are allowed. The $safety\ value$, $Q_i(s, \pi_i a_{-i})$, is the minimum value that agent i will get if it plays the risk dominant equilibrium and the other players diverge from it. Once these two constraints are applied, the agent then selects from these equilibria by reducing the set further by selecting the equilibria that have the highest utility for itself. The resulting set is further reduced by selecting the equilibria which give the highest value to the other agents as well. The agents can reduce the set until only one equilibrium is left or until the set cannot be reduced based on the above constraints.

The agents must execute the above algorithm in each state during the NashQ learning process. The smaller the number of equilibria in the set after the first criterion, the better the likelihood of convergence would be.

5.4 Results

We applied our algorithm to a Gridworld based game similar to the one mentioned in Chapter 4. We briefly describe our experiment and the results here.

We tested our algorithms on a simple grid world problem similar to the one used by Hu in the NashQ implementations and present our preliminary results here. The grid world problem and its NashQ solution is shown in Figure 5.4.

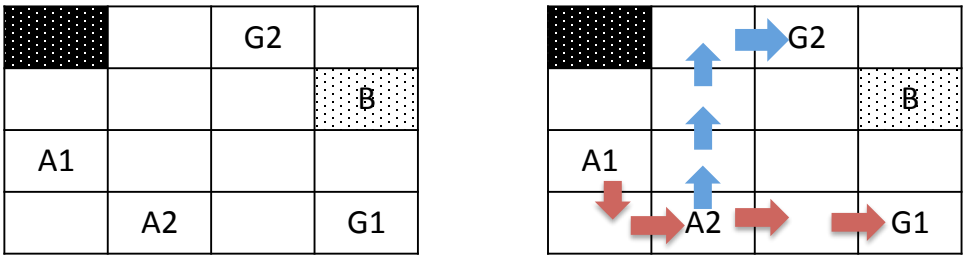

Figure 5.3. Left:The Gridworld game; Right: Agent policies.

The grid world contains two agents, A1 and A2. The agents must reach their goal states marked G1 and G2. We add a bonus state at cell location (3,4). When both agents reach this state at the same time they get a reward. After that there are no rewards to be gained at the bonus cell. The state is modeled as $((X,Y)_{A1},(X,Y)_{A2},BSR)$, where the first two (x,y) coordinates are the locations of Agent 1 and Agent 2, respectively. BSR is the state of the bonus cell; it is initially set to 0 to indicate that the state has never been visited. BSR is set to 1 if both agents arrive there for the first time, and BSR is set to 2 if both of the agents visit the state more then once at the same time. We also added a strictly negative reward cell (4,1), such that the agents will get a negative reward if they enter that cell location. The reward structure of the game is as follows:

If an agent tries to enter the bonus state alone it gets a reward of -200.

The agents gain a reward of 100 when they reach their goal cell locations.

The agents get a reward of 50 if they enter the bonus cell together for the first time.

The agents get a reward of -10 if they enter any cell together.

The agents get a reward of -200 if they the strictly negative reward cell.

The agents get a reward of -10000 if they try to exit the world.

The result of the NashQ learning algorithm using the risk dominance method for selecting between equilibria is shown on the right in Figure 5.4, where the brown arrows indicate the policy of Agent 1 and blue arrows indicate the policy of Agent 2. In the grid game the bonus state is ignored by both agents because the risk factor of entering the state along with the other agent is too low in the initial states since the loss for any agent that deviates from this strategy is very small compared to the equilibrium where the agents don't attempt to enter the state. In different games we have seen that the agents converge to the same policy in most cases, but we have seen some cases where the agents come up with different policies.

CHAPTER 6

INVERSE REINFORCEMENT LEARNING FOR DECENTRALIZED NON-COOPERATIVE MULTIAGENT SYSTEMS

6.1 Introduction

The objective of inverse reinforcement learning is to learn an agent's reward function based on either the agent's policies or the observations of the policy. In this chapter we address the issue of using inverse reinforcement learning to learn the reward function in a multiagent setting where the agents can either cooperate or be strictly non-cooperative.

The case of cooperative agents is a subcase of the non-cooperative setting where the agents collectively try to maximize a common reward function instead of maximizing their individual reward functions. Here we present two IRL algorithms where the first algorithm deals with the case where the policies of the agents are known. The second algorithm deals with the more realistic case where the agent policies are not known, but some trajectories of the policy are known.

We use the framework that was described by Ng and Russell [2001] and extend it for a multiagent setting. We show that in the case of known policies we can reduce the complexity of IRL for an agent in a multiagent systems such that each agent's reward function can be estimated independent of the reward function of the other agents. We assume that the agents are rational and follow an optimal policy. These assumptions are very common in multiagent systems.

In a single agent system, IRL can be defined as a method to determine an agent's reward function if its behavioural rules and a model of the environment are

given. Formally, if the environment and the agent's behaviour are modeled as an MDP, then the objective of IRL is to find the reward function of the MDP. IRL is of particular interest in applications such as human and animal learning, where the reward function is either not known or cannot be easily expressed. IRL has been successfully used in applications such as apprenticeship learning, where an agent observes an experts behaviour over time and then estimate a reward function which can be used to generate a policy for the observer agent to perform the task of the expert with either the same efficiency or in some cases with improved efficiency as the expert [31] [32].

While an MDP is a good starting point for modeling the environment, there are other models such as Multiagent Markov Decision Process (MMDP), General Sum Stochastic Games (GSSG), Partially Observable Markov Decision Process (POMDP) and Partially Observable Stochastic Games (POSG), which are more realistic. The MDP is a very simple case where the agent can observe the world perfectly, and when there is only one agent in the world. There are limits to the applicability of MDPs as it assumes perfect understanding of the world. In most real life situations there is always a certain amount of uncertainty about the world state. Also, there are many cases where more than one agent acts in the world and an MDP does not take into account the effect one of the agents has on the actions of the other agents. The case of partial observability can be handled by POMDPs for single agent systems and POSGs for multiple agent systems. MMDPs, GSSGs and POSGs are used in the case of more than one agent. IRL algorithms for POMDPs have been presented in [**?**]. Most of the work on IRL has been focused on the single agent MDP and recently on the single agent POMDP models. There has been barely any work on IRL in multiagent systems.

One attempt at adapting IRL for multiagent systems has been done in [51]. In their work the authors make some very important assumptions, which we believe are not completely realistic. Some of their assumptions are that while the agents are non-cooperative, there exists a centralized mediator which controls the policies of individual agents so as to maximize their joint reward function. Thus the system can be translated into an equivalent cooperative system. Their work also makes the more subtle assumption that there exists a way for agents to communicate. While this type of approach is acceptable in some cases, there exist a vast number of scenarios in which the above assumptions are simply not reasonable.

In this chapter, we consider scenarios where the agents are non-cooperative in nature and the decision process is decentralized such that each agent can execute their policy to maximize their reward while acting in a multiagent environment. This kind of scenario is quite common in various fields such as Robocup soccer, serious games, economics and other fields, where agents are competing with each other to maximize their internal reward. Specifically, we assume that the environment is modeled as a GSSG and that the agents follow an optimal markov perfect policy. The algorithms presented here are motivated from the classical IRL algorithms suggested by Ng and Russell (2000). We provide a general framework for dealing with multiagent systems. We expect that many of the advances in single agent IRL can be extended to multiagent systems as well.

We model our multiagent systems using the General Sum Stochastic Game framework presented in Chapter 2. Formally, the problem can be defined as: GSSG$\backslash\vec{R}$, given $\langle S, N, A, P, \gamma\rangle$ and an optimal Markov perfect policy π^*, find $\vec{R}$. As in the case of the single agent system (mentioned in Chapter 2), either the optimal policies of the agents can be given explicitly or only sampled trajectories of the optimal policies can be used.

6.2 IRL in GSSG\R from Policies

As mentioned in Chapter 2, the overwhelming focus in stochastic games is on Markov perfect equilibria (MPE). A MPE, π ensures that the policy is a Nash equilibrium in each and every state. Therefore, the strategy $\pi_i(s)$ is a best response to strategy profile $\pi_{-i}(s)$ in every state s. The policy π generated from the reward function for an agent i must satisfy the following condition:

$$Q_i(s, \pi_i \pi_{-i}(s)) \geq Q_i(s, a_i \pi_{-i}(s)), \forall a_i \in A_i \tag{6.1}$$

Theorem 6.1: *Markov perfect equilibrium condition: Let a finite state space S, a set of agents N, with each agent i having an action set* $A_i = (a_{i_1}, ..., a_{i_m})$*, transition probability matrices* $P_{a_i, a_{-i}}$ *and a discount factor* $\gamma \in [0, 1)$ *be given. Then the policy* π^* *given by* $\pi^*(s) \equiv \pi_i^* \pi_{-i}^*(s)$ *is a Markov perfect equilibrium if and only if,* $\forall a_i \in A_i$*, the reward* R_i $\forall i \in N$ *satisfies*

$$(P_{\pi_i^* \pi_{-i}^*} - P_{a_i, \pi_{-i}^*})(I - \gamma P_{\pi_i^* \pi_{-i}^*})^{-1} R_i \succeq 0 \tag{6.2}$$

Proof. From Section 2.3.3.1 we know that,

$$\begin{aligned}
V_i^{\pi^*}(s) &= \sum_a \Big(P_{\pi^*}(a|s)\big(R_i(s) + \gamma \sum_{s'} P(s'|s,a)V_i^{\pi^*}(s')\big)\Big) \\
&= \sum_a P_{\pi^*}(a|s)R_i(s) \\
&\qquad + \gamma \sum_a P_{\pi^*}(a|s) \sum_{s'} P(s'|s,a)V_i^{\pi^*}(s') \\
&= R_i(s) + \gamma \sum_{s'} \sum_a P_{\pi^*}(a|s)P(s'|s,a)V_i^{\pi^*}(s') \\
&= R_i(s) + \gamma \sum_{s'} V_i^{\pi^*}(s') \sum_a P_{\pi^*}(a|s)P(s'|s,a) \\
&= R_i(s) + \gamma \sum_{s'} V_i^{\pi^*}(s')P(s'|s,\pi^*(s))
\end{aligned}$$

