# Multiagent Learning in the Presence of Agents with Realistic Limitations

Michael Bowling
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890
mhb+@cs.cmu.edu

April 7, 2000

## Thesis Proposal

**Abstract**

Applications involving or requiring multiple agents (e.g. robotic soccer, search and rescue, automated driving, information agents) are becoming common place as physical and software robots are more prevalent. Environments of multiple intelligent agents present a challenging problem of finding a successful course of action, that is a policy that maximizes reward or goal achievement. Online learning of control policies has great potential for addressing these domains. Reinforcement learning has already been extensively studied and successfully applied to single-agent tasks where the dynamics of the world are stationary. In multiagent environments, the dynamics depend on the actions selected by the other agents. Since the behavior of the other agents can change as they also adapt and learn, the dynamics of the world are not stationary. Hence, in multiagent environments an optimal policy depends on the behavior of the other agents. This is captured by the game theoretic framework of stochastic games.

Previous approaches to learn in multiagent environments either apply naive single-agent learning techniques or attempt to directly learn an equilibrium policy independent of the behavior of the other agents. Single-agent techniques constantly adapt toward optimal responses to the other players' policies, but have no guarantee of converging to a stationary policy. They are rational but not necessarily convergent. Equilibrium learning techniques will converge to a stationary policy, but that policy may not be the optimal response to the players' particular behavior. They are convergent but not rational.

I propose to continue my development of learning algorithms that are both rational and will converge to stationary strategies. I have introduced a preliminary algorithm towards achieving these properties. The algorithm is proven rational and in empirical results converges for a wide variety of problems. My thesis research will use these techniques to focus on multiagent environments where the other agents have physical or rational limitations. I will build upon some of my initial research on performance bounded learning with subproblems, which can be viewed as a limitation on the agent's rationality. I will examine the concept of equilibria when agents have this and other limitations, and I will extend the rational and convergent learning algorithms to the situation of limited agents. I will also investigate relaxing these algorithms to handle issues of observability (e.g. unobservability of other agents' actions or rewards, partially observable state). Finally, I propose to evaluate these algorithms both theoretically in the stochastic game framework, and empirically in a realistic complex multiagent environment.

# Contents

# 1 Introduction

I am interested in the problem of an agent, either a physical robot or software agent, determining a "successful"[1] course of action. Learning can be a very powerful tool that can greatly simplify the problem of robot programming. It also allows a robot to adapt to unforeseen difficulties in the task. Learning has been studied extensively for single-agent problems, where a robot is acting alone in an unchanging environment. With robots and agent technologies becoming more common, domains involving multiple interacting agents arise. In multiagent environments, learning is both more important and more difficult.

In multiagent domains, the agents[2] are forced to interact with other agents, which may have independent goals, assumptions, algorithms, and conventions. In order for robots to handle these environments they must employ some ability to learn and adapt to the other agents' behavior. Since the other agents are also adapting this presents a very difficult learning problem which violates the basic stationary assumption of traditional techniques for behavior learning. Even defining a desirable policy for the agent is difficult, since it depends on the policies of the other agents, which are also being improved through experience. Learning to act in multiagent environments has already been the focus of much research (Sen, 1996; Stone & Veloso, 2000; Weiß & Sen, 1996).

If the other agents are known to be rational and are learning to maximize their rewards, then we can consider the game theoretic concept of equilibria. Equilibria are sets of policies for the players where no player receives higher reward for deviating from their policy. This means equilibria in some sense are self-enforcing. Unfortunately, some games have multiple equilibria and therefore it is important that all the agents play the same equilibrium for it to have its enforcing effects. In addition, the fully rational assumption is not always practical.

Realistically, agents (and humans) are not always capable of acting optimally (**?**). They may have physical limitations (e.g. broken actuator or communication link) that make them incapable of behaving in certain ways. They also may be using learning techniques that sacrifice optimality in order to learn more quickly, such as function approximation and subproblem reuse (Baird & Moore, 1999; Boyan & Moore, 1995; Gordon, 1999; Uther & Veloso, 1998). They may not even be learning at all. These sorts of limitations are real in large, complex environments, and create scenarios where the other agents involved may not appear to be acting rationally. Multiagent learning techniques that are practical must address this possibility. My research will develop practical techniques for learning in the presence of other agents with realistic limitations.

## 1.1 Approach

There are two major steps towards achieving this goal. The first is to develop learning techniques that might be capable of learning in these realistic multiagent settings. The second is to define what such techniques would be expected to learn. In multiagent domains, it is often difficult to define an optimal solution.

**Learning Techniques.** Previous approaches generally fall into two categories. They either apply naive single-agent learning algorithms or attempt to learn an equilibrium directly. Single-agent learning techniques ignore the existence of other agents or assume that their behavior is unchanging. Although they will adapt to maximize their reward given the play of the other agents, if the other agents are learning then the agent's behavior may not converge. In terms that will be formalized in Section 3, they are rational but not convergent.

Equilibrium learning techniques will always converge to an equilibrium policy independently of the behavior of the other agents. They assume the other agents are completely rational, seeking to maximize what is believed to be their reward. These techniques suffer from a number of problems. If there are multiple equilibria, observation of the other agents is required in order for the agents to agree on an equilibrium. In addition if the agents have limitations, converging to a policy independent of their behavior will not allow the agent to take advantage of (or compensate for) the other agents (**?**). They do not always maximize their reward. Hence, these techniques converge, but are not rational.

As a first step towards practical multiagent learning, I propose to develop learning techniques that are both rational and convergent. In Section 3.2, I present an algorithm that moves towards achieving this goal. The algorithm has

[1] Formal definitions for notions like "successful", "optimal", and "rational", will be given in Sections 2 and 3.

[2] Since this work focuses on the multiagent environments using the framework of stochastic games we will use the terms robot, agent, player, or even opponent interchangably. I expect these techniques to be applicable to both software and physical agents, and for a variety of competitive and non-competitive domains.

1. Each player simultaneously selects an action from the set $\{Rock, Paper, Scissors\}$.
2. If both players select the same action, they tie.
3. Otherwise, the winner is determined by the following rules:
   - *Rock* always beats *Scissors*,
   - *Paper* always beats *Rock*,
   - *Scissors* always beats *Paper*.

Table 1: The rules for the game Rock-Paper-Scissors, which is a two-player, single-state game.

properties of a single agent learner, in that it constantly seeks to maximize reward. It also contains a mechanism that encourages convergence.

**Equilibria Under Limitations.** One of the difficulties facing learning in multiagent settings is defining the goal of the algorithm. This difficulty can be seen when considering the children's game, Rock-Paper-Scissors. The rules for this two-player, single-state game are shown in Table 1. In multiagent environments, as demonstrated by this simple game, the reward maximizing (or win maximizing) strategy depends on the strategy of the other agents. In this case, if the other agent were to always select the action *Rock* and were not capable of adapting its strategy, the optimal response is to always play *Paper*. On ther other hand, if the agent were capable of learning and playing any strategy then always playing *Paper* would not be the optimal, or even a good strategy. Instead, randomizing between all three actions would be the best stationary strategy. This strategy is called the game's equlbrium solution, which will be explained in detail in Section 2.1.

One approach is equilibrium learning which assumes the other agent is fully capable and rational and so just learns the game's equilibrium strategy. But if we're interested in learning in situations where other agents may be limited in some fashion we must consider a different approach.

My approach is to assume that the other agents may have limitations but are *otherwise rational*. That is, apart from well defined (but not publicly known) limitations on their capable behavior or learning abilities, they seek to maximize their reward. With this assumption there may exist an *implicit* game with possibly different actions, rewards, or transitions, under which the other agent is acting rationally. This implicit game incorporates the agent's limitations into the game structure. If such an implicit game exists then we can define our desired solution for our multiagent learning algorithm as one that learns an equilibrium in the implicit game.

**Learning with Limited Agents.** I plan to apply the developed rational and convergent learning techniques to successfully learn equilibria in the implicit game when other agents are limited. I also plan to examine further issues of observability to make these algorithms more practical. Observability or knowledge of other agents actions or reward functions may not be possible, but handling these situtations is important for making these techniques practical. Finally, I plan to examine this work in a complex multiagent environment in the presence of other agents with realistic limitations.

## 1.2 Expected Contributions

- Multiagent learning algorithms that are rational and convergent.
- Theoretical examination of the existence of equilibria in stochastic games when other players have limitations.
- Investigation of extensions of rational and convergent learning algorithms to address limited agents.
- Investigation of relaxations of these algorithms to handle partial observability (e.g. opponent's rewards are unknown and unobservable, opponent's actions are unobservable, POMDPs.)

- Examination of knowledge transfer when a task is identical but the participating agents (and their associated limitations) change.

# 2 Background and Related Work

This section presents stochastic games as a model for mulitagent environments. Section 2.1 introduces the stochastic game framework. Sections 2.2 and 2.3 discuss existing techniques for solving these problems. Finally, Section 2.4 presents some other related work on multiagent learning.

## 2.1 Stochastic Game Framework

Before presenting the formal definition of a stochastic game we examine other related models. We begin by examining MDPs – a single-agent, multiple state framework. We then examine matrix games – a multiple-agent, single state framework. Finally we introduce the stochastic game framework, which can be seen as the merging of MDPs and matrix games.

### 2.1.1 Markov Decision Processes

A *Markov decision process* is a tuple, $(\mathcal{S}, \mathcal{A}, T, R)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $T$ is a transition function $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, and $R$ is a reward function $\mathcal{S} \times \mathcal{A} \rightarrow \Re$. The transition function defines a probability distribution over next states as a function of the current state and the agent's action. The reward function defines the reward received when selecting an action from the given state. Solving MDPs consists of finding a policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, which determines the agent's actions so as to maximize discounted future reward, with discount factor $\gamma$.

MDPs are the focus of much of the reinforcement learning work (Kaelbling et al., 1996; Sutton & Barto, 1998). The crucial result that forms the basis for this work is the existence of a stationary and deterministic policy that is optimal. It is such a policy that is the target for RL algorithms.

### 2.1.2 Matrix Games

A *matrix game* or *strategic game* (Osborne & Rubinstein, 1994; Owen, 1995) is a tuple $(n, \mathcal{A}_{1\ldots n}, R_{1\ldots n})$, where $n$ is the number of players, $\mathcal{A}_i$ is the set of actions available to player $i$ (and $\mathcal{A}$ is the joint action space $\mathcal{A}_1 \times \ldots \times \mathcal{A}_n$), and $R_i$ is player $i$'s payoff function $\mathcal{A} \rightarrow \Re$. The players select actions from their available set with the goal to maximize their payoff which depends on *all* the players' actions. These are often called matrix games, since the $R_i$ functions can be written as $n$-dimensional matrices.

Unlike MDPs, it is difficult to define what it means to "solve" a matrix game. A stationary strategy can only be evaluated if the other players' strategies are known. This can be illustrated in the two-player matching pennies game. Here each player may select either *Heads* or *Tails*. If the choices are the same then Player 1 takes a dollar from Player 2. If they are different then Player 1 gives a dollar to Player 2. The matrices for this game are shown in Figure 1(a). If Player 2 is going to play *Heads*, then Player 1's optimal strategy is to play *Heads*, but if Player 2 is going to play *Tails*, then Player 1 should also play *Tails*. So there is no optimal pure strategy, independent of the opponent.

$$R_1 = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \qquad R_2 = -R_1$$

(a) Matching Pennies

$$R_1 = \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \qquad R_2 = -R_1$$

(b) Rock-Paper-Scissors

$$R_1 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \qquad R_2 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$$

(c) Coordination Game

Figure 1: Example matrix games. Games (a) and (b) are zero-sum games, and (c) is a general-sum game.

Players can also play *mixed strategies*, which select actions according to a probability distribution. In the matching pennies game there is also no optimal mixed strategy that is independent of the opponent. We can, though, define an *opponent-dependent* solution or set of solutions called a best-response:

**Definition 1** *For a game, define the* best-response function *for player $i$, $\mathrm{BR}_i(\sigma_{-i})$, to be the set of all, possibly mixed, strategies that are optimal given the other player(s) play the possibly mixed joint strategy $\sigma_{-i}$.*

The major advancement that has driven much of the development of matrix games and game theory is the notion of a best-response equilibrium, or *Nash equilibrium* (Nash, Jr., 1950):

**Definition 2** *A* Nash equilibrium *is a collection of strategies (possibly mixed) for all players, $\sigma_i$, with,*

$$\sigma_i \in \mathrm{BR}_i(\sigma_{-i}).$$

*So, no player can do better by changing strategies given that the other players continue to follow the equilibrium strategy.*

What makes the notion of equilibrium compelling is that all matrix games have a Nash equilibrium, although there may be more than one.

**Types of Matrix Games.** Matrix games can be usefully classified according to the structure of their payoff functions. Two common classes of games are *purely collaborative* and *purely competitive* games. In purely collaborative games, all agents have the same payoff function, so an action in the best interest of one agent is in the best interest of all the agents.

In purely competitive games, there are two agents, where one's payoff function is the negative of the other (i.e. $R_1 = -R_2$). The game in Figure 1(a, b) are examples of such a game. Purely competitive games are also called *zero-sum games* since the payoff functions sum to zero. Other games, including purely collaborative games, are called *general-sum games*. Figure 1(c) is an example of such a game. One appealing feature of zero-sum games is that they contain a unique Nash equilibrium.[3] This equilibrium can be found as the solution to a relatively simple linear program.[4] Finding equilibria in general-sum games requires a more difficult quadratic programming solution (Mangasarian & Stone, 1964).

In the algorithms presented in Section 2.2 we use the functions $\mathrm{Value}_i(\mathrm{MG})$ and $\mathrm{Solve}_i(\mathrm{MG})$ to refer to algorithms for solving matrix games, either linear programming or quadratic programming depending on the context. $\mathrm{Value}_i$ returns player $i$'s expected value of playing the matrix game's equilibrium and $\mathrm{Solve}_i$ returns player $i$'s equilibrium strategy.