The above equation can be written in vector form as,

$$V_i^{\pi^*} = R_i + \gamma P_{\pi^*} V_i^{\pi^*}$$

$$V_i^{\pi^*} = (I - \gamma P_{\pi^*})^{-1} R_i \tag{6.3}$$

The equilibrium Q-value, $Q_i(s, \pi')$, can be expressed as,

$$Q_i(s,\pi') = R_i(s) + \gamma P_{\pi'}(s) V_i^{\pi'}$$

$$Q_i^{\pi'} = R_i + \gamma P_{\pi'} V_i^{\pi'} \tag{6.4}$$

Substituting Equation 6.4 and Equation 6.3, into Equation 6.1, we can see that π^* is optimal, if and only if,

$$Q_i(s,\pi^*) \geq Q_i(s, a_i\pi^*_{-i})$$

$$\big(R_i + \gamma P_{\pi^*}(I - \gamma P_{\pi^*})^{-1} R_i\big) \succeq$$

$$\big(R_i + \gamma P_{a_i\pi^*_{-i}}(I - \gamma P_{\pi^*})^{-1} R_i\big)$$

$$\gamma P_{\pi^*}(I - \gamma P_{\pi^*})^{-1} R_i \succeq \gamma P_{a_i\pi^*_{-i}}(I - \gamma P_{\pi^*})^{-1} R_i$$

$$(P_{\pi_i^*\pi^*_{-i}} - P_{a_i,\pi^*_{-i}})(I - \gamma P_{\pi_i^*\pi^*_{-i}})^{-1} R_i \succeq 0 \tag{6.5}$$

$$\text{Maximize}_{\text{size of } S} \sum_{j=1} \sum_{a_i} \{(P_{\pi_i^* \pi_{-i}^*}(j) - P_{a_i \pi_{-i}^*}(j))(I - \gamma P_{\pi_i^* \pi_{-i}^*}(j))^{-1} R_i\} - \lambda ||R_i||_1$$

$$\text{Subject to:}$$

$$\left(P_{\pi_1^* \pi_{-i}^*} - P_{a_i \pi_{-i}^*}\right)\left(I - \gamma P_{\pi_1^* \pi_{-i}^*}\right)^{-1} R_i \succcurlyeq 0, \forall a_i \in A_i$$

$$|R_{ij}| \leq R_{max}\ j = 1 \ldots \text{size of S.}\ i = 1 \ldots N$$

Figure 6.1. Linear Program for IRL.

The above result shows that a_i is a best response to π^*_{-i}, and thus is in the support of π^* for agent i. When the above condition holds for all agents i, then the joint action is a Nash equilibrium. This completes the proof. □

For a GSSG, this result characterizes the set of all reward functions that are solutions to the IRL problem. As can be seen from Equation 6.5, the reward function R_i of an agent i does not depend on any other agent's reward. This helps us decouple the IRL problem of one agent from the other agents. We can get the reward function R_i of an agent i by solving the linear program shown in Figure 6.1.

We use the λ term as a penalty coefficient for having too many non-zero values in the reward function. λ regulates the trade-off between the stability and the sparsity of the rewards. This aspect of the linear program is similar to the case of IRL in single agent systems mentioned in Chapter 3.

6.3 IRL in GSSR\R from Trajectories

In the case where the optimal policies $\pi_i^* \pi_{-i}^*$ of the agents are not explicitly given, but some trajectories of the optimal policy are available, the IRL problem can be solved in an iterative form. Here we start with a random policy π. The true reward

function R_i should satisfy the condition $V_i^{\pi^*}(s) \geq V_i^{\pi}(s)$, since π^* is the optimal policy with respect to R_i for all agents i. To make this problem more tractable, we assume that the reward function can be linearly expressed as

$$R_i(s) = \alpha_{i_1}\phi_1(s) + \alpha_{i_2}\phi_2(s) + \cdots + \alpha_{i_d}\phi_d(s) \tag{6.6}$$

where $\phi_1, \phi_2, ..., \phi_d$ are fixed known bounded basis functions mapping from S to $\mathbb{R}$, and $\alpha_{i_j}s$ are the unknown parameters that need to be estimated. The goal here is to reduce the complexity of the number of parameters in the reward function from $|S|$ to the number of basis functions. For a candidate reward function $\hat{R}_i = \hat{\alpha_i}^T\phi$, the value of the optimal policy is estimated by the average empirical return, from the K trajectories.

$$\hat{V}_i^{\pi^*}(s_0) = \frac{1}{K}\sum_{k=1}^{K}\sum_{t=0}^{T-1}\gamma^t\hat{R}_i(s_t^k)$$

where $(s_0^k, s_1^k, ..., s_{t-1}^k)$ are the T-step state sequences in the k-th trajectory of the optimal policy. The value of other policies with different candidate reward function $\hat{R}_i$ can be found either using the sample trajectories or using the Bellman equations. The complete algorithm is described in Figure 6.2.

In the algorithm presented in Figure 6.2, we start by selecting α_{i_j} randomly for all agents i and all basis functions j. We then generate a Markov perfect Nash equilibrium policy based on these reward functions and add it to a set of policies. Then, until we reach a maximum number of iterations, and for each agent i, we calculate the new policy such that if we have the trajectory of the optimal policy in a state s then we use that information, and when we don't have the optimal policy information, we use information from the previously generated policy. We then solve the linear program as shown in the algorithm and compute a new policy based on the reward values that we obtain. If the difference between the value functions of

Input GSSG\R <S, n, $A_{1...n}$, P, γ>, basis functions ϕ, M trajectories

1. Choose α_{ij} randomly and set $\Pi = \emptyset$.
2. Compute a Nash equilibrium π using R=$\alpha^T \phi$
3. Set $\Pi = \Pi + \pi$
4. **For** l = 1 to MaxIter **do**
    a. **For** each agent i= 1 to n **do**
        i. $\pi_i^{\Delta} = \begin{cases} \pi_i^*(s), \forall s \in S_T \\ \pi_{l_i}(s), \forall s \notin S_T \end{cases}$
        ii. Find $\hat{\alpha}$ by solving the linear program

$$Maximize_{\hat{\alpha}} \sum_{\pi \in \Pi} \sum_{s \in S} p(V_i^{\pi^*}(s) - V_i^{\pi_{l_i} \pi_{-i}^{\Delta}}(s)) - \lambda ||\hat{\alpha}_i^T \phi_i||_1$$

Subject to:

$$|\alpha_j| \leq 1, \text{j =1 ....d}$$

    b. **end for**
    c. Compute π_{l+1} for the GSSG using $\hat{R} = \hat{\alpha}_l^T \vec{\phi}$
    d. **if** $|V^{\pi^*}(s) - V^{\pi_{l+1}}(s)| \leq \varepsilon, \forall s \in S$ **then**
        i. **return** $\hat{R} = \hat{\alpha}_l^T \vec{\phi}$
    e. **else**
        i. $\Pi = \Pi \cup \{\pi_{l+1}\}$
    f. **end if**
5. **end for**
6. **return** $\hat{R}$

Figure 6.2. IRL from trajectories.

the optimal policy and the newly generated policy is less then an ϵ threshold for all states, then we are satisfied with the reward function we have obtained.

6.4 Experimental Results

We have experimented on some small gridworld problems, such as the sample gridworld shown in Figure 6.3. In this section we present our results. We first generate a Markov perfect equilibrium policy then use the IRL algorithm to get the reward information from this policy information.

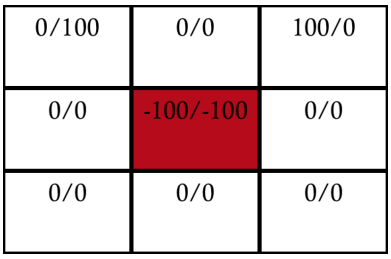

Figure 6.3. Grid World with rewards for Agent 1 and Agent 2.

In the gridworld shown in Figure 6.3. the first number shown in each cell is the reward that agent 1 gets when he enters the grid location, and the second number denotes the reward that agent 2 gets when he enters the grid location. The state variables in this world are the grid locations of both agents, such as (1,1),(1,3). The red square in the middle denotes the state where the agents both get a reward of -100 when they enter the location. The reward functions for both agents are described below.

Agent 1 reaches its goal position of (3,3) = 100.
Agent 2 reaches its goal position of (3,3) = 100.
Agent 1 or Agent 2 reaches the position (2,2) alone = -100.
Agent 1 and Agent 2 both reach the same grid position together = -100.

The agents start at any random grid location and try to get to the goal. The game ends when either one of the agents reaches their individual goals.

We generate the policies for both agents using the Nash-Q learning algorithm [25]. In Nash-Q learning, we need to solve for a Nash equilibrium in each state. We use the Gambit game theory software [52] to solve the Nash Equilibrium at each state. The Nash-Q learning algorithm has been shown to converge for a high number

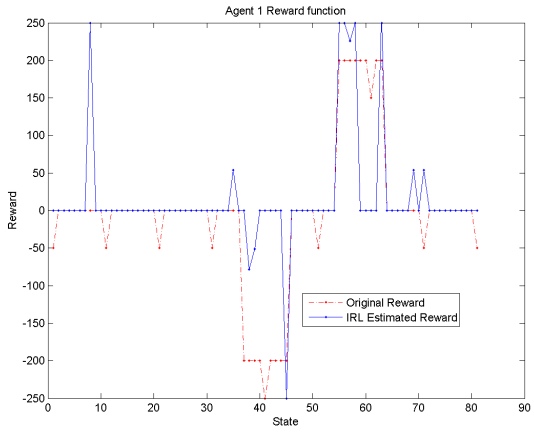

Figure 6.4. Estimated reward for Agent 1.

of iterations. We run the Nash-Q learning algorithm for over 5000 iterations to get to an optimal policy. We then provide the optimal policy from the Nash-Q algorithm to the IRL algorithm that we developed. The estimated reward function for Agent 1 is shown in Figure 6.4, and the estimated reward function for Agent 2 is shown in Figure 6.5. We can see that our IRL captures most of the relevant information about the agents' reward functions. It should be noted that there can be some states for which we will not be able to accurately capture the reward because the policy provided does not transition into these states or the reward is not necessary to enforce the optimal policy. Since all the constraints are met, the original policy is a Markov perfect Nash equilibrium policy for this reward function. But, since Nash-Q can pick between multiple Nash equilibria, it does not always generate the exact same policy.