### 2.1.3 Stochastic Games

A *stochastic game* is a tuple $(n, \mathcal{S}, \mathcal{A}_{1\ldots n}, T, R_{1\ldots n})$, where $n$ is the number of agents, $\mathcal{S}$ is the set of states, $\mathcal{A}_i$ is the set of actions available to agent $i$ (and $\mathcal{A}$ is the joint action space $\mathcal{A}_1 \times \ldots \times \mathcal{A}_n$), $T$ is the transition function $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$, and $R_i$ is the reward function for the $i$th agent $\mathcal{S} \times \mathcal{A} \to \Re$. This looks very similar to the MDP framework except there are multiple agents selecting actions and the next state and rewards depend on the joint action of those agents. It's also important to notice that each agent has its own separate reward function. We are interested in determining a course of action for an agent in this environment. Specifically we want to learn a stationary though possibly stochastic policy, $\pi : \mathcal{S} \to PD(\mathcal{A}_i)$, that maps states to a probability distribution over its actions. The goal is to find such a policy that maximizes the agent's discounted future reward with discount factor $\gamma$.

SGs are a very natural extension of MDPs to multiple agents. They are also an extension of matrix games to multiple states. Each state in a SG can be viewed as a matrix game with the payoffs for each joint action determined by $R_i(s, a)$. After playing the matrix game and receiving the payoffs the players are transitioned to another state (or matrix game) determined by their joint action. We can see that SGs then contain both MDPs and matrix games as subsets of the framework.

A non-trivial result, proven by Shapley (Shapley, 1953) for zero-sum games and by Fink (Fink, 1964) for general-sum games, is that there exist equilibria solutions for stochastic games just as they do for matrix games.

**Types of Stochastic Games.** The same classification for matrix games can be used with stochastic games. Purely collaborative games are ones where all the agents have the same reward function. Purely competitive, or zero-sum, games are two-player games where one player's reward is always the negative of the other's. Like matrix games, zero-sum stochastic games have a unique Nash equilibrium, although finding this equilibrium is not so easy. Section 2.2 presents two algorithms from game theory for finding this equilibrium.

[3] There can actually be multiple equilibria, but they will all have equal payoffs and are interchangeable (Osborne & Rubinstein, 1994).

[4] The value of the equilibrium is the solution to the following linear program: $\max_{\sigma_1 \in PD(\mathcal{A}_1)} \min_{a_2 \in \mathcal{A}_2} \sum_{a_1 \in \mathcal{A}_1} R_1(s, \langle a_1, a_2 \rangle)\sigma_{a_2}$.

1. Initialize $V$ arbitrarily.
2. Repeat,
   (a) For each state, $s \in \mathcal{S}$, compute the matrix,
   $$G_s(V) = \left[ g_{a \in \mathcal{A}} : \quad R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s,a,s')V(s') \quad \right].$$
   (b) For each state, $s \in \mathcal{S}$, update $V$,
   $$V(s) \leftarrow \mathrm{Value}_1 \left[ G_s(V) \right].$$

Table 2: Algorithm: Shapley.

## 2.2 Static Solutions

This section presents a number of algorithms for "solving" stochastic games. As is apparent from the matrix game examples there is not likely to be an optimal solution to a stochastic game that is independent of the other agents. This makes solving SGs for a single agent difficult to define. The existence of equilibria does not alleviate our problem since they are only compelling if *all* the agents participate in playing equilibrium. There also may be multiple equilibria, so which equilibria to play is dependent on the other agents. This problem is central to my proposed work and will be discussed throughout this paper.

The algorithms presented in this section make the assumption that the stochastic game contains a unique equilibrium. All but one of the algorithms require the even stronger assumption that the game is zero-sum. The goal of these algorithms, then, is to learn this equilibrium solution.

Since stochastic games were first studied in game theory, we first present two algorithms developed in this community. These algorithms assume a model of the world is available, that is the transition function $T$ and reward functions $R_i$ are known. They also focus more on learning the value, $V^*(s \in \mathcal{S})$ of the game rather than on the policy of any particular agent. We also present two relatively recent algorithms from the reinforcement learning community. These algorithms don't make use of a world model, and only receive observations of the functions $T$ and $R_i$ as the agents act in the environment. They also tend to focus on learning the equilibrium policy for a single agent.

### 2.2.1 Shapley

The very first algorithm for solving a SG was given by Shapley (1953) as a result of his proof of the existence of equilibria in zero-sum SGs. The algorithm is shown in Table 2. The algorithm uses a temporal differencing technique to backup values of next states into a simple matrix game, $G_s(V)$. The value function $V$ is then updated by solving the matrix game at each state.

Notice the algorithm is nearly identical to value iteration for MDPs, with the "$\max$" operator replaced by the "$\mathrm{Value}$" operator. The algorithm also shows that equilibria in stochastic games are solutions to Bellman-like equations. The algorithm's value function $V$ can be shown to converge to $V^*$ which satisfies the following equations,

$$\forall s \in \mathcal{S} \qquad V^*(s) = \mathrm{Value}_1 \left( [G_s(V^*)] \right).$$

The equilibrium policies can be found from the equilibrium value function: $\pi_i^* = \mathrm{Solve}_i \left( [G_s(V^*)] \right)$.

### 2.2.2 Pollatschek and Avi-Itzhak

Just as Shapley's algorithm is an extension of value iteration to stochastic games, Pollatschek and Avi-Itzhak (Vrieze, 1987) introduced an extension of policy iteration (Howard, 1960). The algorithm is shown in Table 3. Each player selects the equilibrium policy according to the current value function, making use of the same temporal differencing matrix, $G_s(V)$, as in Shapley's algorithm. The value function is then updated based on the actual rewards of following these policies.

1. Initialize $V$ arbitrarily.
2. Repeat,

$$\begin{aligned} \rho_i &\leftarrow \text{Solve}_i\left[G_s(V)\right] \\ V(s) &\leftarrow E\left\{\sum \gamma^t r_t | s_0 = s, \rho_i\right\}. \end{aligned}$$

Table 3: Algorithm: Pollatschek and Avi-Itzhak. The function $G_s$ is the same as presented in Table 2.

1. Initialize $Q(s \in \mathcal{S}, a \in \mathcal{A})$ arbitrarily, and set $\alpha$ to be the learning rate.
2. Repeat,
   (a) From state $s$ select action $a_i$ that solves the matrix game $\left[\, Q(s,a)_{a \in \mathcal{A}} \,\right]$, with some exploration.
   (b) Observing joint-action $a$, reward $r$, and next state $s'$,

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma V(s')),$$

   where,

$$V(s) = \text{Value}\left(\left[\, Q(s,a)_{a \in \mathcal{A}} \,\right]\right).$$

Table 4: Algorithm: Minimax-Q and Hu & Wellman. The difference between the algorithms is in the $\text{Value}$ function and the $Q$ values. Minimax-Q uses the linear programming solution for zeros-sum games and Hu & Wellman uses the quadratic programming solution for general-sum games. Also, the $Q$ values in Hu & Wellman are actually a vector of expected rewards, one entry for each player.

Like Shapley's algorithm, this algorithm also computes the equilibrium value function, from which can be derived the equilibrium policies. This algorithm, though, is only guaranteed to converge if the transition function, $T$, and discount factor, $\gamma$, satisfies a certain property. A similar algorithm by Hoffman and Karp (Vrieze, 1987) which alternates the policy learning (rather than it occurring simultaneously) avoids the convergence problem, but requires obvious control over when the other agents in the environment are learning.

#### 2.2.3 Minimax-Q

Littman (1994) was the first to examine stochastic games as a framework for multiagent reinforement learning. He extended the traditional Q-Learning algorithm for MDPs to zero-sum stochastic games. The algorithm is shown in Table 4. The notion of a $Q$ function is extended to maintain the value of *joint* actions, and the backup operation computes the value of states differently, by replacing the $\max$ operator with the $\text{Value}$ operator. It is interesting to note that this is basically the off-policy reinforcement learning equivalent of Shapley's "value iteration" algorithm.

This algorithm does in fact converge to the stochastic game's equilibrium solution, assuming the other agent executes all of its actions infinitely often. This is true even if the other agent does not converge to the equilibrium, and so provides an *opponent-independent* method for learning an equilibrium solution.

#### 2.2.4 Hu and Wellman

Hu and Wellman (1998) extended the Minimax-Q algorithm to general-sum games. The algorithm is structurally identical and so is also shown in Table 4. The extension requires that for each state/joint-action pair an $n$-vector of

Matrix Game Solver + Temporal Differencing = Stochastic Game Solver

| MG | + | TD | = | Game Theory | RL |
|---|---|---|---|---|---|
| LP | | TD(0) | | Shapley | MiniMax-Q |
| LP | | TD(1) | | Pollatschek and Avi-Itzhak | – |
| LP | | TD($\lambda$) | | Van der Wal(Vrieze, 1987) | – |
| QP | | TD(0) | | – | Hu and Wellman |
| FP | | TD(0) | | Fictitious Play | Opponent-Modeling / JALs |

LP: linear programming QP: quadratic programming FP: ficitious play

Table 5: Summary of algorithms to solve stochastic games. Each contains a matrix game solving component and a temporal differencing component. "Game Theory" algorithms assume the transition and reward function is known. "RL" algorithms only receive observations of the transition and reward function.

values are maintained, one for each agent. This means,

$$i = 1 \ldots n \qquad \left[ Q^i(s, a)_{a \in \mathcal{A}} \right],$$

is actually a general-sum matrix game. So, the linear programming solution to compute $\mathrm{Value}$ of the matrix game is replaced by the quadratic programming solution.

This algorithm is the first to address the complex problem of general-sum games. But the algorithm requires a number of very limiting assumptions. The most restrictive of which limits the structure of all the intermediate matrix games faced while learning (i.e. $Q(s, a)$.) The largest difficulty is that it is impossible to predict whether this assumption will remain satisfied while learning. The other assumption to note is that the game must have a unique equilibrium, which is not always true of general-sum stochastic games. This is necessary since the algorithm strives for the opponent-independence property of Minimax-Q, which allows the algorithm to converge almost regardless of the other agent's actions. With multiple equilibria it's important for all the agents to play the same equilibrium in order for it to have its reinforcing properties. So, learning independently is not possible. Further examination of the convergence problems in their algorithm are discussed in Section 3.1.

### 2.2.5 Observations

A number of observations can be drawn from these algorithms, especially by examining the structure of the algorithms. Notice, they all have a matrix game solving component and a temporal differencing component. A summary of the components making up these algorithms can be seen in Table 5.

The first thing to note is the lack of an RL algorithm with a multiple step backup (i.e. policy iteration or TD($\lambda$).) One complication for multi-step backup techniques is that they are *on-policy*, that is they require that the agent be executing the policy it is learning. In a multiagent environment this would require that the other agents also be learning *on-policy*. Despite this limitation it is still very suggestive of new algorithms that may be worth investigating. Also, there has been work to make an off-policy TD($\lambda$) (Peng & Williams, 1996; Watkins, 1989) that may be useful in overcoming this limitation.

A second observation is that proving convergence to an equilibrium solution is not a trivial thing. For convergence almost all the algorithms assume a zero-sum game, and the Pollatschek and Avi-Itzhak algorithm requires stringent assumptions on the game's transition function. The only other algorithm, Hu and Wellman, although doesn't assume zero-sum, still requires that the game have a unique equilibrium along with additional assumptions about the payoffs.

The final observation is that all of these algorithms use a closed-form solution for matrix games (i.e. linear programming or quadratic programming.) Although this provides what appears to be opponent-independent algorithms for finding equilibria, it automatically rules out stochastic games with multiple equilibria. If a game has multiple equilibria, the "optimal" policy must depend on the policies of the other agents. Values of states therefore depend on

1. Initialize $U(s \in \mathcal{S}, a_i \in \mathcal{A}_i) \leftarrow 0$, and $C(s \in \mathcal{S}) \leftarrow 0$.
2. Repeat,
   (a) From state $s$ select action $a_i$ that maximizes $U(s, a_i)$.
   (b) Observing other agents' actions $a_{-i}$, reward $r$, and next state $s'$,

$$\begin{aligned} \forall a_i \in \mathcal{A}_i \quad U(s, a_i) &\leftarrow U(s, a_i) + G_s(V)\langle a_i, a_{-i}\rangle \\ C(s) &\leftarrow C(s) + 1 \end{aligned}$$

   where,

$$V(s) = \max_{a_i} \frac{U(s, a_i)}{C(s)}.$$

Table 6: Algorithm: Fictitious play for player $i$ in two-player, zero-sum stochastic games using a model.

the other agents policies, and static opponent-independent algorithms simply will not work. In the next section we examine algorithms that do not use a static matrix game solver.

## 2.3 Learning Solutions

We now examine algorithms that use some form of *learning* as the matrix game solving component. There are two reasons that one would want to consider a matrix game learner rather than a closed form solution. The first was mentioned in the previous section that selecting between multiple equilibria requires observing and adapting to the other players' behavior. It is important to point out that observing other players' behavior cannot be a step that follows a static learning algorithm. Equilibrium selection affects both the agent's policy at a state and that state's value. The value of a state in turn affects the equilibria of states that transition into it. In fact the number of equilibrium solutions in stochastic games can grow exponentially with the number of states, and so equilibrium selection after learning is not feasible.

The second reason is that using an opponent-dependent matrix game solver, allows us to examine the problem of behaving when the other agents have limitations, the focus of my proposed work. These limitations might prevent the agent from playing certain equilibria strategies, which the static algorithms would not be able to exploit. Handling these situations are important since solving large problems often requires generalization and approximation techniques that impose learning limitations.

Here we present two similar algorithms, one from game theory and the other from reinforcement learning, that depend on the other players' behavior. Both of these algorithms are only capable of playing deterministic policies, and therefore can only converge to pure strategy equilibria. Despite this fact, they still have interesting properties for zero-sum games that have a mixed policy equilibrium.