6.5 Summary

The objective of IRL is to estimate the reward function of an agent, given its policy information and the model of the environment. IRL has been successfully used in various domains such as transfer learning or Apprenticeship learning. Most of the

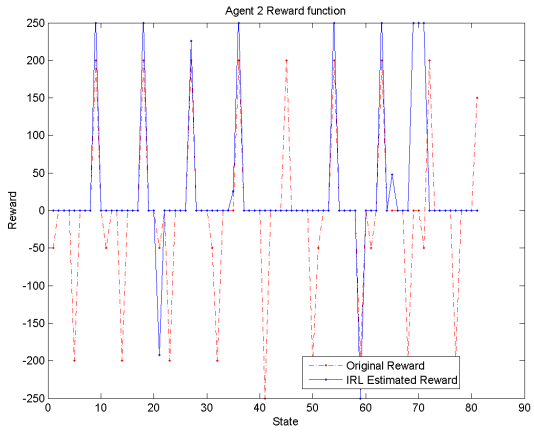

Figure 6.5. Estimated reward for Agent 2.

work done on IRL so far has been in the single agent domain. In this chapter we have presented algorithms for IRL in decentralized multiagent systems.

We derived the necessary conditions for the estimated reward to guarantee the Markov perfect equilibrium of the agents' policies, and presented an optimization problem where we can solve the IRL for each agent in the GSSG\R, when the agents' optimal policies are explicitly given. In the case where the agents' optimal policies are not explicitly given, but trajectories of the optimal policy are provided, we have presented an iterative algorithm for estimating the agents reward function using IRL. We have shown some experimental results validating our work.

Based on these algorithms the agents can estimate the reward function of the other agents and either verify if the reward function that the other agents communicate to it are accurate or use these estimated rewards for planning.

CHAPTER 7

GAME THEORETIC FRAMEWORK FOR COMMUNICATION IN FULLY OBSERVABLE MULTIAGENT SYSTEMS

7.1 Introduction

As mentioned in earlier chapters, communication is a very important part of multiagent systems (MAS). Agents need to communicate to accomplish their tasks. Here we look at communication in non-cooperative MAS modeled as General Sum Stochastic Games (GSSG) mentioned in Chapter 2. Both cooperative and non-cooperative systems can be modelled as GSSGs. The only assumption we make in this chapter is that the MAS is fully observable.

In this chapter we present a methodology to model communication as a game. Using the proposed approach, the agents can learn to communicate effectively such that they minimize the cost of communication, while maximizing their rewards. They do this by estimating when and what to communicate during execution time. The agents learn their policies without communicating and only communicate when they are executing the policies that they have learned.

7.2 Communication in Fully Observable Multiagent System

Our assumption in this chapter is that the world is completely observable. In a fully observable world all agents know the state of the world perfectly. Therefore, the agents do not need to communicate any information about the state of the world. The only piece of information the agents have any benefit of communicating is about what they are going to do next in the world.

The risk dominant equilibrium presented in Chapter 5 is useful in the case when there is no coordination between agents. We now present a case for coordination based on communication between agents. We do not make any assumption about cooperation between agents. The agents can coordinate their actions based on communication only in the case where it is in their self interest.

In the case of fully cooperative systems, the agents can communicate their future actions. In a more general case that is not always possible because in the case of mixed strategy Nash equilibria, if an agent informs another agent about its action, then the equilibrium might allow the other agent to exploit the information revealed by the first agent to play an action which results in higher gain for itself. An example of such a case would be in the repeated game of matching pennies.

7.3 When to Communicate

Before the agents can communicate any information, they must decide *when* they need to communicate. In th communication approach proposed here the agents individually compute their Q-tables based on the NashQ learning algorithm using the risk dominance based equilibrium selection criteria described in Chapter 5.

The agents first consider a $safety\ value$, $Q_i(s, \pi_i a_{-i})$, this is the value that agent i will get if it plays the risk dominant equilibrium and the other players diverge from it. This is the value that the agent is assured to get. When the agent faces a situation where it has to select from multiple equilibria which have the same risk factor, it should choose the one with higher safety value. The agents then compute the value of the risk dominant equilibrium, $Q_i(s, \pi_r)$(using Equation 2.7), and can reasonably expect that the other agents will play this equilibrium, where π_r is the risk dominant equilibrium that is selected by the agent.

The agent then computes the value of its highest value equilibrium, $Q_i(s, \pi_p)$, where π_p, is the highest value Nash equilibrium for the agent (if a payoff dominant equilibrium exists then the agents will select it).

The value of communication, V_c, is then computed as the difference between the value of the risk dominant equilibrium and the expected value of the equilibrium agreed upon after communication, taking C, the cost of communication into account (in other words this is the measure of the benefit that an agent can get by communicating).

$$V_c = V_{pe_1}P(pe_1) - C + (1 - P(pe_1)(V_{pe_2}P(pe_2) - 2C + (1 - P(pe_2)(V_{pe_3}P(pe_3) - 3C + ...)) - V_{\pi_r}$$

Which can be rewritten as,

$$V_c = \sum_{pr=1}^{\infty} \left[\Pi_{l=1}^{pr}(1 - P(pe_l))\right]\left(P(pe_{pr})V(pe_{pr}) - pr \times C\right) - V_{\pi_r} \tag{7.1}$$

where pe_t is the t^{th} Nash equilibrium proposed and pr is the number of proposals. Since we need at least two messages to agree upon π_p (one per agent), the $\max(V_c)$ is defined as

$$\max(V_c) = Q_i(s, \pi_p) - Q_i(s, \pi_r) - C \tag{7.2}$$

If the $\max V_c$ is positive then the agents may communicate.

7.4 What to communicate

Once the agents decide to communicate they are free to communicate any possible equilibrium and the other agents can accept or counter propose a different equilibrium. This process of proposing and counter proposing can be modeled as an

extensive form game where the starting node is any agent that communicates first (we do not focus on which agent communicates first or how they can synchronize communication between agents. There are a vast number of algorithms that deal with these issues already available in the literature).

Based on our definition of an extensive form game, the extensive form communication game (EFCG) is defined as follows:

- N, is the set of agents
- A, is the set of actions, in our case each agent can either propose an equilibrium, accept a proposed equilibrium or choose not to communicate
- H, is a set of nonterminal choice nodes, where the agents can either choose to execute a communication action, to accept a proposed equilibrium, or to terminate communication.
- Z, is the set of terminal nodes, which occur when the agents choose either the accept communication action or decide to not communicate in the parent node.
- $\chi : H \rightarrow 2^A$ is the action function, that assigns to each choice node a set of actions such as, accept the proposal, counter propose, or decide not to communicate.
- $\rho : H \rightarrow N$ is the player function which assigns to each nonterminal node a player n $\in$ N, who chooses an action at that node.
- $\sigma : H \times A \rightarrow H \cup Z$ is the transition function which maps a choice node and an action to a new choice node or terminal node.
- $u = (u_1, ..., u_n)$, where $u_i : Z \rightarrow \mathbb{R}$ maps the terminal nodes to Q-values. The Q-values at the terminal nodes are assigned based on the value of the equilibrium the agents agree upon, discounted by the cost of communication actions.

A Nash equilibrium solution to this extensive form game will give each agent the policy of how to maximize their individual gain. If at any time the agents see that

the value of communicating is lower than that of not communicating, the agents can enter a terminal node using the do not communicate action.

The branching factor, B_f of the game is,

$$B_f = Numberof NashEquilibria + 2$$

A simple heuristic to estimate the depth of the game tree, B_f, can be computed using Equation 7.3 as follows,

$$Depth = (\max(V_c))/(C/2) \tag{7.3}$$

Here we only consider the possible actions till the depth of the tree reaches $Depth$, continuing beyond this point in the tree would allow to get values lower then the risk dominance equilibrium. We do not want any solutions where the value of communication is negative.

The entire process is shown in Algorithm 1. The game can also be modeled as a GSSG, but we believe that modeling it as an extensive form game is more intuitive as it takes the sequence of actions into account, and also remembers the history of the execution of the game. A simple example of this communication game is shown in Figure 7.4. In this example there are two agents denoted by "A" and "B". The agents have two equilibria available to them, the first equilibrium denoted by NE1 with a utility of 100 for agent A and 90 for agent B. The second equilibrium is denoted by NE2 and has a utility of 50 for agent A and 60 for agent B. The agents have 4 actions available to them in this communication game: first, Accept a proposed equilibrium denoted by Ac (or A in action), secondly, a no communication action denoted by NC, and finally propose or counter propose the equilibrium denoted by NE1 and NE2. The figure shown is a game played by agent A communicating first.

Algorithm 1 Accept a List of NEs, Payoff Matrix

numNEs = len(NEs)

for $k = 0$ to $numNEs$ **do**

Compute $Q_i(s, \pi_k)$ for agent i using Equation 2.7

Compute $R_f(\pi_k)$ using Equation 5.4.

Compute safety value $Q(s, \pi_{k_i} a_{-i})$ for agent i

end for

Compute $SelectedNE$ using Algorithm 5.1 for agent i

Compute $max\,(V_c)$ using Equation 7.2

if $max\,(V_c) > 0$ **then**

Compute the Nash equilibrium for the EFCG

Play the EFCG

if Accept a Proposed NE **then**

Return the accepted NE,QvalueofAccepted NE

else if Executed a Do Not Communicate Action **then**

Return the $SelectedNE$,QvalueofSelectedNE

end if

end if

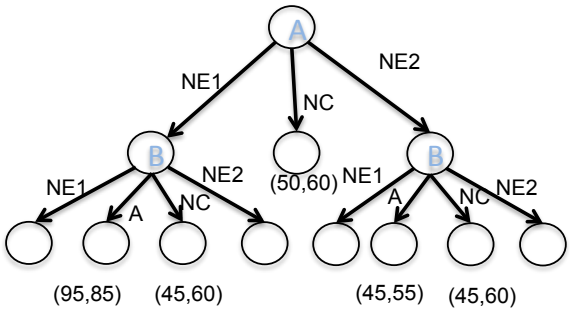

Figure 7.1. A simple communication game modeled as an Extensive Form Game.