### 2.3.1 Fictitious Play

Fictitious play (Robinson, 1951; Vrieze, 1987) assumes opponents play stationary strategies. The basic game theory algorithm is shown in Table 6. The algorithm maintains information about the average value of each action (i.e. $U(s,a)/C(s)$ is the average expected discounted reward from past experience.). The algorithm then deterministically selects the action that has done the best in the past. This is nearly identical to single agent value iteration with a uniform weighting of past experience. Like value iteration the algorithm is deterministic and cannot play a mixed strategy. Despite this limitation, when all players are playing fictitious play, they will converge to a Nash equilibrium in games that are iterated dominance solvable (Fudenberg & Levine, 1999) or fully collaborative (Claus & Boutilier, 1998). In addition, in zero-sum games the players' empirical distribution of actions selected converges to the game's equilibrium mixed strategy (Robinson, 1951; Vrieze, 1987).

1. Initialize $Q$ arbitrarily, and $\forall s \in \mathcal{S} \quad C(s) \leftarrow 0$ and $n(s) \leftarrow 0$.
2. Repeat,
   (a) From state $s$ select action $a_i$ that maximizes,
   $$\sum_{a_{-i}} \frac{C(s, a_{-i})}{n(s)} Q(s, \langle a_i, a_{-i} \rangle)$$
   (b) Observing other agents' actions $a_{-i}$, reward $r$, and next state $s'$,
   $$\begin{aligned} Q(s,a) &\leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma V(s')) \\ C(s, a_{-i}) &\leftarrow C(s, a_{-i}) + 1 \\ n(s) &\leftarrow n(s) + 1 \end{aligned}$$
   where,
   $$\begin{aligned} a &= (a_i, a_{-i}) \\ V(s) &= \max_{a_i} \sum_{a_{-i}} \frac{C(s, a_{-i})}{n(s)} Q(s, \langle a_i, a_{-i} \rangle). \end{aligned}$$

Table 7: Algorithm: Opponent Modeling Q-Learning for player $i$.

The algorithm has a number of advantages. It is capable of finding equilibria in both zero-sum games and some classes of general-sum games. It also finds an equilibrium without keeping track of the values or rewards of the other players. It suffers from the problem that for zero-sum games, it can only *find* an equilibrium strategy, without actually *playing* according to that strategy. Section 2.4 discusses an extension to address this drawback.

#### 2.3.2 Opponent Modeling

Opponent modeling (Uther & Veloso, 1997) or joint action learners (JALs) (Claus & Boutilier, 1998) are RL algorithms that are similar to fictitious play. The algorithm is shown in Table 7. Explicit models of the opponents are learned as stationary distributions over their actions (i.e. $C(s, a_{-i})/n(s)$ is the probability the other players will select joint action $a_{-i}$ based on past experience.) These distributions combined with learned joint-action values from standard temporal differencing are then used to select an action. Uther and Veloso (1997) investigated this algorithm in the context of a fully competitive domain, and Claus and Boutilier (1998) examined it for fully collaborative domains.

The algorithm has very similar behavior to fictitious play. It does require observations of the opponents' actions, but not of their individual rewards. Like fictitious play, it's empirical distribution of play may converge to an equilibrium solution, but its action selection is deterministic and cannot play a mixed strategy.

### 2.4 Other Related Work

Other learning techniques have focused on goals other than explicit learning of a game's equilibrium. All of the techniques below have only been investigated in single-state matrix games. It is not clear how they might be applied or scale to games with multiple states. But they do provide insights into other methods for adaptation in multiagent systems.

**Smooth Fictitious Play.** Smooth fictitious play (Fudenberg & Levine, 1999) is an extension to fictitious play to allow it to play mixed strategies. In a matrix game, fictitious play always plays a best-response to the past empirical play of the other players. Smooth fictitious play instead uses a smooth approximation to the best-response function. This not only allows it to play mixed strategies but also, under certain approximations, has the property of *$\epsilon$-universal consistency*. A learning algorithm satisfies this property if, regardless of the play of the other players, the player receives an average payoff that is within $\epsilon$ of the best average payoff that could've been received by a stationary strategy.

**Regret Bounds.** Total regret is the difference between the total reward received by the learner and the total reward of the best action in hindsight and are often given as asymptotic bounds. This is very similar to $\epsilon$-universal consistency, and in fact an algorithm that is $\epsilon$-universally consistent has a regret that grows $O(T)$, where $T$ is the number of rounds the game is played. Auer Et. Al. (Auer et al., ) developed an algorithm, Exp3[5], that is similar to smooth fictitious play, but with regret that grows $O(\sqrt{T})$. This means that the average payoff actually converges to that of the best action in hindsight, which is no less than the equilibrium value of the game. In addition, it does not require that the game be known in advance, and so fits within the traditional reinforcement learning paradigm.

**Evolutionary Game Theory.** Evolutionary game theory (see Weibull (Weibull, 1995) for a summary) has also examined techniques for matrix games that do not explicitly learn equilibria. These techniques represent a strategy as a "population of players" where each player plays one of the actions deterministically. Strategies learn or evolve by having the subsets of the population grow in proportion to the expected payoff of their action. This allows the changes in the population to be modeled mathematically and the resulting system dynamics to be examined. This work has focused on examining the dynamics of plausible models for change in biological populations. It has not been used to help derive desirable learning rules, and also hasn't been examined in the context of stochastic games.

**Limitations as Finite Automaton.** Carmel and Markovitch (Carmel & Markovitch, 1996; Carmel, 1997) examined a model-based approach to multiagent learning. They introduced algorithms for efficiently learning a model of the opponent as a deterministic finite automaton (DFA). Using this learned model, they then can compute the optimal best response. They examined this in the situation of repeated matrix games and showed that their technique can learn effectively against a variety of of DFA opponents. They also showed that their algorithm learned much more quickly (i.e. was more data efficient) than Q-learning, a model-free reinforcement learner. It's unknown how this compares with a model-based reinforcement learner such as prioritized sweeping (Moore & Atkeson, 1993).

This technique, though, does not entirely address the problem of concurrent learning in a multiagent environment. Although a learning rule can be be modeled by a DFA, the number of states that a rule requires may be huge. For example, Q-learning with a decaying learning rate would be a completely intractable DFA. Also, the model-based learning agent is required to be strictly more powerful than it's opponent, since it's learning rule both learns the opponent's DFA and also generates its own. It's unclear wether any sort of equilibrium would be reached or even exists if two model-based DFA learners played eachother. It's also not certain how DFA modelling scales to stochastic, multi-state games.

[5] Exp3 stands for "Exponential-weight algorithm for Exploration and Exploitation."

# 3 Previous Work

This section discusses our previous work related to the problem of multiagent learning. Section 3.1 presents problems in the convergence of the Hu & Wellman algorithm, and helps to illustrate why multiagent learning is difficult (Bowling, 2000). Section 3.2 presents a new learning algorithm that is both rational and can converge to mixed equilibria in stochastic games. The algoritm is the first known RL algorithm to achieve this and is the basis for much of my proposed work discussed in Section 4. Finally, Section 3.3 presents some work on using solutions to subproblems to speed learning. This work can be seen as an example of a practical limitation that might be encountered in a multiagent learning setting.

## 3.1 Convergence Problems in General-Sum Games

Hu & Wellman (1998) introduced a Q-learning method for solving general-sum stochastic games (see Table 4). Their algorithm is an extension of Minimax-Q, and replaces the computation of the value of a zero-sum game with the computation of the value of a general-sum game. The algorithm strives to maintain the same property of Minimax-Q that it can learn equilibria independent of the actions selected by the other players.

The algorithm, like Minimax-Q, maintains $Q$ values for every state/joint-action pair. Since the agents' rewards in general-sum games can be completely independent, multiple $Q$ values (one for each agent) must be maintained for each state/joint-action pair. The entry $Q^i(s,a)$ approximates the expected discounted reward for player $i$ if the players select joint action $a$ from state $s$ and then follow the same Nash equilibrium of the stochastic game. On an observation, $\langle s, a, s', r \rangle$ where $r$ is an $n$-vector of each of the agents' rewards, the update is,

$$Q^i(s,a) \leftarrow (1-\alpha)Q^i(s,a) + \alpha(r^i + \gamma V^i(s')),$$

where,

$$V^i(s') = \mathrm{Value}_i\, Q(s')$$

For ease of notation we use $Q(s)$ to represent the general-sum matrix game that is represented by the $n$ matrices whose entries correspond to $Q^i(s, a \in \mathcal{A})$. So in the update rule the $\mathrm{Value}_i$ operation computes the equilibrium value to player $i$ of the general-sum matrix game defined by the $Q$ values at the state $s'$[6]. Since the update operation makes use of the equilibrium value of a state, this is only well-defined if there is a single equilibrium (or multiple equilibria with the same value.) It is assumed for their algorithm and theoretical results that there is in fact a unique equilibrium.

Their main result is the theorem below. The theorem claims convergence of the algorithm for two-player, general-sum games. In the theorem, $r_t^k$ refers to the immediate reward received by player $k$ at time $t$, and $\pi^1(s)Q^k(s)\pi^2(s)$ refers to player $k$'s expected value of playing the matrix game $Q(s)$ where the players select actions according to the strategies, $\left(\pi^1(s), \pi^2(s)\right)$.

**Theorem 1** *(Hu & Wellman, 1998) Under assumptions below, let the sequences* $\left(Q_t^1, Q_t^2\right)$ *be updated by,*

$$\begin{aligned} Q_{t+1}^k(s, \langle a^1, a^2 \rangle) \quad = \quad & (1-\alpha_t) Q_t^k(s, \langle a^1, a^2 \rangle) + \\ & \alpha_t \left( r_t^k + \gamma \pi^1(s') Q_t^k(s') \pi^2(s') \right), \end{aligned}$$

*where* $\left(\pi^1(s), \pi^2(s)\right)$ *is a mixed strategy Nash equilibrium for the matrix game* $\left(Q_t^1(s'), Q_t^2(s')\right)$*. Then these sequences converge to the Nash equilibrium Q values* $\left(Q_*^1, Q_*^2\right)$ *defined by,*

$$Q_*^k(s, \langle a^1, a^2 \rangle) = r_t^k + \gamma \pi_*^1(s') Q_t^k(s') \pi_*^2(s'),$$

*where* $\left(\pi_*^1(s), \pi_*^2(s)\right)$ *is a Nash equilibrium for the stochastic game.*

The theorem requires three assumptions. Two of these assumptions deal with exploration and the proper decay of the learning rate. These are the standard assumptions of Q-learning and are not provided in this paper. The final assumption deals with the nature of the matrix games that are faced while learning.

[6] Finding the value of a general-sum game requires an involved quadratic programming solution (Filar & Vrieze, 1997).

**Assumption 1** *(Hu & Wellman, 1998) A Nash equilibrium* $\left(\pi^1(s), \pi^2(s)\right)$ *for any matrix game* $\left(Q_t^1(s), Q_t^2(s)\right)$ *satisfies one of the following properties:*

1. *The equilibrium is a global optimal.*

$$\forall \rho^k \quad \pi^1(s) Q^k(s) \pi^2(s) \geq \rho^1(s) Q^k(s) \rho^2(s)$$

2. *The equilibrium receives a higher payoff if the other agent deviates from the equilibrium strategy.*

$$\begin{array}{ll} \forall \rho^k & \pi^1(s) Q^1(s) \pi^2(s) \leq \pi^1(s) Q^1(s) \rho^2(s) \\ & \pi^1(s) Q^2(s) \pi^2(s) \leq \rho^1(s) Q^2(s) \pi^2(s) \end{array}$$

It is interesting to note what matrix games the properties encompass. The first property states that there's a set of strategies for the players where each player individually receives its maximum possible payoff. This also ensures that such a set of strategies is an equilibrium, since no player could benefit from deviating from the strategy. The second property states that the Nash equilibrium for the game is a "saddle point". This means that not only does a player *not benefit* from deviating from the equilibrium, but also that all other players *do benefit* if the player deviates. Notice that all zero-sum games satisfy this second property.

The proof of their theorem makes use of a crucial lemma claiming their update function is a contraction mapping. Let $P_t^k$ be an update function defined as,

$$P_t^k Q^k(s) = r_t^k + \gamma \pi^1(s) Q^k(s) \pi^2(s),$$

where $(\pi^1(s), \pi^2(s))$ are a Nash equilibrium for the two-player matrix game where the matrices are $\left(Q^1(s), Q^2(s)\right)$. Notice that this is the same update function as in Theorem 1, but without the stochastic approximation. Their lemma claims that $P_t^k$ satisfies the following property,

$$\forall Q^k \qquad ||P_t^k Q^k - P_t^k Q_*^k|| \leq \gamma ||Q^k - Q_*^k||,$$

where $||\cdot||$ is the max-norm operator over all states and actions. This lemma effectively states that the update function will always move $Q^k$ closer to $Q_*^k$.

### 3.1.1 Counterexample and Flaw

In this section we show a counterexample to their lemma and also the flaw in their proof of it. We follow this by introducing strengthened assumptions under which the lemma is valid.

**Counterexample.** Consider the stochastic game with three states shown in Figure 2. State $s_0$ always transitions to state $s_1$ with rewards 0. State $s_1$ is a $2 \times 2$ game with all actions causing the game to transition to a terminating state, $s_2$.

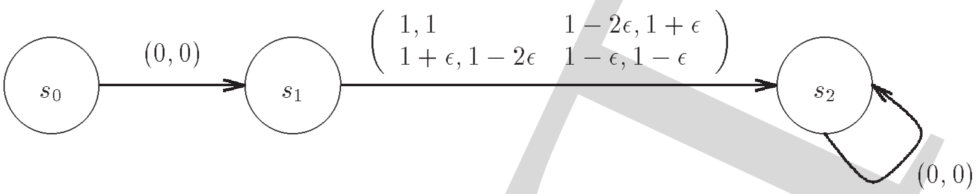
Consider the following $Q$ function,

$$\begin{array}{rcl} Q(s_0) & = & (\gamma, \gamma) \\ Q(s_1) & = & \left( \begin{array}{ll} \boxed{1+\epsilon, 1+\epsilon} & 1-\epsilon, 1 \\ 1, 1-\epsilon & 1-2\epsilon, 1-2\epsilon \end{array} \right) \\ Q(s_2) & = & (0, 0). \end{array}$$

The matrix game corresponding to $Q(s_1)$ has a unique Nash equilibrium where both players choose their first action. Notice that $||Q - Q_*|| = \epsilon$. If $P$ is a contraction mapping then after applying $P$ the values should be closer to $Q^*$. The actual values for $PQ$ are,

$$\begin{array}{rcl} PQ(s_0) & = & (\gamma(1+\epsilon), \gamma(1+\epsilon)) \\ PQ(s_1) & = & \left( \begin{array}{ll} 1, 1 & 1-2\epsilon, 1+\epsilon \\ 1+\epsilon, 1-2\epsilon & \boxed{1-\epsilon, 1-\epsilon} \end{array} \right) \\ PQ(s_2) & = & (0, 0). \end{array}$$

Notice that $||PQ - PQ^*|| = 2\gamma\epsilon$, which is greater than $\epsilon$ if $\gamma > 0.5$. Hence, with the max norm $P$ is not a contraction mapping and the lemma is false.




$$Q^*(s_1) = (\gamma(1-\epsilon), \gamma(1-\epsilon)) \qquad Q^*(s_1) = \begin{pmatrix} 1,1 & 1-2\epsilon, 1+\epsilon \\ 1+\epsilon, 1-2\epsilon & \boxed{1-\epsilon, 1-\epsilon} \end{pmatrix} \qquad Q^*(s_2) = (0,0)$$

Figure 2: A three state stochastic game, including the Nash equilibrium values, $Q^*$. All transitions are deterministic and are independent of the actions selected. The only choice available to the agents is in state $s_1$, where the corresponding matrix game has a unique Nash equilibrium where both players choose their second action.

**Proof Flaw.** The flaw in their proof of the lemma is due to a missing case. It handles the case where $Q_*(s)$ meets property 1 of Assumption 1, and the case where $Q(s)$ meets property 2 of Assumption 1. It fails to address when $Q_*(s)$ meets property 2 and $Q(s)$ meets property 1. This is exactly the case of the counterexample. The matrix games $Q(s_1)$ and $Q_*(s_1)$ both have unique pure strategy equilibriums (as shown by the boxes in their respective matrices.) The Nash equilibrium of $Q(s)$ meets property 1 because both players playing their first action is the best either player can do. The Nash equilibrium of $Q_*(s)$ meets property 2 because it's a "saddle point", that is if a player deviates from the equilibrium the other player gets rewarded. So the counterexample does indeed satisfy their assumptions.

It is important to note that the counterexample and flaw only show that the backup used by the algorithm is not a contraction mapping using the max norm. This does not necessarily disprove Theorem 1, as there may be another norm for which the backup *is* a contraction mapping. On the other hand, it leaves open whether the theorem is actually true and provides evidence to the contrary, which is discussed further in Section 3.1.3.

#### 3.1.2 Strengthened Assumptions

The original assumption on which their proof is based can be strengthened in order to make the crucial lemma true. This is done by simply ruling out the case not handled in their proof and under which the counterexample falls.

**Assumption 2** *The Nash equilibrium of all matrix games, $Q_t(s)$, as well as $Q_*(s)$ must satisfy property 1 in Assumption 1 **or** the Nash equilibrium of all matrix games, $Q_t(s)$, as well as $Q_*(s)$ must satisfy property 2 of Assumption 1.*

This new assumption, although making the original theorem true, places heavy limitations on the applicability of the results. This is examined in the next section.

#### 3.1.3 Discussion

In this section we discuss the applicability of Theorem 1 with the strengthened assumption that it requires. We also examine other issues surrounding the problem of convergence in general-sum stochastic games.

**Applicability of the Theorem.** Under strengthened Assumption 2 the theorem has a number of limitations. One of these limitations was also problematic under the original assumption, but is even more so under Assumption 2.

The first limitation is that if the current $Q_t$ values satisfy the assumption and the $Q_*$ values are known to satisfy the assumption, there's still no guarantee that future $Q_{t+1}$ values will satisfy the assumption. Since no guarantees are made on future values, the theorem cannot actually guarantee convergence. Rather it only guarantees that *if* the $Q$ values converge while always satisfying the assumption, then they've converged to the game's equilibrium. Since Assumption 2 makes a further requirement that the values always satisfy the *same property*, this only magnifies this problem.

The second limitation is due directly to the strengthened assumption. The $Q$ values must always satisfy the same property that the unknown $Q^*$ values satisfy. Although, the $Q$ values can be initialized to satisfy both properties (e.g. by initializing all the values to zero), the initial steps of learning will undoubtedly move the $Q$ values into satisfying only one of the properties. The theorem, then, only guarantees convergence if the unknown $Q^*$ values for the stochastic game satisfies the *exact same* property.

**Convergence in General-Sum Games** One question that arises is why general-sum games are more problematic for convergence of learning algorithms. The counterexample presented in this paper gives some insight into this question. Small changes in the values of joint-actions can cause a large change in the state's Nash equilibria. This can cause drastic changes in the value of that state. Overall, this means small changes in values can propagate into even larger changes in values. This is evident in the counterexample where an $\epsilon$ change in the $Q$ values propagates into a $2\epsilon$ change.

Despite this fact, there are some classes of general-sum stochastic games where convergent learning is not so problematic. In fact, there is a large class where even naive single-agent learners (e.g. standard Q-learning (Watkins, 1989)) can find Nash equilibria. Fully collaborative games, where all the agents have identical reward functions, can be learned by simple Q-learners (Claus & Boutilier, 1998). Games that are iterated dominance solvable, where the process of eliminating universally inferior policies by the players leaves only Nash equilibria policies, are also solvable by naive learning (Fudenberg & Levine, 1999).

In fact even stochastic games that do not fall under these two categories can still be solved with single-agent learners. Consider the two-agent gridworld game depicted in Figure 3, which was introduced as an example domain for the Hu & Wellman algorithm (Hu, 1999). The agents start in two corners and are trying to reach the goal square on the opposite wall. The players have the four compass actions (i.e. N, S, E, and W), which are in most cases deterministic. If the two players attempt to move to the same square, both moves fail. To make the game interesting and force the players to interact, from the initial starting position the North action is uncertain, and is only executed with probability 0.5. The optimal path for each agent, then, is to move laterally on the first move and then move North to the goal, but if both players move laterally then the actions will fail. There are two Nash equilibria for this game. They involve one player taking the lateral move and the other trying to move North.

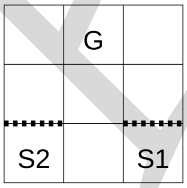



Figure 3: Gridworld game. The dashed walls represent the actions that are uncertain.

| Strategy | Trials | |
|---|---|---|
| W-N | 114 | (57%) |
| N-E | 86 | (43%) |
| Other | 0 | (0%) |
| Total | 200 | (100%) |

Table 8: Q-learning for the gridworld game. The table shows the number of trials where the players converged to a particular policy. "W-N" and "E-N" correspond to the two Nash equilibria where one player moves laterally in initial state while the other tries to move North. "Other" corresponds to all other policies.

Table 8 shows the results of training two agents in the gridworld game, using standard Q-learning. The agents were trained with a decayed learning rate for one million steps to be sure the policies had converged. The table shows how often the agents converged to specific policies in the 200 trials. The important thing to note is that the agents always converged to one of the game's Nash equilibria. So naive single-agent learners can also converge to Nash equilibrium in games that aren't fully-collaborative or iterated dominance solvable. Understanding what characterizes these easier general-sum games may be helpful for isolating the difficulties present in other general-sum games.

## 3.2 Rational and Convergent Learning

In Section 2.2 we presented some algorithms that directly learn to play a game's equilibrium. As we pointed out, this approach will fail to take advantage of the specific situations I'm interested in, namely when the other players have limitations. In Section 2.3 we presented algorithms that are not opponent-independent, but cannot play mixed policies and so in some games will not converge to a stationary policy. In this section we formalize these two properties and then present a new algorithm that moves towards simultaneously achieving them.

### 3.2.1 Motivation

Before presenting our new learning technique, we first introduce two properties that are desirable for a multiagent learner. We also examine how current multiagent RL techniques compare to these criteria.

**Properties.** The following are two desirable properties in a multiagent learning algorithm, where all the players are learning simultaneously.

**Property 1** *(Rational) If the other players' strategies converge to stationary strategies then the player will converge to a strategy that is optimal given their strategies.*

**Property 2** *(Convergent) Given that the other players are following behaviors from a class of behaviors, $\mathcal{B}$, all the players will converge to stationary strategies.*

The first property requires the player to behave optimally when the other players find stationary strategies. The second property requires that for some class of behaviors (i.e. learning algorithms) for the other players (ideally a well-encompassing class) the strategies will indeed converge. In this paper we focus on the convergence of a learning algorithm when playing other agents that are using the same algorithm. This is a crucial yet difficult first step towards examining convergence with more general classes of behaviors. In combination, these two properties guarantee that the player will converge to a stationary strategy that is optimal given the play of the other players.

It is interesting to note that when all players are rational, if they converge they must have converged to a Nash equilibria. Since the other players converge to a stationary policy, each player must converge to a best response to their policies. Since this is true of each player, then this is an equilibrium. If all players satisfy *both properties* with classes that include the other players' behavior, then convergence to a Nash equilibrium is guaranteed.

Some additional justification for the desirability of these properties is given later in Section 3.2.4.

**Other Reinforcement Learners.** There are few RL techniques that directly address learning in a multiagent system. We examine three RL techniques: independent learners (ILs), joint-action learners (JALs), and minimax-Q. ILs are single agent learning algorithms and so capture the existence of other agents as part of the environment (e.g. Q-learning (Watkins, 1989), TD($\lambda$) (Sutton, 1988), prioritized sweeping (Moore & Atkeson, 1993), etc.) This is the most common approach to learning in multiagent systems, even though the MDP assumption of stationary transition probabilities is violated. JALs and minimax-Q are aware of the other players, but account for them differently.

**Independent Learners.** ILs are just the naive application of single agent (MDP) algorithms to multi-agent domains (SGs). They ignore the existence of other agents, assuming their rewards and transitions are Markov and stationary. As a result, these algorithms only learn deterministic policies. This naive approach does satisfy one of the two properties. If the other agents play, or converge to, stationary strategies then their Markovian assumption holds and they converge to an optimal response. So, an IL is rational. On the other hand, it is not generally convergent with other players that are rational. This is obvious in games where the equilibrium is a mixed policy, which ILs cannot learn.

**Joint Action Learners.** JALs (Claus & Boutilier, 1998), or opponent modelling Q-learning (Uther & Veloso, 1997), observe the actions of the other agents. They assume the other players are selecting actions based on a stationary policy, which they learn and play optimally with respect to this learned policy. Like ILs they are rational, but not convergent, since they also cannot learn mixed policies.

**Minimax-Q.** Minimax-Q (Littman, 1994) and Hu & Wellman's extension of it to general-sum SGs (Hu & Wellman, 1998) take a different approach. These algorithms observe both the actions and rewards of the other players and try to learn a Nash equilibrium explicitly. The algorithms learn and play the equilibrium independent of the behavior of other players. These algorithms are convergent with rational players, since they always converge to a stationary policy, so the other players being rational will converge to optimal responses. However, these algorithms are not rational. This is most obvious when considering a game of Rock-Paper-Scissors against an opponent that almost always plays "Rock". Minimax-Q will still converge to the equilibrium solution, which is not optimal given the opponent's policy.

In essence, we are looking for a learning technique that is rational, and therefore plays optimally when playing not necessarily rational players. Yet, it should still be capable of converging to mixed policies. We want the rational behavior of ILs and JALs, and the convergent behavior of minimax-Q.

### 3.2.2 A New Algorithm

In this section we contribute an algorithm towards the goal of a rational and convergent learner. We first introduce an algorithm that is rational and capable of playing mixed policies, but does not converge in empirical experiments. We then introduce a modification to this algorithm that results in a rational learner that does converge to mixed policies.

**Policy Hill Climbing.** A simple extension of Q-learning to play mixed strategies is policy hill-climbing (PHC) as shown in Table 9. The algorithm, in essence, performs hill-climbing in the space of mixed policies. Q-values are

1. Let $\alpha$ and $\delta$ be learning rates. Initialize,
$$Q(s,a) \leftarrow 0, \qquad \pi(s,a) \leftarrow \frac{1}{|\mathcal{A}_i|}.$$
2. Repeat,
   (a) From state $s$ select action $a$ according to mixed strategy $\pi(s)$ with some exploration.
   (b) Observing reward $r$ and next state $s'$,
$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a')\right).$$
   (c) Update $\pi(s,a)$ and constrain it to a legal probability distribution,
$$\pi(s,a) \quad \leftarrow \quad \pi(s,a) + \begin{cases} \delta & \text{if } a = \operatorname{argmax}_{a'} Q(s,a') \\ \frac{-\delta}{|A_i|-1} & \text{Otherwise} \end{cases}.$$

Table 9: The policy hill-climbing algorithm (PHC) for player $i$.

maintained just as in normal Q-learning. In addition the algorithm maintains the current mixed policy. The policy is improved by increasing the probability that it selects the highest valued action according to a learning rate $\delta \in (0,1]$. Notice that when $\delta = 1$ the algorithm is equivalent to Q-learning, since with each step the policy moves to the greedy policy executing the highest valued action with probability $1$ (modulo exploration).