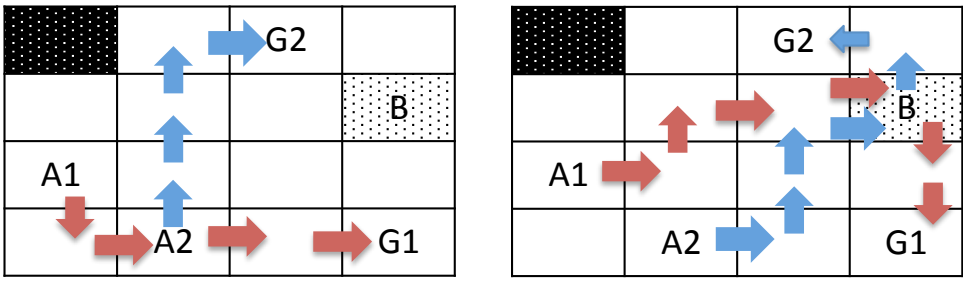

Figure 7.2. Gridworld game: policy without communication (left) and policy with communication (right).

7.5 Experimental Results

We tested our algorithms on a simple grid world problem similar to the one used by Hu in the NashQ implementations and present our preliminary results here. The grid world problem and its NashQ solution are shown in Figure 7.2.

The grid world contains two agents, A1 and A2. The agents must reach their goal states marked G1 and G2. We add a bonus state at cell location (3,4). When both agents reach this state at the same time they get a reward. After that there are no rewards to be gained at the bonus cell. The state is modeled as $((X,Y)_{A1},(X,Y)_{A2},BSR)$, where the first two (x,y) coordinates are the locations of Agent 1 and Agent 2, respectively. BSR is the state of the bonus cell; it is initially set to 0 to indicate that the state has never been visited. BSR is set to 1 if both agents arrive there for the first time, and BSR is set to 2 if both of the agents visit the state more then once at the same time. We also added a strictly negative reward cell (4,1), such that the agents will get a negative reward if they enter that cell location. The reward structure of the game is as follows: If an agent tries to enter the bonus state alone it gets a reward of -200. The agents gain a reward of 100 when they reach their goal cell locations. The agents get a reward of 50 if they enter the bonus cell together for the first time. The agents get a reward of -10 if they enter any cell together. The

agents get a reward of -200 if they enter the strictly negative reward cell. The agents get a reward of -10000 if they try to exit the world.

The result of the NashQ learning algorithm using the risk dominance method for selecting between equilibria without communication is shown on the left in Figure 7.2, where the brown arrows indicate the policy of Agent 1 and blue arrows indicate the policy of Agent 2. In the grid game the bonus state is ignored by both agents because the risk factor of entering the state along with the other agent is too low in the initial states since the loss for any agent that deviates from this strategy is very small compared to the equilibrium where the agents don't attempt to enter the state. To demonstrate this, we present the analysis of the selection process at state ((2,1),(1,2),0):

There are three equilibria; two pure ones, $(right, right)$ denoted as NE_1 and $(down, up)$ denoted as NE_2, and a mixed behavioral Nash equilibrium, $(0.46 : right, 0.54 : down), (0.17 : right, 0.83 : up)$ denoted as NE_3. The utilities for both agents are:

$Q_i(s, NE_1) = (171,\ 171)$

$Q_i(s, NE_2) = (155.2,\ 157.4)$

$Q_i(s, NE_3) = (158,\ 162)$

Based on these values we can see that by coordinating through communication, the agents can reach a higher valued equilibrium. Therefore, the agents create a EFCG and then play it. In this game the cost of communication was selected to be 5 (cost is negative payoff). A small part of the resulting game is shown in Figure 7.3 where the three equilibria are marked as NE1 - NE3, and Agent 1 starts communicating first. The NC and Acc represent not communicate and accept equilibrium actions. At each timestep the agents can propose any equilibrium, accept a proposed equilibrium or counter propose an equilibrium or choose not to communicate. In this game the

agents agree to take the higher valued equilibrium NE1 instead of NE2, ending up with a higher utility. This new equilibrium is shown on the right in Figure 7.2.

7.6 Summary

In this chapter we have presented a methodology to model communication between agents in a stochastic game as an extensive form game. We have also presented an algorithm that can be used to calculate when communication is beneficial to the agents and what information agents should communicate in a fully observable multi-agent system. In our method we make no explicit assumption of cooperation between agents and our method is useful for the domain of non-cooperative systems where there is a cost to communication.

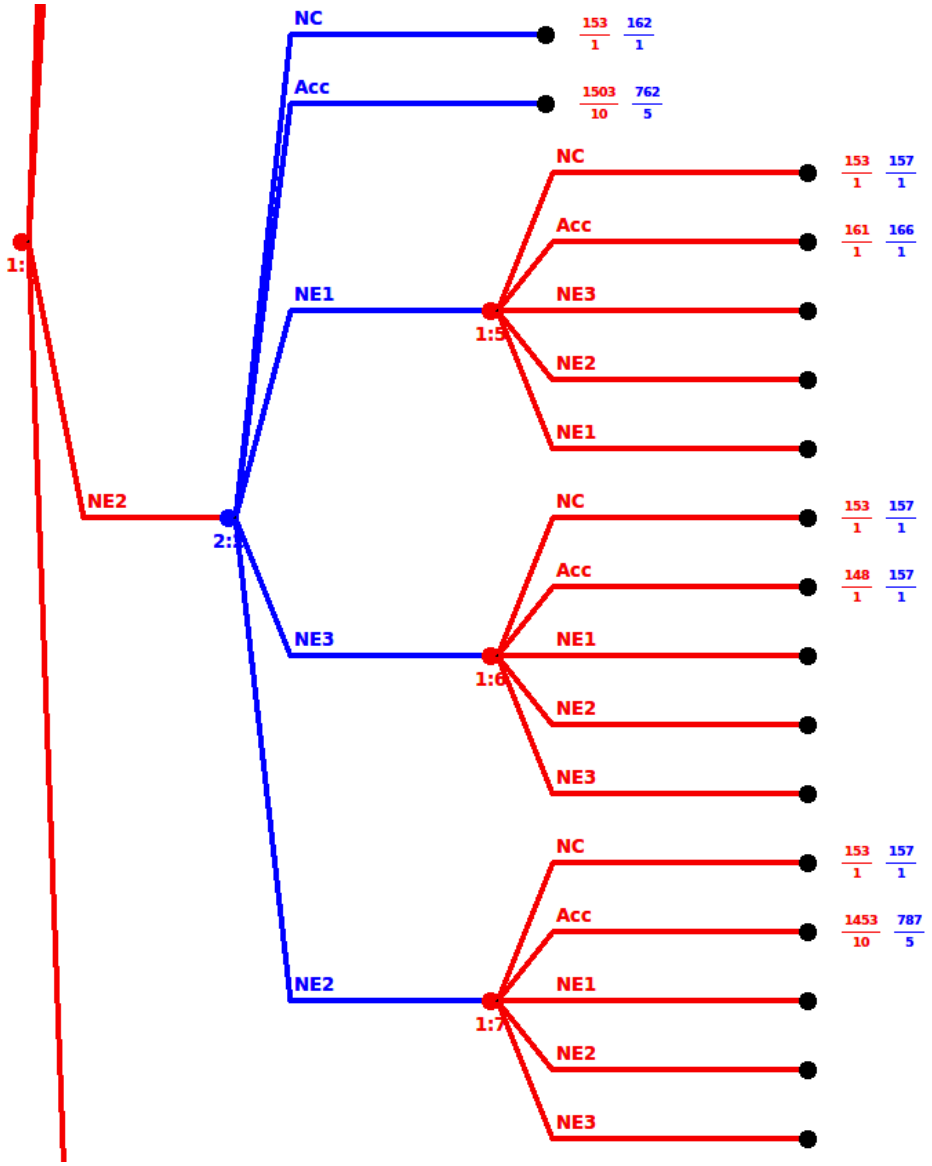

Figure 7.3. A part of the Communication as an Extensive Form Game.

CHAPTER 8

LEARNING WITH COMMUNICATION IN FULLY OBSERVABLE MULTIAGENT SYSTEMS

In the previous chapter we presented a method to model communication as an extensive form game and use this game to achieve higher payoff than what the agents would have gotten if they did not communicate. One of the assumptions that was made in the previous chapter is that all agents will agree on an equilibrium. But, in a decentralized system this is not always true. In this chapter we present an online learning algorithm based on Nash-SARSA [46], that can be used by the agents to learn a policy where each agent can maximize its payoffs.

8.1 Nash-SARSA

Nash-SARSA is a multiagent extension of the single agent SARSA [24] algorithm that is widely used in policy generation for MDPs. In Nash-SARSA, each agent maintains a Q-table for itself and for all other agents. In our approach the agent selects its action based on a Nash equilibrium that it calculates using the risk dominance selection criterion we discussed in Chapter 5. Each agent then plays its part of the Nash equilibrium strategy that it has picked and observes the other agents actions. It then updates its Q-values for the previous state and joint action pairs based on the actions that have been played in the current state. The idea behind this process is that, if all agents select the same Nash equilibrium, then the strategy they each execute will result in a Nash equilibrium.

Unfortunately, if there exists more than one Nash equilibrium, then the agents may select different equilibria to play and the resulting strategy profile might not be a Nash equilibrium. To avoid this miscoordination between the agents, we propose to use communication and online learning using Nash-SARSA to allow agents to agree upon the best equilibrium and to optimize and synchronize their estimates of their utilities . The communication between the agents is modelled using the communication game that was presented in the previous chapter.

One issue that comes up during the practical implementation of this approach is that if the agents' Q-tables prior to this online learning do not converge to the same values, then it might be possible that the Nash equilibrium that one agent selects might not be recognized by the other agents as a Nash equilibrium strategy. In these situations the agents might want to consider if the proposed Nash equilibrium is within an ϵ distance of a known Nash equilibrium. This type of equilibrium is known as $\epsilon - Nash\ equilibrium$ [53]

A Nash equilibrium has an ϵ value of 0.

8.2 Online Algorithm for Communication

The algorithm works on the basis of allowing the agents to first learn offline about the world using the Nash-Q learning algorithm. This happens effectively "in the head" of the agents and it reflects an agent's estimate of the other agents' utilities prior to any interaction. Once the agents have explored the world using the Nash-Q learning algorithm, they start interacting with the other agents by playing the game using the Nash-SARSA algorithm. In each state the agents play the communication game and select their actions based on the outcome of the communication game.

We make one modification to our communication game, we allow the other agents to propose an unknown equilibrium. We then allow each agent to either accept

the proposed unknown equilibrium or counter propose some other equilibrium from the agent's set of equilibria. When an unknown equilibrium is proposed, the agents can estimate if the utility of the best response to the proposed equilibrium is within ϵ of a known equilibrium. Here the distance between the proposed equilibrium and a known equilibrium is calculated by allowing the other agents to play their part of the equilibrium while the agent plays its part of known equilibria. If the proposed equilibrium is within ϵ distance of any known equilibrium then the agent shifts his strategy from the current node of the extensive form game to the node which has the known equilibrium within ϵ distance. If no known strategy is within ϵ distance then the proposed equilibrium is rejected and the execution follows the original strategy. An example of this modified communication game is shown in Figure 8.2.

The agents update their Q-tables based on the Q-values of the equilibrium reached by the communication game

$$Q_i(s^{(t)}, \vec{a}^{(t)}) = (1-\alpha)(Q_i(s^{(t)}, \vec{a}^{(t)})) + \alpha(r_i^{(t)} + \gamma Q_i^c(s^{(t+1)}, \vec{a}^{(t+1)}))\forall i \in N \qquad (8.1)$$

$Q_i^c(s^{(t+1)}, \vec{a}^{(t+1)})$ is the Q-value obtained at a terminal node after playing the communication game.

The agents can learn to coordinate their actions based on Algorithm 8.2. This ensures that the agents can play a Nash equilibrium strategy at every time step. The branching factor of the tree increases by 1 compared to the game in the previous Chapter.

$$B_f = NumofNashEquilibria + 3 \qquad (8.2)$$

Algorithm 2 Nash-SARSA with Communication: Accept The GSSG and Q-tables

$s = s_0$

for $k = 0$ to $numofEpisodes$ **do**

Create a communication game for the current state denoted by s

SelectedNE, QvalueofComm = Play the NE strategy for the communication game

Play the SelectedNE returned by the communication game

The current state of the game advances to next state denoted by $s^{'}$

Update the Q-table entries of the previous state using Eq 8.1 and QvalueofComm

$s = s^{'}$

end for

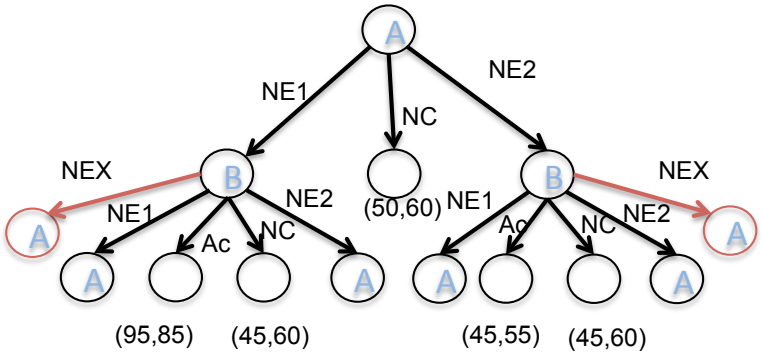

Figure 8.1. A simple communication game modeled as an Extensive Form Game.

8.3 Experimental Results

We have implemented this algorithm on the simple 3 by 3 grid world that was described in the previous chapters. We show the results of the NashQ and Nash-SARSA algorithm in the Figure 8.2.

In Figure 8.2 the grid game on the top left shows the initial game, and the results before communication are shown in the grid game on the bottom left. Similarly, the game on the right shows a different grid game where there is an obstacle in one of the grid locations. In the first game we see that based on the risk dominance method

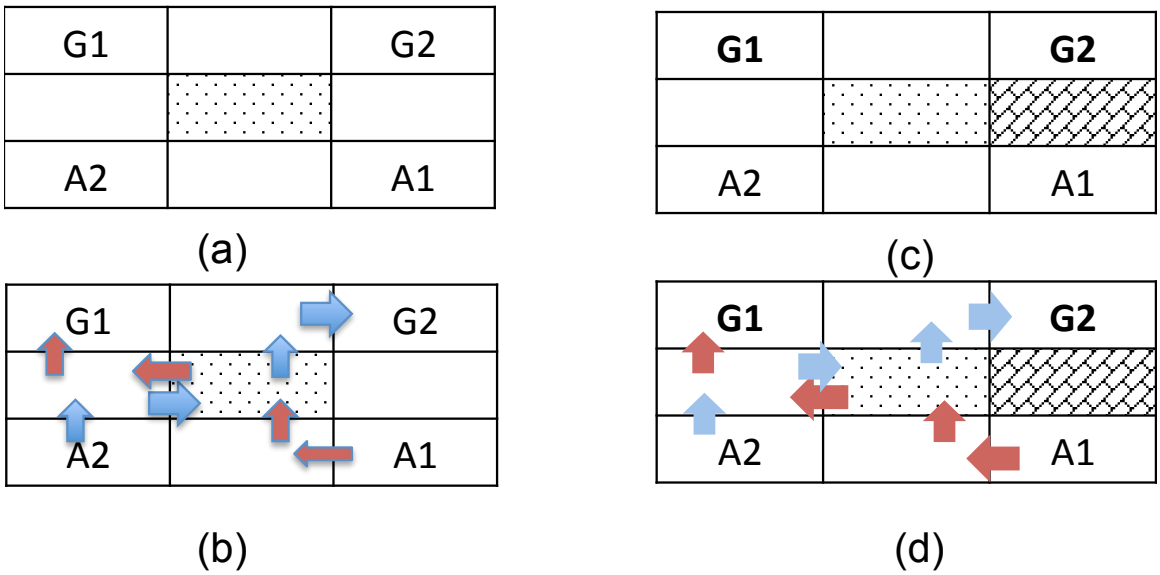

Figure 8.2. Gridworld game example: (a) describes a simple gridworld example. (b) shows the policies learned by the agents. (c) shows a different example of the gridworld. (d) shows the policies learned by the agents.

of equilibrium selection both agents select the same equilibria in their path. In the second game the agents select almost the same equilibria. In the second time step while the first agent selects a pure strategy of going (Up,Right), the second agent takes a mixed strategy of going ((Right-0.17, Up-0.83), (Right-0.91, Up-0.09)). This is mostly because of a lack of exploration in the Nash-SARSA part of our approach. Both agents have both equilibria (with a small difference in probability of the mixed strategy) in their set of equilibria.

In Figure 8.2, we have shown two examples of the Nash-SARSA algorithm which does not allow for communication. When we allow for communication in Nash-SARSA using the method described in this chapter, we see that agents get higher rewards in the case of the game (a) shown in the Figure 8.2. This is shown in the Figure 8.3.

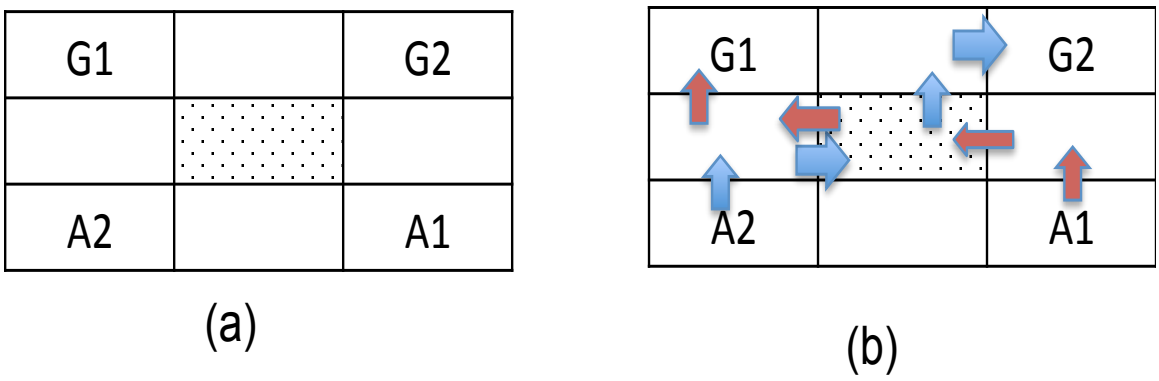

Figure 8.3. Gridworld game example: (a) Describes a simple gridworld example. (b) Shows the policies learned by the agents when communication is allowed.

8.4 Summary

We have presented an online learning algorithm based on Nash-SARSA and the communication game described in the previous chapter. This concludes our discussion on communication in fully observable domains. We now look at the partially observable domain in the next chapter.

CHAPTER 9

LEARNING WITH COMMUNICATION IN PARTIALLY OBSERVABLE MULTIAGENT SYSTEMS

Planning in partially observable scenarios is much harder than in a fully observable world. Most of the current approaches to solving partially observable multiagent systems are not accurate. In most cases the algorithms proposed either solve for an approximate solution that works in certain cases such as in the case of cooperative domains, i.e. domains with common payoffs, or they address small problems which do not scale. The partially observable MAS is generally modelled as a Partially Observable Stochastic Game (POSG).

Before we look at what to communicate in a POSG, we first need to look at how the POSGs are to be solved and more importantly, what information do the agents base their decisions on.

9.1 Planning in POSGs

We first look at previous algorithms that have been proposed to solve the POSG. The first approach is the one proposed by Bernstein et al [54]. In their approach they consider a POSG where all agents get observations from all other agents. This ensures that at each time step all agents know what observations the other agents are getting and thus they can accurately update the belief states of all agents plus they have the same belief state. This allows them to plan based on belief state.

In the more general definition of a POSG, all agents do not get the observations of all agents. This forces the agents to estimate the belief state of the other agents

based on its own observations. Since the agents are not able to estimate the belief state of the other agents accurately, the agents must take a more conservative strategy while calculating their Nash equilibrium.