This technique, like Q-learning, is rational and will converge to an optimal policy if the other players are playing stationary strategies. The proof follows from the proof of Q-learning, which guarantees the $Q$ values will converge to $Q^*$ with a suitable exploration policy[7]. Similarly, with a suitable exploration policy $\pi$ will converge to the policy that is greedy according to $Q \to Q^*$, which is a best response. Despite the fact that it is rational and can play mixed policies, it still doesn't show any promise of being convergent. We show examples of its convergence failures in Section 3.2.3.

[7] We mostly ignore the issue of exploration. See (Singh et al., In Press) for suitable exploration policies for online learning.

**Adjusted Policy Hill-Climbing.** We now introduce a modification to the naive policy hill-climbing algorithm, which encourages the desired convergence property without sacrificing the rational property. The algorithm, called adjusted policy hill-climbing (APHC), consists of a modification to the algorithm's learning rate. The modification has a very simple intuition: "Learn quickly while losing and slowly while winning." The required changes to policy hill-climbing are shown in Table 10.

1. Let $\alpha, \delta_l > \delta_w$ be learning rates. Initialize,
$$Q(s,a) \leftarrow 0, \qquad \pi(s,a) \leftarrow \frac{1}{|\mathcal{A}_i|}, \qquad C(s) \leftarrow 0.$$
2. Repeat,
   (a) Same as PHC in Table 9
   (b) Same as PHC in Table 9
   (c) Update estimate of average policy, $\bar{\pi}$,
$$\begin{aligned} C(s) &\leftarrow C(s) + 1 \\ \forall a' \in \mathcal{A}_i \quad \bar{\pi}(s,a') &\leftarrow \frac{1}{C(s)} \left(\pi(s,a) - \bar{\pi}(s,a)\right). \end{aligned}$$
   (d) Update $\pi(s,a)$ and constrain it to a legal probability distribution,
$$\pi(s,a) \quad \leftarrow \quad \pi(s,a) + \begin{cases} \delta & \text{if } a = \operatorname{argmax}_{a'} Q(s,a') \\ \frac{-\delta}{|A_i|-1} & \text{Otherwise} \end{cases},$$
   where,
$$\delta = \begin{cases} \delta_w & \text{if } \sum_{a'} \pi(s,a')Q(s,a') > \sum_{a'} \bar{\pi}(s,a')Q(s,a') \\ \delta_l & \text{otherwise} \end{cases}.$$

Table 10: The adjusted policy-hill climbing algorithm (APHC) for player $i$.

This aids in convergence by giving more time for the other players to adapt to changes in the player's strategy that at first appear beneficial, while allowing the player to adapt more quickly to other players' strategy changes when they are harmful. Practically, the algorithm requires two learning parameters $\delta_l > \delta_w$. The parameter that is used to update the policy depends on whether the agent's current expected value is greater than the expected value of the average policy. If the current expected value is lower (i.e. the agent is "losing") then the larger learning rate $\delta_l$ is used.

Although the learning rate modification may appear to be a somewhat unfounded heuristic, it has been studied in some form in other areas, notably when considering an adversary. In evolutionary game theory the *adjusted replicator dynamics* (Weibull, 1995) scales the individual's growth rate by the inverse of the overall success of the population. This will cause the population's composition to change more quickly when the population as a whole is performing poorly. A form of this also appears as a modification to the *randomized weighted majority* algorithm (Blum & Burch, 1997). In this algorithm, when an expert makes a mistake, a portion of it's weight loss is redistributed among the other experts. If the algorithm is placing large weights on mistaken experts (i.e. the algorithm is "losing"), then a larger portion of the weights are redistributed (i.e. the algorithm adapts more quickly.)

Adjusted policy hill-climbing is still rational, since only the speed of learning is altered. Its convergence properties, though, are quite different. In the next section we show examples that this technique converges to rational policies for a number and variety of stochastic games.

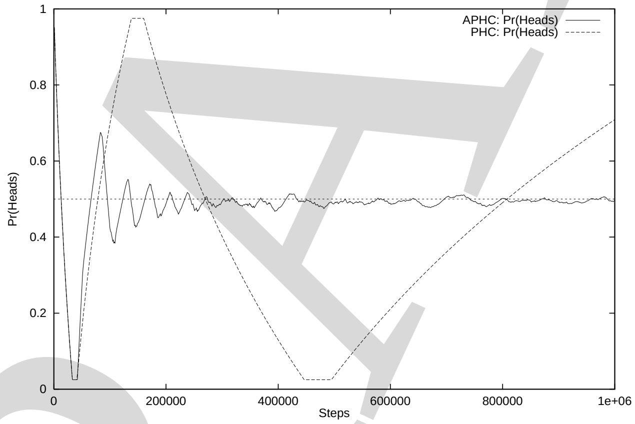
### 3.2.3 Results

In this section we show results of applying policy hill-climbing and adjusted policy hill-climbing to a number of different games. The domains include two matrix games, which help to show how the algorithms work and the effect of the adjustment on convergence. The algorithms were also applied to two multi-state SGs. One is a general-sum grid world domain used by Hu & Wellman (Hu & Wellman, 1998). The other is a zero-sum soccer game introduced by Littman (Littman, 1994).

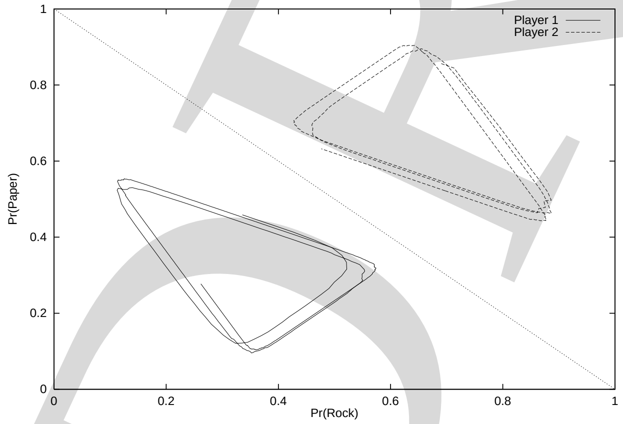
The experiments involve training the players using the same learning algorithm. Since PHC and APHC are rational, we know that if they converge against themselves, then they must have converged to a Nash equilibrium. For the matrix game experiments $\delta_l/\delta_w = 2$, but for the other results a more aggressive $\delta_l/\delta_w = 4$ was used. In all cases both the $\delta$ and $\alpha$ learning rates were decreased proportionately to $1/C(s)$, although the exact proportion varied between domains.

**Matrix Games.** The algorithm was applied to two zero-sum matrix games, matching-pennies and rock-paper-scissors, see Figure 1. In both games, the Nash equilibrium is a mixed policy consisting of executing the actions with equal probability. The large number of trials and small ratio of the learning rates were used for the purpose of illustrating how the algorithm learns and converges.

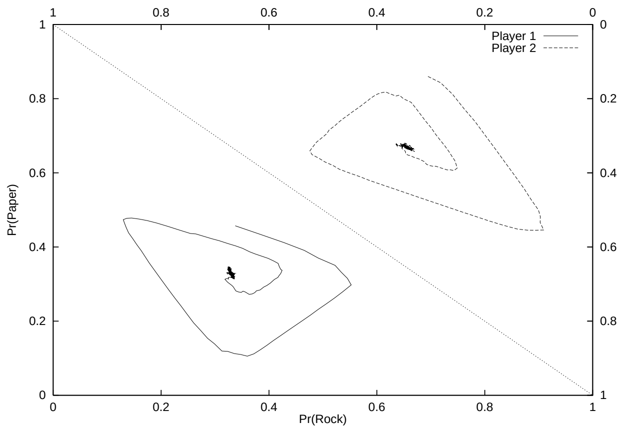
The results of applying both policy hill-climbing and adjusted policy hill-climbing to the matching pennies game is shown in Figure 4(a). APHC quickly begins to oscillate around the equilibrium, with ever decreasing amplitude. Without the adjustment, it oscillates around the equilibrium, but without any appearance of converging. This is even more obvious in the game of rock-paper-scissors. The results are shown in Figure 4(b,c), and show trajectories of the players' strategies in policy space through one million steps. Policy hill-climbing circles the equilibrium policy without any hint of converging, while adjusted policy hill-climbing very nicely spirals towards the equilibrium.




(a) Matching Pennies Game



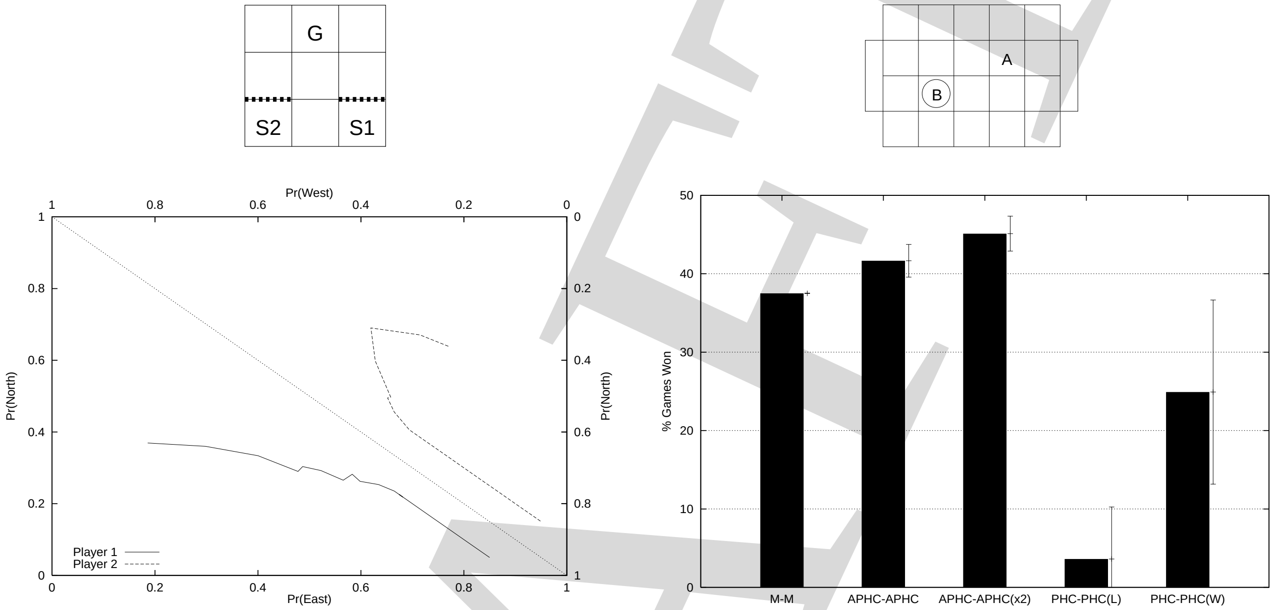

(b) RPS: Policy Hill-Climbing




(c) RPS: Adjusted Policy Hill-Climbing

Figure 4: (a) Results for the matching pennies matrix game: the policy for one of the players as a probability distribution while learning with PHC and APHC. The other player's policy looks similar. (b, c) Results for the rock-paper-scissors matrix game: trajectories of the two players' policies while learning with PHC (b), and APHC (c).

**Gridworld.** We also examined Hu's gridworld domain from Section 3.1.3 and Figure 3. As a reminder the game is general-sum and has two deterministic equilibria. They involve one player taking the lateral move and the other trying to move North. Hence the game requires that the players coordinate their behaviors.




(a) Gridworld Game

(b) Soccer Game

Figure 5: (a) Gridworld game. The dashed walls represent the actions that are uncertain. The results show trajectories of two players' policies while learning with APHC. (b) Soccer game. The results show the percentage of games won against a specifically trained worst-case opponent after one million steps of training.

Adjusted policy hill-climbing successfully converges to one of these equilibria. Figure 5(a) shows an example trajectory of the players' strategies for the initial state while learning over 100,000 steps. In this example the players converged to the equilibrium where player one moves East and player two moves North from the initial state. This is evidence that adjusted policy hill-climbing can learn an equilibrium even in a general-sum game with multiple equilibria.

**Soccer.** The final domain is a comparatively large zero-sum soccer game introduced by Littman (Littman, 1994) to demonstrate Minimax-Q. An example of an initial state in this game is shown in Figure 5(b), where player 'B' has possession of the ball. The goal is for the players to carry the ball into the goal on the opposite side of the field. The actions available are the four compass directions. The players select actions simultaneously but they are executed in a random order, which adds non-determinism to their actions. If a player attempts to move to the square occupied by its opponent, the stationary player gets possession of the ball, and the move fails. Unlike the grid world domain, the Nash equilibrium for this game requires a mixed policy. In fact any deterministic policy (therefore anything learned by an IL or JAL) can always be defeated (Littman, 1994).

Our experimental setup resembles that used by Littman in order to compare with his results for Minimax-Q. Each player was trained for one million steps. After training, its policy was fixed and a challenger using Q-learning was trained against the player. This determines the learned policy's worst-case performance, and gives an idea of how close the player was to the equilibrium policy, which would perform no worse than losing half its games to its challenger. Unlike Minimax-Q, APHC and PHC generally oscillate around the target solution. In order to account for this in the results, training was continued for another 250,000 steps and evaluated after every 50,000 steps. The worst performing policy was then considered the value of the policy for that learning run.

Figure 5(b) shows the percentage of games won by the different players when playing their challengers. "M-M" represents Minimax-Q when trained against itself (the results were taken from Littman's original paper.) "APHC-APHC" represents adjusted policy hill-climbing trained against itself. "PHC-PHC(L)" and "PHC-PHC(W)" represents policy hill-climbing with $\delta = \delta_l$ and $\delta = \delta_w$, respectively. "APHC-APHC(2x)" represents adjusted policy hill-climbing trained with two million steps of training. The performance of the policies were averaged over fifty training runs and the standard deviations are shown by the lines beside the bars. The relative ordering by performance is statistically significant.

APHC does extremely well, performing equivalently to Minimax-Q with the same amount of training[8] and continues to improve with more training. The exact affect of the adjustment can be seen by its out-performance of PHC, using either the larger or smaller learning rate. This shows that the success of APHC is not simply due to changing learning rates, but rather to changing the learning rate at the appropriate time to encourage convergence.