9.2 Linear Nash-Q

As mentioned in Chapter 2, a POSG is defined as a tuple $\langle S, N, A, Z, T, O, R, \gamma \rangle$. In our definition, Z_i is the set of observations for agent i, and $O(\vec{z}|s, \vec{a})$ denotes the probability of observing $\vec{z}$ by taking joint action $\vec{a}$ in state s resulting in state s'. Note that we do not assume that all agents get the entire observation vector $\vec{z}$.

We solve the problem of learning in POSGs in multiple steps. To obtain a first approximation, we solve the underlying GSSG using NashQ, which gives us the Q-values of state action pairs following the optimal policy for the GSSG denoted by, $Q_i(s, \vec{a})$. We then use a variation of Q-MDP [55] for multiagent systems to get the approximate Q values of all the belief states by using the following equation.

$$Q_i(b, \vec{a}) = \sum_s b_i(s) Q_i(s, \vec{a}) \tag{9.1}$$

As in the case of Q-MDP for POMDP, this initial approximation assumes that after the next action any uncertainty about the agent's current belief state will be gone. This is not a very realistic assumption, but nonetheless it will allow us to obtain a first estimate for the POSG.

The Bellman equations for general sum stochastic games can be modified for the case of POSGs similar to the case of POMDP. After the calculation of the Q-values of the belief states, we use a variation of linear Q-learning [55] for multiagent systems. The updates in linear Nash-Q learning are similar to the updates in NashQ. The update rule used in linear Nash-Q learning is,

$$Q^{(t+1)}(b,\vec{a}) = (1-\alpha)Q^{(t)}(b,\vec{a}) + \alpha(r(b) + \gamma Nash_{\vec{a}}Q^{(t)}(b^{'},\vec{a^{'}})) \tag{9.2}$$

or it can be rewritten as,

$$\Delta Q(b,\vec{a}) = \alpha(r(b) + \gamma Nash_{\vec{a}}Q^{(t+1)}(b^{'},\vec{a^{'}}) - Q^{(t)}(b,\vec{a})) \tag{9.3}$$

The linear function approximation is,

$$Q_i(b,\vec{a}) = \sum_s b_i(s)q_i(s,\vec{a}) \tag{9.4}$$

and the parameters $q_i(s,\vec{a})$ are updated according to

$$\Delta q(s,\vec{a}) = b_i(s)\Delta Q(b,\vec{a}) \tag{9.5}$$

The belief state of each agent can be updated using just the observation z_i that each agent gets using the the following equations.

$$b_i(s^{'}) = \alpha O(z_i|s^{'},\vec{a}) \sum_s T(s^{'}|s,\vec{a})b_i(s) \tag{9.6}$$

where α is a normalizing constant that is calculated by the following equation,

$$\alpha = \frac{1}{\sum_{s^{'}\in S} O(z_i|s^{'},\vec{a}) \sum_{s\in S} T(s^{'}|s,\vec{a})b_i(s)} \tag{9.7}$$

This algorithm was selected because we need the Q-values of the belief states that can be used to measure the impact of the information that is communicated.

Each agent computes the Nash equilibrium based on their belief, they can estimate the Q-values by taking a weighted average of the Q-values of individual states based on their probabilities and then calculate Nash equilibrium over these values. The second approach is to compute the Nash equilibria of each state individually and then choose the Nash equilibrium which has the highest cumulative probability based on the probabilities assigned to the states.

9.3 Belief State Update

For belief state updates we use a bayes filter based approach. Each agent can maintain multiple beliefs depending on the information that it uses to update the belief states. As mentioned earlier each agent only gets its part of the observation vector instead of the observations for all agents. Each agent can get different observations which causes the agents to have different belief states. An environment where agents have different belief states makes planning difficult as a Nash equilibrium computed by one agent can be completely different from the Nash equilibrium computed by the other agents.

To make planning more accurate across all agents, each individual agent can compute their belief based on not only the observation that they get from the environment but also a probabilistic estimate of the other agents' observations. The agents can also compute their belief based on a probabilistic estimate of the other agents observations and their estimate of the other agents' estimate about their own observations. The agents should try to use as much of the common information as possible as this ensures that the belief state across agents is more consistent. The different belief states considered and their updated process is listed below.

- Each agent i can maintain a belief about its own state b_i^- based on its own observations z_i and the joint actions $\vec{a}$.

$$b_i^{-'}(s') = P(s'|z_i, \vec{a}b_i^-) \tag{9.8}$$

  This can be expressed as,

$$b_i^{-'}(s') = \alpha O(z_i|s', \vec{a}) \sum_s T(s'|s, \vec{a}) b_i^-(s) \tag{9.9}$$

- Each agent can maintain a belief about itself and the other agent denoted as b_i^+, based on its own observations z_i, a probabilistic estimate of the other agents'

possible observations $\hat{z_j}$ and the joint action $\vec{a}$. $\{b_i^{+k}\}$ is a set of belief states where k is used to denote one possible observation in the set of possible observations denoted by $\hat{z_j}^k$. k is dependent on $\hat{z_j}$ and $\bar{k}$ where, $\bar{k}$ is the predecessor belief state out of the set of belief states in the previous timestep and $\hat{z_j}$ is the observation in the current timestep.

$$b_i^{+k}(s^{'}) = P(s^{'}|z_i, \hat{z_{-i}}^k, \vec{a}, b_i^{+\bar{k}}) \tag{9.10}$$

This can be expressed as,

$$b_i^{+k}(s^{'}) = \beta \left(\prod_{j \neq i} O(\hat{z_j}^k | s^{'}, \vec{a}) \right) O(z_i | s^{'}, \vec{a}) \sum_s T(s^{'}|s, \vec{a}) b_i^{+\bar{k}}(s) \tag{9.11}$$

$$\beta = \frac{1}{\sum_{s^{'} \in S} \left(\prod_{j \neq i} O(\hat{z_j}^k | s^{'}, \vec{a}) \right) O(z_i | s^{'}, \vec{a}) \sum_{s \in S} T(s^{'}|s, \vec{a}) b_i^{+\bar{k}}(s)} \tag{9.12}$$

where k is used to denote an index in the set of possible observations $\hat{z_j}^k$.

$$P(b_i^{+k}) = \left(\sum_{s^{'}} \left(\prod_{j \neq i} O(\hat{z_j}^k | s^{'}, \vec{a}) \right) \alpha O(z_i | s^{'}, \vec{a}) \sum_s T(s^{'}|s, \vec{a}) b_i^{+\bar{k}}(s) \right) P(b_i^{+\bar{k}}) \tag{9.13}$$

$$\alpha = \frac{1}{\sum_{s^{'} \in S} O(z_i | s^{'}, \vec{a}) \sum_{s \in S} T(s^{'}|s, \vec{a}) b_i^{+\bar{k}}(s)} \tag{9.14}$$

- Each agent i also maintains an estimate of the other agents' belief denoted by $\hat{b}_{i,j}^{-}$ based on the other agents possible observations $\hat{z_j}$ given the initial agent's current observation z_i. This is the estimate of the other agents b_j^{-}. $\{\hat{b}_{i,j}^{-}{}^k\}$ is a set of belief states where k is used to denote one possible observation in the set of possible observations denoted by $\hat{z_j}^k$

$$\hat{b}_{i,j}^{-}{}^k(s^{'}) = P(s^{'}|\hat{z_j}^k, z_i, \hat{b}_{i,j}^{-}{}^{\bar{k}}, \vec{a}) \tag{9.15}$$

This can be expressed as,

$$\hat{b}_{i,j}^{-\ k}(s') = \beta O(\hat{z}_j^k|s',\vec{a})O(z_i|s',\vec{a})\sum_s T(s'|s,\vec{a})\ \hat{b}_{i,j}^{-\ \bar{k}}(s) \tag{9.16}$$

$$\beta = \frac{1}{\sum_{s'\in S} O(\hat{z}_j^k|s',\vec{a})O(z_i|s',\vec{a})\sum_{s\in S} T(s'|s,\vec{a})\ \hat{b}_{i,j}^{-\ \bar{k}}(s)} \tag{9.17}$$

where k is used to denote an index in the set of possible observations $\hat{z}_j$.

$$P\left(\hat{b}_{i,j}^{-\ k}\right) = \left(\sum_{s'} O(\hat{z}_j{}^k|s',\vec{a})\alpha O(z_i|s',\vec{a})\sum_s T(s'|s,\vec{a})\ \hat{b}_{i,j}^{-\ \bar{k}}(s)\right) P(\hat{b}_{i,j}^{-\ \bar{k}}) \tag{9.18}$$

$$\alpha = \frac{1}{\sum_{s'\in S} O(z_i|s',\vec{a})\sum_{s\in S} T(s'|s,\vec{a})\ \hat{b}_{i,j}^{-\ \bar{k}}(s)} \tag{9.19}$$

- Each agent i also maintains an estimate of the other agents' belief denoted by $\hat{b}_{i,j}^{+}$ based on the other agents possible observations $\hat{z}_j$ given the initial agent's current observations and an estimate of the initial agent's observations $\hat{z_{-j}}$ based upon the estimation of the other agents possible observations $\hat{z}_j$. This is the estimate of the other agents b_j^+. This is also a set of belief states, denoted by $\{\hat{b}_{i,j}^{+\ k}\}$, based on a possible observation from the set of possible observations.

$$\hat{b}_{i,j}^{+\ k}(s') = P(s'|\hat{z_j}{}^k, \hat{z}_{-j}^k, z_i, \hat{b}_{i,j}^{+\ \bar{k}}, \vec{a}) \tag{9.20}$$

This can be expressed as,

$$\hat{b}_{i,j}^{+\ k}(s') = \beta\prod_l O(\hat{z}_l|s',\vec{a})\sum_s T(s'|s,\vec{a})\hat{b}_{i,j}^{+k}(s) \tag{9.21}$$

$$\beta = \frac{1}{\sum_{s' \in S} \prod_l O(\hat{z_l}^k | s', \vec{a}) \sum_s T(s'|s, \vec{a}) \hat{b}_{i,j}^{+k}(s)} \tag{9.22}$$

where k is used to denote an index in the set of possible observations $\hat{z_j}$ and l denotes an agent.

$$P\left(\hat{b}_{i,j}^{+\ k}\right) = P\left(\hat{z}_j^k | z_i, \hat{b}_{i,j}^{+\ \bar{k}}\right) P\left(\hat{z}_{-j}^k | \hat{z}_j^k, \hat{b}_{i,j}^{+\ \bar{k}}\right) P\left(\hat{b}_{i,j}^{+\ \bar{k}}\right)$$

$$P\left(\hat{b}_{i,j}^{+\ k}\right) = \left(\sum_{s'} O(\hat{z_j}^k | s', \vec{a}) \alpha O(z_i | s', \vec{a}) \sum_s T(s'|s, \vec{a})\ \hat{b}_{i,j}^{+\ \bar{k}}(s)\right) \left(\sum_{s'} \prod_{l \neq j} O(\hat{z_l}^k | s', \vec{a}) \gamma O(\hat{z_j}^{\ k} | s', \vec{a}) \sum_s T(s'|s, \vec{a})\ \hat{b}_{i,j}^{+\ \bar{k}}(s)\right) P\left(\hat{b}_{i,j}^{+\ \bar{k}}\right) \tag{9.23}$$

$$\alpha = \frac{1}{\sum_{s' \in S} O(z_i | s', \vec{a}) \sum_{s \in S} T(s'|s, \vec{a})\ \hat{b}_{i,j}^{+\ \bar{k}}(s)} \tag{9.24}$$

$$\gamma = \frac{1}{\sum_{s' \in S} O(\hat{z_j}^{\ k} | s', \vec{a}) \sum_{s \in S} T(s'|s, \vec{a})\ \hat{b}_{i,j}^{+\ \bar{k}}(s)} \tag{9.25}$$

Here we assume that each agent only gets its own observation z_i and not the entire set of observations $\vec{z}$. We also assume that each agent can perceive the entire joint action vector $\vec{a}$. If each agent would get the entire observation vector $\vec{z}$ then all agents would have perfect understanding of the belief of the other agents (note that this assumption does not make it a fully observable MAS). When all agents have a good understanding of the belief of the other agents, then they can plan effectively using a Nash equilibrium policy over the belief states of the agents.

The question of planning becomes, should an agent plan based on b_i^- *or* b_j^+ *or* $\hat{b}_{i,j}^-$ *or* $\hat{b}_{i,j}^+$. In principle $\hat{b}_{i,j}^+$ is the belief that is closest to being common across all

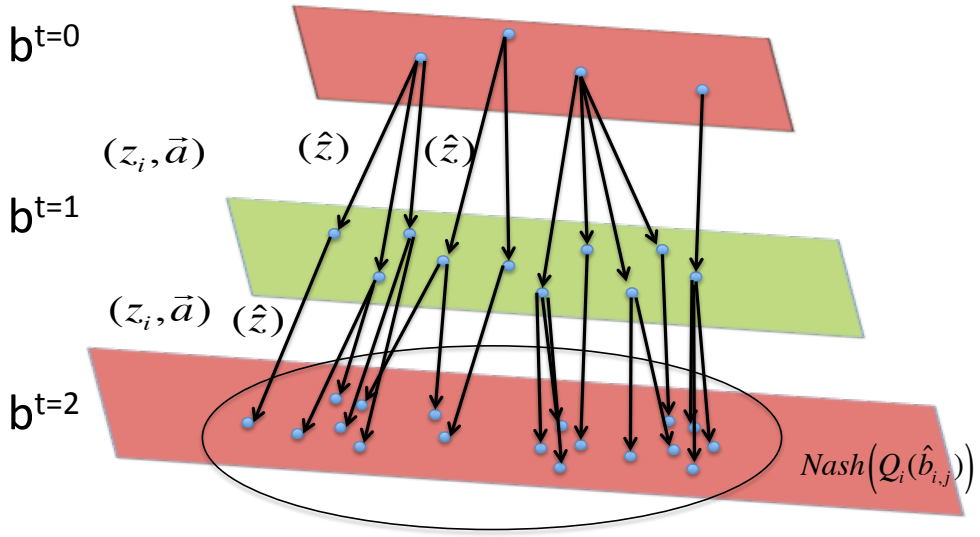

Figure 9.1. Belief tracking without communication.

agents (as it does not use any information an agent gets directly, but estimates common information based on the information the agents receive from the environment), therefore the agents here will plan based on this belief. In principle if $\hat{b}^{+}_{i,j}$ comes closer to b^{-}_{i} then we can plan most efficiently as all agents use the most information obtained from the environment.

In Figure 9.1, the belief state tracking process is shown where the agents start from some belief state and after each time step updates the belief based on the probabilistic observation it gets and its estimates for the other agents as discussed above.

9.4 Communication in POSG

As mentioned in the previous section, the closer the agents' beliefs about the other agents is to what the other agents believe their own belief is, the more accurately we can plan.

Based on this we can now define our vocabulary (set of information that is beneficial to be communicated). From above we can say that the agents might have benefit in communicating their observations at any time step, their current belief, and the action they have taken. Once the agents have synchronized their belief, the agents can still communicate the Nash equilibrium they would like to take (this is similar to the case of the fully observable MAS described in the previous chapters). The agents must make the decision of when to communicate based on the Q values over the belief states.

9.5 Communication Game

We consider three stages for the communication game. In the first state the agent considers the observations that it has received from the environment. It can choose to communicate its observations based on how the observation can impact its own utility. The second stage of communication would be to transmit its actual beliefs b_i^-, this allows the other agents to set their beliefs about the agent expressed as $\hat{b}_{i,j}^+$ to b_i^-. And, finally in the third stage the agents can choose to communicate the equilibrium they would like to play (as in the case of fully observable MAS).

The agents must make a decision to communicate or not to communicate in each time step, and then use either of the three stages to communicate their information. Should the agents select to communicate their information, then the agents can make good decisions in the system. The first two step of the process are modeled as single shot games and the third step is modeled as the extensive form game as in the case of a fully observable multiagent system.

In Figure 9.2 we have shown the process of tracking the belief state by allowing the agents to communicate their observations z_i at each time step. Given the communication of an observation z_i, all branches in the belief tracking tree that are not

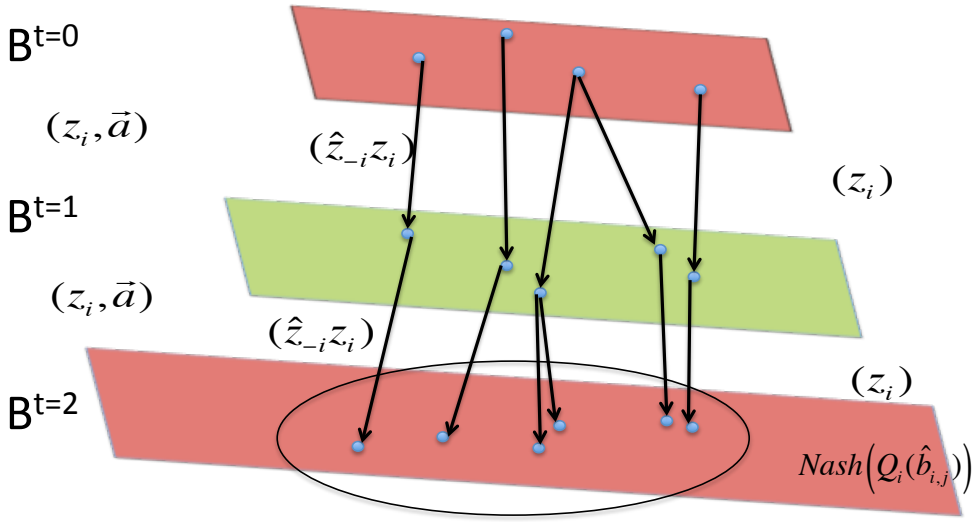

Figure 9.2. Belief tracking with observation communication.

consistent with this observation are removed and the probabilities of the remaining branches,and thus of the remaining potential belief states, are re-normalized. Computing the difference of the value of the Nash equilibrium of the aggregated belief state of the resulting set and of the value of the equilibrium of the aggregated belief state before communication represents the value of communicating the observation, allowing the agents to make a decision when and what to communicate. Note that even by communicating their individual observations at each time step the system does not reduce to a fully observable multiagent system.

We have made a few assumptions in our approach, such as the fact, that the agents always communicate truthful information and the agents always trust the information that is proposed by the other agents. In principle the issue of trust between agents is an important area of research that would impact what information they communicate with the other agent. We leave this issue as future work.

CHAPTER 10

CONCLUSION

In this dissertation, we have presented a framework based on which agents can make decision about communication actions in a multiagent system. We have also provided algorithms for inverse reinforcement learning in decentralized multiagent system, and we have also provided enhancements to the popular NashQ and Nash-SARSA algorithms based on *Risk Dominance in equilibrium selection*.

The communication of strategies between agents was modeled as a extensive form game and it was successfully used to improve the utility of the agents. We have provided experimental results for our algorithms in the case of a fully observable multiagent system. For partially observable domains we have also introduced a new algorithm to compute the Q-values over belief states, a new representation to establish different, more coordinated belief satte estimates, and an algorithm to determine when to communicate observation and which observation to communicate.

Mathematical foundations of this framework were introduced in the case of partially observable stochastic game, though, experimental results have not been included.

10.1 Future Work

The framework presented here is meant to be used as a base for more extensive work on the study of communication decision making in multiagent systems. The learning algorithm for fully observable multiagent system can be further enhanced by

the consideration of the quantal response equilibrium (this is not an enhancement of Nash equilibrium).

The algorithms for inverse reinforcement learning can be improved much further by using gradient techniques, and can be extended to the case of centralized POSGs. Though it remains to be seen if IRL methods can be applied to decentralized POSGs.

Finally, we have introduced the framework for communication decision making in partially observable multiagent systems, but, we have made some unrealistic assumptions about trust between agents. Since the issue of trust between agents has a direct impact on the informations that the agents can use to communicate, we hope to extending our framework by relaxing this constraint in the future.

REFERENCES

[1] I. J. McEwan, "Edited by gerhard krauss. biochemistry of signal transduction and regulation. wileyvch, 2001, 2nd edn, 506 pp. price 45.00. isbn 3-527-30378-2," *Applied Organometallic Chemistry*, vol. 16, no. 11, pp. 675–675, 2002.

[2] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2009.