#### 3.2.4 Discussion

Adjusted policy hill-climbing is both rational and capable of converging to mixed policy equilibria. In this section we present justification for the importance of these properties.

**Convergence.** One major reason for desiring techniques that converge to stationary policies is a matter of practicality. Non-stationary policies are very difficult to evaluate, requiring a different value criteria than discounted future reward (e.g. average reward.) In addition, non-stationary policies can add a major complication to the problem of assigning credit with delayed reward. If the value of a state can oscillate as the policy changes then credit assignment based on this value can lead to an even poorer policy.

**Rationality.** The alternative to a rational learner is one that directly attempts to learn a game's equilibrium solution, like Minimax-Q. Problems with such an approach is that it assumes the other players will cooperate and also play according to the equilibrium. This approach completely fails with general-sum games that have multiple equilibria, where selection of the same equilibria is itself a coordination problem with no answer independent of the other players' policies.

Even if you assume there's a single equilibrium and the other player is trying to play optimally, there are still reasons why directly learning the equilibrium may not be desirable. The other players may have physical limitations that make them not capable of playing the equilibrium solution. They may not be capable of executing some action, or they can't differentiate between which of their actions get executed. The other players may also have rational limitations, such as using a function approximator to represent policies or values. We believe these sorts of limitations are going to become much more common as multiagent learning begins to address larger problems.

### 3.3 Subproblem Reuse as a Limitation

In this section we discuss the idea of subproblem reuse. We show how this limitation can be useful for increasing the speed of learning in a restricted class of single-agent MDPs (Bowling & Veloso, 1999). The subset of MDPs are called *goal state MDPs* and replace the usual reward function with a set of goal states, $\mathcal{G} \subseteq \mathcal{S}$. Reaching a goal state ends the trial, so all actions from a goal state simply transition to the same state with zero reward. Other transitions into goal states receive a reward of 1, while all other transitions receive a reward of 0. Goal state MDPs are therefore defined by the tuple $(\mathcal{S}, \mathcal{A}, T, \mathcal{G})$.

#### 3.3.1 Subproblem Formalization

We first need to formalize the notion of a subproblem within the framework of our restricted class of MDPs. We also need to define what it means to *use* a subproblem. These notions hinge on the definition of the boundary of a set of states.

**Definition 3** *For a set of states, $S$, let the* boundary *of $S$, denoted $B(S)$, be the set of all states not in $S$ with a non-zero probability transition from a state in $S$.*

In other words the boundary of $S$ are all the states not in $S$ that are reachable from $S$ in a single time step.

We can now define our notion of a subproblem.

[8] The results are not directly comparable due to the use of a different decay of the learning rate. Minimax-Q uses an exponential decay which decreases too quickly for use with APHC.

**Definition 4** *A subproblem, $\mathbb{S}$, of a goal state MDP, $\mathbb{P} = (\mathcal{S}, \mathcal{A}, T, \mathcal{G})$, is a pair $(\mathcal{S}', \mathcal{G}')$ satisfying:*

- $\mathcal{S}', \mathcal{G}' \subseteq \mathcal{S}$,
- $\mathcal{S}' \cap \mathcal{G} = \emptyset$ *and* $B(\mathcal{S}') \cap \mathcal{G} = \emptyset$,
- $\mathcal{G}' \subseteq B(\mathcal{S}')$.

*This definition induces a new MDP over the states $\mathcal{S}'$. The state transitions remain the same as $\mathbb{P}$, but the reward function is zero for all transitions except those entering states in $\mathcal{G}'$, where the reward is one.*

Given this notion of a subproblem we want to investigate the effects of using a subproblem solution. So we define a formal notion of *using* a subproblem.

**Definition 5** *For a subproblem, $\mathbb{S} = (S', G')$, we say that a policy, $\pi$ for $\mathbb{P}$* uses $\mathbb{S}$ *if and only if $\pi$ restricted to states in $\mathcal{S}'$ is optimal for the subproblem, $\mathbb{S}$. Additionally, the optimal policy for $\mathbb{P}$ that* uses $\mathbb{S}$ *is denoted $\pi^*_{\mathbb{S}}$.*

### 3.3.2 Examples

Given this definition of subproblems and how they are used, we want to examine the tradeoffs of reusing subproblem solutions. We explored this in the context of some simple stochastic maze problems, which are within our class of goal state MDPs. The two problems discussed here are representative of more complex MDPs.

**Hallway Example.** Figure 6 depicts a simple grid environment that was first investigated by Thrun and Schwartz (1995) within the context of sharing policies. The agent has actions that move it in any of the eight grid directions (including diagonal moves), but with probability 0.1 its action is ignored and it is moved in a random grid direction. There is a single goal state labeled 'G' in the diagram. The agent receives a reward of 1 for a transition into the goal state, and a reward of 0 otherwise. The agent is trying to maximize its discounted future reward with a discount factor, $\gamma = 0.9$.

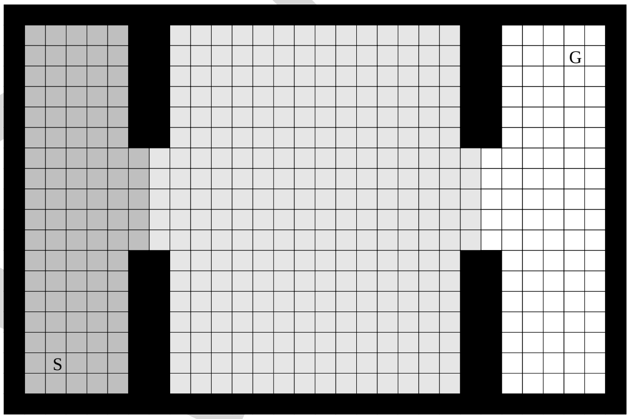



Figure 6: Hallway example. Each grid position represents a state. 'S' denotes the start state, and 'G' the goal state. The shaded regions mark off two different subproblems.

The shaded regions in the diagram denote subproblems. For example the subproblem of exiting the left room has the dark shaded region as its state space, $\mathcal{S}'$. The five states in the rightmost half of the doorway into the middle room are the boundary of $S'$, and in this case $\mathcal{G}' = B(\mathcal{S}')$. Another subproblem is exiting both the left and middle rooms. In this case the states, $\mathcal{S}'$, would be both shaded regions, and the goal states would be the five rightmost states of the doorway to the room on the right.

We explored how using the solution to these subproblems might speed learning of the complete problem. We fixed the policy to always select the subproblem's optimal action while within the subproblem's state space (i.e. the shaded regions.) We then use Q-Learning (Watkins, 1989) with a decreasing learning factor in order to learn the optimal policy given this constraint.

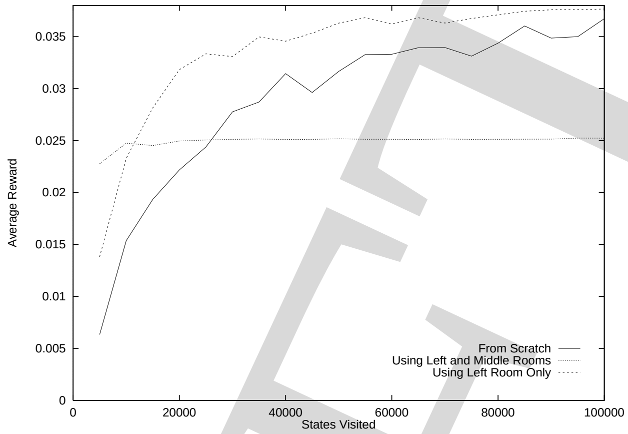



Figure 7: Results from the hallway example.

The results of this learning can be seen in figure 7. When we use the subproblem of exiting the left and middle rooms, we have a huge boost in learning speed. After visiting 5000 states, it performs approximately six times better than learning from scratch. Notice that this sacrifices convergence to the optimal solution. The more conservative approach of using the subproblem of exiting just the left room has a more conservative learning improvement. After visiting 5000 states it performs almost three times better than learning from scratch. Although the improvement isn't as pronounced, it appears to converge to an optimal solution.

**Doorways Example.** A second example that we explored consisted of a similar grid world but with multiple doorways, where $\mathcal{G}' \neq B(\mathcal{S}')$. This is depicted in figure 8. The actions and transitions remain the same as was described for the hallways example. Like that example, this also has intuitive subproblems, such as reaching a subset of the three doorways. We explored this domain using a number of these subproblems. Results are shown in figure 9 for learning when using the subproblem of reaching the right doorway, reaching either the right or middle doorway, and learning from scratch.

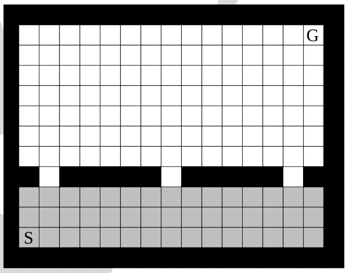



Figure 8: Doorways example. 'S' denotes the start state, and 'G' the goal state. The shaded region marks off the state space of the subproblem.

The results are very similar to the previous example. Subproblems speed learning but are likely to converge on a suboptimal solution. These results also motivate our goal, to be able to make a claim about the amount of optimality sacrificed in order to gain the improved learning speed.

#### 3.3.3 Bounding the Suboptimality

In order to better judge this tradeoff it is important to know the suboptimality of the solution obtained. Formally, we want to find a bound on the suboptimality of $\pi^*_{\mathbb{S}}$. This would provide us with a quantitative measure of the trade-off of using $\mathbb{S}$ as a subproblem. Before we discuss possible bounds we need a notion of suboptimality.

**Definition 6** *A policy, $\pi$, is $\alpha$-*optimal *if and only if* $\forall s\ V^\pi(s) \geq \alpha V^*(s)$.

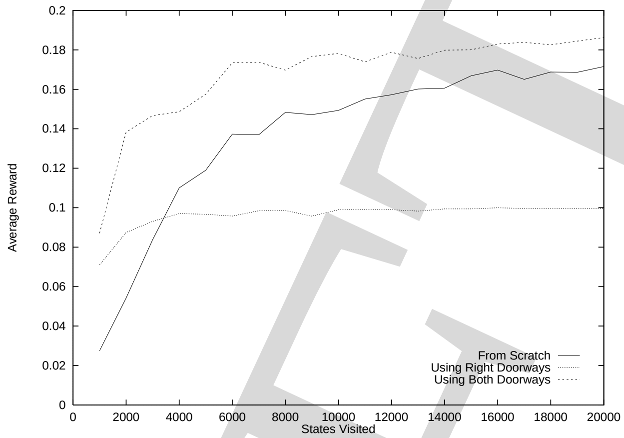



Figure 9: Results from the doorways example.

| | | Optimality | | |
|---|---|---|---|---|
| **Example** | **Subproblem** | **Actual** | **Bellman Bound** | **Boundary Bound** |
| Hallway | Left and Middle Rooms | 0.66 | 0.01 | 0.65 |
| | Left Room | 0.99 | 0.22 | 0.98 |
| Doorways | Right and Middle Doorway | 0.99 | 0.38 | 0.77 |
| | Right Doorway | 0.52 | 0.04 | 0.16 |

Table 11: Comparison of the suboptimality bounds computed from the Bellman error and from our method, against the actual suboptimality of using the subproblem. Larger values signify a more optimal solution (or a better guarantee).

Using this definition, $\alpha$ values are in the range $[0, 1]$, where large values are closer to the optimal solution. A summary of the optimality for our examples (figures 7 and 9) is given in table 11, under the column labeled "actual". These values were computed by taking the minimum over all states of the ratio between the value of the state when using the subproblem and the optimal value of the state. We can quantify our goal as finding a guaranteed lower bound for these values without requiring knowledge of the optimal policy or value.

We will first examine a bound based on Bellman error and give some empirical results on our examples that show that this bound is not satisfactorily tight. We will then introduce a new bound based on boundary values and show its results on these same examples.

**Bound Using Bellman Error.** The use of the Bellman error to bound the suboptimality of a policy was developed by Williams and Baird (1993). Bellman error, as mentioned earlier, is simply the maximum error in the Bellman equations, and the optimal policy would have a Bellman error of zero. They proved the following for all MDPs,

**Theorem 2** *For any value function, $V$, with Bellman error, $\epsilon$. Let $\pi$ be a policy greedy for $V$. Then for all states, $x$,*

$$V^{\pi}(x) \geq V^*(x) - \frac{2\gamma\epsilon}{1-\gamma}.$$

Qualitatively, this means we can bound the suboptimality of a policy by examining the Bellman error of a value function that induces that policy.

We can use this to compute a bound by simply calculating the Bellman error of $V_{\mathbb{S}}^*$. We define a value function that induces $\pi_{\mathbb{S}}^*$ and satisfies the Bellman equations for states in $\mathcal{S}'$ (because we are using an optimal solution to the subproblem) and also for states outside of $\mathcal{S}'$ (because we are acting optimally outside of $\mathcal{S}'$.) Then the only states with a non-zero Bellman error would be on the boundary of $\mathcal{S}'$.

**Results from the Examples.** Table 11 gives the results of using this bound in the subproblems in subsection 3.3.2. These values were computed by finding the optimal solution using the subproblem, calculating the Bellman error for that value function, and applying the equation. The results are quite disappointing. Even for subproblems that are nearly optimal (e.g. the left and middle rooms of the hallway problem) result in a weak constraint on the suboptimality. The computed bound does not help identify when a subproblem results in a good solution.

**Bound Using Boundary Values.** The following two definitions play a crucial role in our bound on suboptimality. The first provides a quantitative measure of the stochasticity of a problem. The second provides a measure for comparing values of sets of states.