[3] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. New York, NY, USA: Cambridge University Press, 2008.

[4] B. CHAIB and F. DIGNUM, "Trends in agent communication language," *Computational Intelligence*, vol. 18, no. 2, 2002.

[5] J. Pitt and A. Mamdani, "Communication protocols in multi-agent systems: a development method and reference architecture," *Issues in agent communication*, pp. 160–177, 2000.

[6] J. Dale and E. Mamdani, "Open standards for interoperating agent-based systems," *Software Focus*, 2001.

[7] M. Roth, R. Simmons, and M. Veloso, "What to communicate? execution-time decision in multi-agent pomdps," in *The 8th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2006.

[8] P. Parikh, *Language and Equilibrium*. The MIT Press, 2010.

[9] N. Allott, "Game theory and communication."

[10] L. S. Shapley, "Stochastic games," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953.

[11] T. S. Reddy, V. Gopikrishna, G. V. Zaruba, and H. Manfred, "Inverse reinforcement learning for decentralized non-cooperative multiagent systems," in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, oct. 2012.

[12] T. S. Reddy, G. V. Zaruba, and H. Manfred, "Game theoretic framework for communication in fully observable multiagent systems," in *Machine Learning and Applications, 2012. ICMLA '12. Eleventh International Conference on*, 2012.

[13] J. Hu and M. P. Wellman, "Nash q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, Dec. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=945365.964288

[14] T. Rahwan, *Algorithms for Coalition Formation in Multi-Agent Systems*, ser. British Comp Society Series. British Informatics Society Limited, 2009. [Online]. Available: http://books.google.com.pe/books?id=zBxTPgAACAAJ

[15] Y. Labrou and T. Finin, "Semantics and conversations for an agent communication language," in *Readings in Agents*, M. N. Huhns and M. P. Singh, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 235–242. [Online]. Available: http://portal.acm.org/citation.cfm?id=284907

[16] S. Kraus, *Strategic Negotiation in Multiagent Environments*, ser. Intelligent Robots And Autonomous Agents. MIT Press, 2001. [Online]. Available: http://books.google.com/books?id=of3W6dlL_zoC

[17] D. Shuai and X. Feng, "Distributed problem solving in multiagent systems: A spring net approach," *IEEE Intelligent Systems*, vol. 20, no. 4, pp. 66–74, July 2005. [Online]. Available: http://dx.doi.org/10.1109/MIS.2005.68

[18] N. R. Jennings and M. Wooldridge, "Agent-oriented software engineering," *ARTIFICIAL INTELLIGENCE*, vol. 117, pp. 277–296, 2000.

[19] M. Wooldridge and P. Ciancarini, "Agent-oriented software engineering: the state of the art," *Lecture notes in computer science*, pp. 1–28, 2001.

[20] W. Sheng, Q. Yang, J. Tan, and N. Xi, "Distributed multi-robot coordination in area exploration," *Robotics and Autonomous Systems*, vol. 54, no. 12, pp. 945–955, 2006.

[21] R. Fierro, A. Das, J. Spletzer, J. Esposito, V. Kumar, J. Ostrowski, G. Pappas, C. Taylor, Y. Hur, R. Alur, *et al.*, "A framework and architecture for multi-robot coordination," *The International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 977–995, 2002.

[22] H. McMahan, "Robust planning in domains with stochastic outcomes, adversaries, and partial observability," Ph.D. dissertation, Stanford University, 2006.

[23] F. Bellifemine, A. Poggi, and G. Rimassa, "Developing multi-agent systems with jade," *Intelligent Agents VII Agent Theories Architectures and Languages*, pp. 42–47, 2001.

[24] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. Cambridge Univ Press, 1998, vol. 1, no. 1.

[25] J. Hu and M. P. Wellman, "Nash q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, Dec. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=945365.964288

[26] A. R. Greenwald and K. Hall, "Correlated Q-Learning," in *International Conference on Machine Learning*, 2003, pp. 242–249.

[27] M. L. Littman, "Friend-or-foe q-learning in general-sum games," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 322–328. [Online]. Available: http://dl.acm.org/citation.cfm?id=645530.655661

[28] M. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the eleventh international conference on machine learning*, vol. 157, 1994, p. 163.

[29] L. Busoniu, R. Babuska, and B. De Schutter, "Multi-agent reinforcement learning: A survey," in *Control, Automation, Robotics and Vision, 2006. ICARCV '06. 9th International Conference on*, dec. 2006, pp. 1 –6.

[30] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670. [Online]. Available: http://dl.acm.org/citation.cfm?id=645529.657801

[31] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *In Proceedings of the Twenty-first International Conference on Machine Learning*. ACM Press, 2004.

[32] G. Neu and C. Szepesvari, "Apprenticeship learning using inverse reinforcement learning and gradient methods," in *Proceedings of the Twenty-Third Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-07)*. Corvallis, Oregon: AUAI Press, 2007, pp. 295–302.

[33] D. V. Pynadath and M. Tambe, "The communicative multiagent team decision problem: analyzing teamwork theories and models," *Journal of Artificial Intelligence Research*, vol. 16, no. 1, pp. 389–423, June 2002.

[34] P. Xuan, V. Lesser, and S. Zilberstein, "Formal Modeling of Communication Decisions in Cooperative Multi-agent Systems," *Proceedings of the Second Workshop on Game Theoretic and Decision Theoretic Agents (GTDT 2000)*, January 2000.

[35] C. V. Goldman and S. Zilberstein, "Optimizing information exchange in cooperative multi-agent systems," in *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, ser. AAMAS '03. New York, NY, USA: ACM, 2003, pp. 137–144.

[36] C. V. Goldman and S. Zilberstein, "Decentralized control of cooperative systems: categorization and complexity analysis," *Journal of Artificial Intelligence Research*, vol. 22, no. 1, pp. 143–174, Nov. 2004.

[37] M. T. J. Spaan, F. A. Oliehoek, and N. A. Vlassis, "Multiagent planning under uncertainty with stochastic communication delays," in *ICAPS'08*, 2008, pp. 338–345.

[38] M. Roth, R. Simmons, and M. Veloso, "Reasoning about joint beliefs for execution-time communication decisions," in *The Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS)*, 2005.

[39] R. Nair, M. Roth, and M. Yohoo, "Communication for improving policy computation in distributed pomdps," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3*, ser. AAMAS '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 1098–1105.

[40] F. Wu, S. Zilberstein, and X. Chen, "Multi-agent online planning with communication," *Proceedings of ICAPS-2009*, 2009.

[41] R. Emery-Montemerlo, "Game-theoretic control for robot teams," Ph.D. dissertation, Carnegie Mellon University, 2005.

[42] A. Beynier and A.-I. Mouaddib, "A rich communication model in opportunistic decentralized decision making," in *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 02*, ser. WI-IAT '10. Washington, DC, USA: IEEE Computer

Society, 2010, pp. 133–140. [Online]. Available: http://dx.doi.org/10.1109/WI-IAT.2010.41

[43] A. Burkov and B. Chaib-draa, "Distributed planning in stochastic games with communication," in *Machine Learning and Applications, 2008. ICMLA '08. Seventh International Conference on*, dec. 2008, pp. 396 –401.

[44] C. Murray and G. J. Gordon, "Multi-robot negotiation: Approximating the set of subgame perfect equilibria in general-sum stochastic games," in *NIPS'06*, 2006, pp. 1001–1008.

[45] R. Emery-Montemerlo, G. Gordon, and S. Thrun, "Run-time communication policies for partially observable stochastic games."

[46] B. Banerjee, S. Sen, and J. Peng, "On-policy concurrent reinforcement learning," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 16, no. 4, pp. 245–260, 2004.

[47] J. C. Harsanyi and R. Selten, *A General Theory of Equilibrium Selection in Games*, ser. MIT Press Books. The MIT Press, 1988, vol. 1, no. 0262582384.

[48] J. C. Harsanyi, "A new theory of equilibrium selection for games with complete information," *Games and Economic Behavior*, vol. 8, no. 1, pp. 91 – 122, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0899825605800181

[49] M. Kandori, G. J. Mailath, and R. Rob, "Learning, mutation, and long run equilibria in games," *Econometrica*, vol. 61, no. 1, pp. 29–56, 1993.

[50] M. Kandori, G. J. Mailath, and R. Rob, "Learning, mutation, and long run equilibria in games," *Econometrica*, vol. 61, no. 1, pp. pp. 29–56, 1993.

[51] S. Natarajan, G. Kunapuli, K. Judah, P. Tadepalli, K. Kersting, and J. Shavlik, "Multi-agent inverse reinforcement learning," in *Proceedings of the 2010 Ninth International Conference on Machine Learning and Applications*, ser. ICMLA

'10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 395–400. [Online]. Available: http://dx.doi.org/10.1109/ICMLA.2010.65

[52] R. D. Mckelvey, A. M. Mclennan, and T. L. Turocy, "Gambit: Software Tools for Game Theory, Version 0.2010.09.01," Tech. Rep., 2010. [Online]. Available: http://www.gambit-project.org.

[53] H. Dixon, "Approximate bertrand equilibria in a replicated industry," *The Review of Economic Studies*, vol. 54, no. 1, pp. pp. 47–62, 1987.

[54] E. Hansen, "Dynamic programming for partially observable stochastic games," in *IN PROCEEDINGS OF THE NINETEENTH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, 2004.

[55] M. LITTMAN, "Learning policies for partially observable environments: Scaling up," in *Proc. of International Conference on Machine Learning, 1995*, 1995.

BIOGRAPHICAL STATEMENT

Tummalapalli Sudhamsh Reddy received his B.E. in Computer Science and Engineering in 2003 from Vellore Engineering Collage (Previously affiliated with Madras University, now known as Vellore Institute of Technology), Vellore, India. He got his M.S in Computer Science in 2006 from the University of Texas at Arlington (UTA). Sudhamsh is currently pursuing his PhD in Computer Science under the supervision of Dr. Manfred Huber.

Sudhamsh has previously worked at Fermi National Accelerator Lab for a year as a Graduate Research Assistant employed by UTA, and at Brookhaven National Labs for one and half years while employed as Research Associate Scientist 2 by UTA. He has worked on Grid computing software development for the Dzero and Atlas physics experiments.