**Definition 7** *Let $\rho_s$ be the probability that following the optimal policy at $s$ will transition to a state, $s'$, where $V^*(s') < V^*(s)$. Let $\rho = \max_{s \in \mathcal{S}} \rho_s$. This captures the stochasticity of the problem, where $\rho = 0$ for a deterministic problem.*

**Definition 8** *For a set of states $S_1$ and $S_2$, and a policy $\pi$, we say $S_1 \geq^{\pi}_{\alpha} S_2$ if and only if,*

$$\forall s_1 \in S_1, s_2 \in S_2 \qquad V^{\pi}(s_1) \geq \alpha V^{\pi}(s_2).$$

*If $S \geq^{\pi}_{\alpha} S$ then $S$ is $\alpha$ -equivalent with respect to $\pi$, i.e. the values of all the states in the set are within $\alpha$.*

The following theorem provides a bound on the suboptimality of using a subproblem where the goal states for the subproblem are its entire boundary.

**Theorem 3** *Given a subproblem $\mathbb{S} = (\mathcal{S}', B(\mathcal{S}'))$ of $\mathbb{P}$, if $B(\mathcal{S}')$ is $\alpha$-equivalent with respect to $\pi^*_{\mathbb{S}}$, then,*

*where $\beta = \frac{1-\sigma}{1-\sigma\alpha}$ and $\sigma = \frac{\rho\gamma^2}{1-(1-\rho)\gamma}$.*

The $\sigma$ value may appear cryptic but it is simply the discounted probability of ever reaching a state of lower value. The value emerges from the proof which is presented in appendix A.

We have attempted to generalize the theorem to all subproblems with the following conjecture,

**Conjecture 1** *Given a subproblem $\mathbb{S} = (\mathcal{S}', \mathcal{G}')$ of $\mathbb{P}$, let $\bar{\mathcal{G}}' = B(\mathcal{S}') \backslash \mathcal{G}'$. If $\mathcal{G}' \geq^{\pi^*_{\mathbb{S}}}_{\alpha_1} \mathcal{G}'$ and $\bar{\mathcal{G}}' \leq^{\pi^*_{\mathbb{S}}}_{\alpha_2} S$, then*

*$\pi^*_{\mathbb{S}}$ is $\alpha_1\alpha_2\beta$-optimal for $\mathbb{P}$,*

*where $\beta = \frac{1-\sigma}{1-\sigma\alpha_1\alpha_2}$ and $\sigma = \frac{\rho\gamma^2}{1-(1-\rho)\gamma}$.*

**Theorem 4** *Conjecture 1 is true if $\mathbb{P}$ is deterministic (i.e. if $\rho = 0$.)*

Theorem 4 proves the conjecture true for deterministic MDPs and the $\beta$ factor should provide the necessary adjustment for the non-determinism as it did in theorem 3. The proof for theorem 4 is given in appendix B.

**Results from the Examples.** Table 11 gives the results of using this bound in our example subproblems. The values were computed much like those for the Bellman error; an optimal solution was learned using the subproblem, the $\alpha$ values were calculated, and the bound computed. The results show a reasonably tight bound, especially for the single exit subproblems. Thought it's not as tight on the multiple exit subproblems it is still much improved over the bound computed from Bellman error.

This bound provides considerable information as to the loss in optimality of using a subproblem. In addition to this guarantee on the resulting policy it also helps to identify a possibly better subproblem. This is done by examining the $\alpha$ values. For example, in the doorways problem using the right doorway, the guaranteed optimality is fairly weak. But, by examining the $\alpha$ values it becomes obvious that this bound could be improved if the middle door was in the subproblem's goal states. By using the "new" subproblem we achieve a better policy and a better bound on its suboptimality.

# 4 Proposed Research

I propose to study the problem of learning to act in the presence of other learning agents, specifically when these agents may have realistic limitations on their capabilities or rationality. In general this problem can be nearly impossible as other agents can arbitrarily change their actions, which reduces the problem to that of learning in an arbitrary non-stationary environment. Realistically, other agents rarely change their policies arbitrarily. Rather they adapt their behavior towards specific goals. It is precisely this regularity that makes the multiagent learning problem possible.

As I presented in Section 2, current multiagent algorithms rely on the assumption that agents always act optimally. They limit what they can learn to equilibrium solutions, which are the only stationary strategies that make sense against an agent that is perfectly optimal. In situations where multiple equilibria exist, the exact rewards of the other agents are unknown and unobservable, or where agents can not or do not act perfectly optimal, then these algorithms don't make sense. I propose to develop learning algorithms to specifically address the latter case, where agents do not always act optimally. I hope, though, to apply the same techniques to the other multiagent situations where current techniques fail.

In particular, I propose to:

- Continue development of multiagent learning algorithms that are rational and convergent.
- Examine the existence of equilibria in stochastic games when other players have limitations.
- Investigate extensions of rational and convergent learning algorithms to address limited agents.
- Examine questions that arise in multiagent learning, such as partial observability and knowledge transfer.

The specifics of my proposed research are divided into the problems and issues which I will definitely investigate, and other work which I hope to address.

## 4.1 Main Focus

The main focus of my proposed work is to develop multiagent learning algorithms that can address situations where other agents have realistic limitations. This goal can be divided into a number of steps.

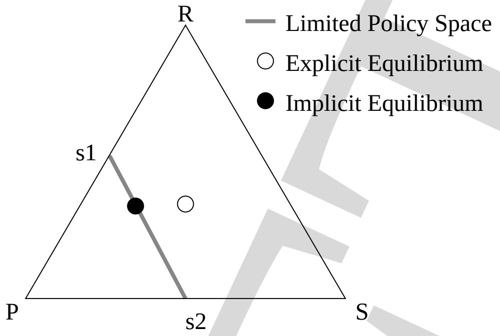
### 4.1.1 Rational and Convergent Learning Techniques

The rational and convergent properties embody much of the goal of learning with multiple agents. I am interested in learning techniques that are rational and will converge against a class of players that includes practical learning techniques, which may be subject to limitations. A first step towards this goal is to focus on players at the two extremes of the limitation continuum: players that are completely rational and players that do not learn at all (i.e. they follow stationary policies.)

I hope to continue our work from Section 3.2, which introduces an algorithm, APHC, that converges when playing agents that do not learn and agents that are also using APHC. Specifically, I hope to extend the learning principle in this algorithm to construct a family of learning techniques: model-based learner, SARSA on-policy learner, etc. I also plan to extend the Exp3 algorithm discussed in Section 2.4 to stochastic games as alternative learning method. I plan to examine the dynamics of these algorithms theoretically in the case when all players are using the same algorithm. I also hope to broaden the analysis to examine the algorithm when playing against broader class of learners.

### 4.1.2 Equilibria with Limited Agents

In order to extend the principles of rational and convergent learners to learning in the presence of agents with limitations, it first must be determined what is the "goal" of these algorithms. I am interested in learning against other agents that have limitations but are *otherwise rational*. That is they seek to maximize their reward within the scope of their limitations. This is similar to the notion of *bounded-optimal* agents (Russell, 1997), but applied to the other agents of a multiagent system. We know that if otherwise rational players were to converge to stationary policies, these policies must be an equilibrium solution given their limitations. That is, the players must be playing best responses given their limitations to the other players' policies.




| | Explicit Game | Implicit Game |
|---|---|---|
| Payoffs | $\begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$ | $\begin{pmatrix} 0 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{2} \end{pmatrix}$ |
| Equilibrium | $\left\langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \right\rangle, \left\langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \right\rangle$ | $\left\langle 0, \frac{1}{3}, \frac{2}{3} \right\rangle, \left\langle \frac{1}{3}, \frac{2}{3} \right\rangle$ |
| Limited Equilibrium | $\left\langle 0, \frac{1}{3}, \frac{2}{3} \right\rangle, \left\langle \frac{1}{3}, \frac{1}{2}, \frac{1}{6} \right\rangle$ | |

Table 12: Example of an equilibrium under limitations. The column player in Rock-Paper-Scissors is restricted to playing only linear combinations of the strategies $s_1 = \left\langle \frac{1}{2}, \frac{1}{2}, 0 \right\rangle$ and $s_2 = \left\langle 0, \frac{1}{2}, \frac{1}{2} \right\rangle$.

An important question that needs answered in order to address this problem is wether such an equilibrium exists under various type of limitations. This is not trivial. For example, consider the limitation restricting a player to only playing deterministic policies. In the game R-P-S there is no equilibrium given this limitation.

Some limitations, on the other hand, do preserve the existence of equilibrium. Consider the limitation which restricts a player to only playing linear combinations of a small set of policies.[9] Table 12 shows an example of this situation in R-P-S where one player can only play linear combinations of the strategies $s_1$ and $s_2$. The complete triangle is the complete space of mixed strategies (i.e. the three dimensional simplex projected into two dimensions). The grey line represents the class of strategies available to the limited agent. With this sort of limitation, it can be viewed that the limited player is actually unlimited in the context of a different SG, which I call the *implicit game*. The implicit game must have an equilibrium, and this equilibrium has a representation in the original explicit game, which satisfies the player's limited policy space. The table shows the matrices and equilibrium for the the implicit game, and the set of strategies this corresponds to in the explicit game. This shows that some limitations preserve the existence of equilibria, which is necessary if we desire to learn stationary policies.

I plan to formalize the notion of an *implicit game*, and examine classes of limitations that can be modeled with this notion. I also hope to define a broader class of policy limitations that still contain an equilibrium. Specifically I hope to prove the following conjecture,

**Conjecture 2** *For any zero-sum stochastic game, $(2, \mathcal{S}, \mathcal{A}_{1,2}, T, R_{1,2})$, where player 2's set of available policies, $\Pi_2$, is convex and compact. There exists policies, $\pi_1$ and $\pi_2$ such that,*

$$\forall s \in \mathcal{S} \qquad \begin{array}{lcl} V_1^{\pi_1,\pi_2}(s) & \geq & \max_{\pi_1' \in PD(\mathcal{A}_1)} V_1^{\pi_1',\pi_2}(s), \quad \textit{and} \\ V_2^{\pi_1,\pi_2}(s) & \geq & \max_{\pi_2' \in \Pi_2} V_2^{\pi_1,\pi_2'}(s). \end{array}$$

The conjecture claims that if a player is restricted to a policy space that is compact and convex, then there still exists an equilibrium point within the player's restricted space of policies. Notice that the example limitation in Table 12 satisfies the assumptions of a compact and convex policy set, while the limitation of deterministic policies fails the assumption, since the set of deterministic strategies is not convex.

[9] This is in fact a crude form of policy approximation by reducing the number of parameters that define a strategy or policy.

### 4.1.3 Learning with Limited Agents

I propose to extend our original work with rational and convergent learners and build upon my results of the existence of equilibria under limitations to the problem of learning in the presence of limited agents. I plan to investigate empirical learning in the context of agents with various realistic limitations. These limitations include:

- Physical limitations such as inability to execute certain actions, or the inability to distinguish between which of its actions get executed.
- Function approximation, specifically those that limit the space of policies that can be executed.
- Use of subproblems.
- Maximizing an incorrect/different reward function.
- Partial observability or state aliasing.

All of these limitations in some form restrict the agent to learning often suboptimal solutions. This means current learning techniques (e.g. Q-learning or Minimax-Q) will not be able to converge to policies that can take advantage of or compensate for the other agents' limitations. On the other hand, the scaling of reinforcement learning and the practical application to real problems demand these limitations making this problem critical to making multiagent learning actually practical.

Many of these limitations can be embodied within the notion of an implicit game, therefore preserving the existence of equilibrium. Others can fit into the assumptions of Conjecture 2 and so also may have an equilibrium. This shows that it is possible for rational and convergent learners to exist within the limited agent framework. I believe that algorithms similar to APHC from Section 3.2 show promise to be able to extend to these situations. I propose to examine such algorithms in detail within arealistic multiagent domains.

## 4.2 Other Possible Directions

There are also other interesting questions that I would like to address.

### 4.2.1 Partial Observability

Partial observability is an important issue in the application of learning algorithms to practical problems. The issues that arise in single-agent still remain in the multiagent learning problem. Although, the explicit representation of a mixed policy can be helpful for learning in POMDPs (Singh et al., 1994).

Additionally, a number of new partial observability issues are raised in the multiagent setting. Observability of the actions selected by the other agents may not be feasible. Likewise, the actual rewards received by the other agents may not be known or observable, or may be partially observable. I hope to address some of these issues in my development of multiagent learning techniques.

### 4.2.2 Multiagent Knowledge Transfer

Another issue in multiagent learning is that of knowledge transfer. The goal we present in this paper is to perform on-line learning of a best response policy when playing specific other players, possibly with limitations. This raises the question of what sort of knowledge can be transferred to playing against different players that may have different limitations. One obvious approach is to specifically learn a model of the transition function, which would easily transfer when changing opponents.

What is not obvious, is whether other higher-level knowledge might still be useful? For example, would the knowledge of an equilibrium solution for playing a particular player, be useful when learning against another player, where the equilbrium may be different? Is there other high-level, player independent knowledge about the structure of a game that may be useful to speed learning. I hope to consider some of these questions in my research.

## 4.3 Evaluation

The evaluation of my work will be based on compelling results from the development of multiagent learning techniques, both from a theoretical aspect and its application in an emprical domain. Theoretically, I will analyze the existence of equilibria in the presence of agents with limitations. I also plan to analyze properties of the devloped algorithms to learn in these situations. The theoretical results, though, will likely require a number of restrictions possibly on the game structure, the learning algorithms of the other players, or the applicable classes of limitations.

Empirically, I will test the algorithms on a wide variety of stochastic games, both simple controlled envioronments as well as examples considered by other researchers. The controlled experiments will consider a wide range of dimensions: game structure (e.g. zero-sum, fully-collaborative, general-sum), number of agents, and classes of limitations. I also plan on applying these techniques to a non-trivial domain involving learning agents with realistic limitations. I tentatively plan to apply these learning techniques to the RoboCup Rescue domain (Kitano et al., 1999), a simulated multiagent search and rescue task that is under development.

# A Proof of Theorem 3

**Proof.** The proof makes use of a special non-stationary policy, $\pi$. This policy initially selects the same action as $\pi^*_{\mathbb{S}}$ for states in $\mathcal{S}'$, and the same action as $\pi^*$ for states not in $\mathcal{S}'$. After reaching a state in $B(\mathcal{S}')$ the policy changes so that it always selects the same action as $\pi^*_{\mathbb{S}}$. Notice that $V^*_{\mathbb{S}}(s) \geq V^{\pi}(s) \quad \forall s \in \mathcal{S}$. This is because $\pi$ *uses* $\mathbb{S}$, and non-stationary policies cannot do better than an optimal stationary one.

Let $s_m = \operatorname{argmax}_{s \in B(\mathcal{S}')} V^*(s)$. We will first show that $\pi^*_{\mathbb{S}}$ is $\beta$-optimal for $s_m$. From this we will be able to show that for all other states it is $\alpha\beta$-optimal.

We can write $V^*(s_m)$ as,

$$V^*(s_m) = \sum_{\text{all paths}} \Pr(path)\,\mathrm{Value}(path).$$

We can then split the sum into the paths containing a state in $\mathcal{S}'$ and those that don't contain a state in $\mathcal{S}'$. Let $p$ be the probability a path from $s_m$, following the optimal policy, contains a state in $\mathcal{S}'$. Then,

$$\begin{aligned} V^*(s_m) = \quad & (1-p) \quad E(\mathrm{Value}(path) | \mathcal{S}' \cap path = \emptyset) + \\ & \quad p \qquad E(\mathrm{Value}(path) | \mathcal{S}' \cap path \neq \emptyset). \end{aligned}$$

Let $\nu = (1-p)E(\mathrm{Value}(path)|\mathcal{S}' \cap path = \emptyset)$. Notice that any path that contains a state in $\mathcal{S}'$ must also contain a state in $B(\mathcal{S}')$. So we can write the second term as a value of that state discounted by the length of the path before the state is reached. Let $\ell(path, s)$ be the number of steps in $path$ until $s$ is reached. So,

$$V^*(s_m) = \nu + \sum_{\substack{\text{paths with} \\ s \in B(\mathcal{S}')}} \Pr(path)\gamma^{\ell(path,s)} V^*(s).$$

Since $V^*(s_m) \geq V^*(s)$, we can substitute $V^*(s_m)$ for $V^*(s)$ and pull it outside of the sum. The resulting sum is just the expected discount between $s_m$ and $s$ when following the optimal policy. Let this value be $\delta$. So we get,

$$\begin{aligned} V^*(s_m) &\leq \nu + p\delta V^*(s_m) \\ &\leq \left(\frac{\nu}{1-p\delta}\right). \end{aligned}$$

We can also achieve a similar result for $V^*_{\mathbb{S}}(s_m)$. Since our non-stationary policy, $\pi$, acts the same as $\pi^*$ for $s \notin \mathcal{S}'$, then it will achieve the same value for paths not containing $s \in \mathcal{S}'$. As before, paths containing a state in $\mathcal{S}'$, must contain a state in $B(\mathcal{S}')$. And upon entering this state $\pi$ will select actions according to $\pi^*_{\mathbb{S}}$. So we can write,

$$V^*_{\mathbb{S}}(s_m) \geq V^{\pi}(s_m) = \nu + \sum_{\substack{\text{paths with} \\ s \in B(\mathcal{S}')}} \Pr(path)\gamma^{\ell(path,s)} V^*_{\mathbb{S}}(s).$$

By the $\alpha$-equivalence of $B(\mathcal{S}')$ we know that $V^*_{\mathbb{S}}(s) \geq \alpha V^*_{\mathbb{S}}(s_m)$. So we can substitute $\alpha V^*_{\mathbb{S}}(s_m)$ for $V^*_{\mathbb{S}}(s)$ and pull it outside of the sum. The resulting sum is the expected discount between $s_m$ and $s$ when following $\pi$. Let this value be $\delta'$. So we get,

$$\begin{aligned} V^*_{\mathbb{S}}(s_m) &\geq \nu + p\delta'\alpha V^*_{\mathbb{S}}(s_m) \\ &\geq \left(\frac{\nu}{1-p\delta'\alpha}\right). \end{aligned}$$

Since $\pi$ is optimal over the subproblem, it will reach a state in $B(\mathcal{S}')$ with no worse expected discount than $\pi^*$. Hence, $\delta' \geq \delta$, and we can substitute $\delta$ for $\delta'$ and maintain the inequality.

Now we examine the ratio of the two values,

$$\frac{V_{\mathbb{S}}^*(s_m)}{V^*(s_m)} \geq \frac{\nu\left(\frac{1}{1-p\delta\alpha}\right)}{\nu\left(\frac{1}{1-p\delta}\right)} \geq \frac{1-p\delta}{1-p\delta\alpha}.$$

The value, $p\delta$, is the probability of returning to a state in $\mathcal{S}'$, discounted by the distance to that state and the distance from that state to a state in $B(\mathcal{S}')$. Since all states in $\mathcal{S}'$ have a smaller value than $s_m$, we can bound this by the discounted probability of ever reaching a state with smaller value, which can be computed from $\rho$,

$$\gamma(\rho\gamma + \rho(1-\rho)\gamma^2 + \rho(1-\rho)^2\gamma^3 \ldots) = \frac{\rho\gamma^2}{1-(1-\rho)\gamma} = \sigma.$$

So, $p\delta \leq \sigma$, hence,

$$\frac{V^\pi(s_m)}{V^*(s_m)} \geq \frac{1-\sigma}{1-\sigma\alpha} = \beta.$$

We can now show that $\pi_{\mathbb{S}}^*$ is $\alpha\beta$-optimal for any state, $s$. We can decompose the value of $s$ just as was done for $s_m$. Let $p$ be the probability that a path from $s$, following $\pi^*$, contains a state in $\mathcal{S}'$. Notice that if $s \in \mathcal{S}'$, then $p = 1.0$. We let $\nu$ and $\delta$ represent the same values with respect to $s$. Then we can bound $V^*(s)$, $V_{\mathbb{S}}^*(s)$, and their ratio as follows:

$$\begin{aligned}
V^*(s) &\leq \nu + p\delta V^*(s_m) \\
V_{\mathbb{S}}^*(s) &\geq \nu + p\delta\alpha V_{\mathbb{S}}^*(s_m) \\
\frac{V_{\mathbb{S}}^*(s)}{V^*(s)} &\geq \frac{\nu + p\delta\alpha V_{\mathbb{S}}^*(s_m)}{\nu + p\delta V^*(s_m)} \\
&\geq \frac{p\delta\alpha V_{\mathbb{S}}^*(s_m)}{p\delta V^*(s_m)} \\
&\geq \alpha\beta.
\end{aligned}$$

Hence $\pi_{\mathbb{S}}^*$ is $\alpha\beta$-optimal for $\mathcal{P}$. $\diamond$

# B Proof of Theorem 4

**Proof.** Since $\mathcal{P}$ is deterministic then $\rho = 0$ and $\beta = 1$. Also, for any state, $s$, and a policy there is a unique path of states to a state in $G$ that follows that policy. Trivially, if the optimal path (path of states from following the optimal policy) from $s$ does not contain a state in $\mathcal{S}'$ then $V_{\mathbb{S}}^*(s) = V^*(s)$.

Consider the case where the optimal path from $s$ contains a state in $\mathcal{S}'$. Then it must contain a state in $B(\mathcal{S}')$. Let $g$ be the last state in the optimal path where $g \in B(\mathcal{S}')$. So the optimal path from $g$ cannot contain a state in $\mathcal{S}'$, and therefore,

$$V_{\mathbb{S}}^*(g) = V^*(g).$$

Now, there are two cases:

1. $g \in \mathcal{G}'$. The path from following $\pi_{\mathbb{S}}^*$ will reach a state $g' \in \mathcal{G}'$ in fewer steps (because $\pi_{\mathbb{S}}^*$ is optimal at reaching goal states of the subproblem.) Since $\mathcal{G}' \geq_{\alpha_1}^{\pi_{\mathbb{S}}^*} \mathcal{G}'$, then

$$V_{\mathbb{S}}^*(g') \geq \alpha_1 V_{\mathbb{S}}^*(g) = \alpha_1 V^*(g),$$

and since $g'$ is reached in fewer steps,

$$V_{\mathbb{S}}^*(s) \geq \alpha_1 V^*(s).$$

2. $g \notin \mathcal{G}'$. Since, $\bar{\mathcal{G}'} \leq_{\alpha_2}^{\pi_{\mathbb{S}}^*} S$, then

$$V_{\mathbb{S}}^*(s) \geq \alpha_2 V_{\mathbb{S}}^*(g) = \alpha_2 V^*(g) \geq \alpha_2 V^*(s).$$

So, $\pi_{\mathbb{S}}^*$ is $\alpha_1\alpha_2$-optimal. $\diamond$

## References


Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. Gambling in a rigged casino: The adversarial multi-arm bandit problem. *36th Annual Symposium on Foundations of Computer Science* (pp. 322–331). Milwaukee, WI: IEEE Computer Society Press.

Baird, L. C., & Moore, A. W. (1999). Gradient descent for general reinforcement learning. *Advances in Neural Information Processing Systems 11*. MIT Press.

Blum, A., & Burch, C. (1997). On-line learning and the metrical task system problem. *Tenth Annual Conference on Computational Learning Theory*. Nashville, TN.

Bowling, M. (2000). Convergence problems of general-sum multiagent reinforcement learning. . In press.

Bowling, M., & Veloso, M. (1999). Bounding the suboptimality of reusing subproblems. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (pp. 1340–1345). Stockholm, Sweden: Morgan Kaufman.

Boyan, J. A., & Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. *Advances in Neural Information Processing Systems 7*. The MIT PRESS.

Carmel, D. (1997). *Model-based learning of interaction strategies in multi-agent systems*. Doctoral dissertation, Computer Science Department, Technion.

Carmel, D., & Markovitch, S. (1996). Learning models of intelligent agents. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Claus, C., & Boutilier, C. (1998). The dyanmics of reinforcement learning in cooperative multiagent systems. *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Filar, J., & Vrieze, K. (1997). *Competitive Markov decision processes*. New York: Springer Verlag.

Fink, A. M. (1964). Equilibrium in a stochastic n-person game. *Journal of Science in Hiroshima University, Series A-I*, *28*, 89–93.

Fudenberg, D., & Levine, D. K. (1999). *The theory of learning in games*. The MIT Press.

Gordon, G. (1999). *Approximate solutions to markov decision problems*. Doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA.

Howard, R. A. (1960). *Dynamic programming and markov processes*. The MIT Press.

Hu, J. (1999). *Learning in dynamic noncooperative multiagent systems*. Doctoral dissertation, Electrical Engineering and Computer Science, University of Michigan.

Hu, J., & Wellman, M. P. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 242–250). San Francisco: Morgan Kaufman.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjou, A., & Shimada, S. (1999). Robocup rescue: Search and rescue in large scale disasters as a domain for autonomous agents research. *Proceedings of IEEE Conference on Man, Systems, and Cybernetics*.

Kuhn, H. W. (Ed.). (1997). *Classics in game theory*. Princeton University Press.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 157–163). San Francisco: Morgan Kaufman.

Mangasarian, O. L., & Stone, H. (1964). Two-person nonzero-sum games and quadratic programming. *Journal of Mathematical Analysis and Applications*, *9*, 348–355.

Moore, A. W., & Atkeson, C. G. (1993). Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, *13*, 103–130.

Nash, Jr., J. F. (1950). Equilibrium points in $n$-person games. *PNAS*, *36*, 48–49. Reprinted in (Kuhn, 1997).

Osborne, M. J., & Rubinstein, A. (1994). *A course in game theory*. The MIT Press.

Owen, G. (1995). *Game theory*. Academic Press.

Peng, J., & Williams, R. J. (1996). Incremental multi-step q-learning. *Machine Learning*, *22*, 283–290.

Robinson, J. (1951). An iterative method of solving a game. *Annals of Mathematics*, *54*, 296–301. Reprinted in (Kuhn, 1997).

Russell, S. (1997). Rationality and intelligence. *Artificial Intelligence*, *94*, 57–77.

Sen, S. (Ed.). (1996). *Adaptation, coevolution and learning in multiagent systems: Papers from the 1996 aaai spring symposium*. Menlo Park,CA: AAAI Press. AAAI Technical Report SS-96-01.

Shapley, L. S. (1953). Stochastic games. *PNAS*, *39*, 1095–1100. Reprinted in (Kuhn, 1997).

Singh, S., Jaakkola, T., Littman, M. L., & Szepesvári, C. (In Press). Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*.

Singh, S. P., Jaakkola, T., & Jordan, M. I. (1994). Learning without state-estimation in partially observable markovian decision processes. *Proceedings of the Eleventh International Conference on Machine Learning*. San Francisco: Morgan Kaufman.

Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*. In press.

Sutton, R. S. (1988). Learning to predict by the method of temporal differences. *Machine Learning*, *3*, 9–44.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning*. The MIT Press.

Thrun, S., & Schwartz, A. (1995). Finding structure in reinforcement learning. *Advances in Neural Information Processing Systems 7* (pp. 385–392). The MIT Press.

Uther, W., & Veloso, M. (1997). *Adversarial reinforcement learning* (Technical Report). Carnegie Mellon University. Unpublished.

Uther, W. T. B., & Veloso, M. M. (1998). Tree based discretization for continuous state space reinforcement learning. *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Vrieze, O. J. (1987). *Stochastic games with finite state and action spaces*. No. 33. CWI Tracts.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Doctoral dissertation, King's College, Cambridge, UK.

Weibull, J. W. (1995). *Evolutionary game theory*. The MIT Press.

Weiß, G., & Sen, S. (Eds.). (1996). *Adaptation and learning in multiagent systems*. Berlin: Springer Verlag.

Williams, R. J., & Baird, L. C. (1993). *Tight performance bounds on greedy policies based on imperfect value functions*Technical Report). College of Computer Science, Northeastern University.