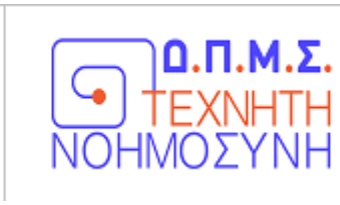
Fictitious play and Reinforcement Learning for computing equilibria

ΠΟΥΛΙΑΝΟΥ ΧΑΙΔΩ mtn2327

ΠΑΠΑΔΟΠΟΥΛΟΣ ΔΗΜΗΤΡΙΟΣ ΙΑΣΩΝ mtn2315



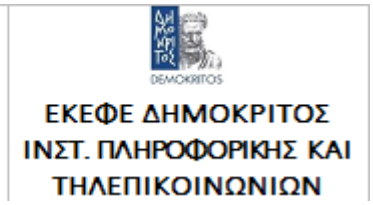


Table of Contents

Introduction

In game theory, the concept of equilibria plays a central role in understanding interactions among rational agents. An equilibrium represents a stable state in a game where agents' strategies are mutually consistent, and no player has an incentive to unilaterally deviate from their chosen strategy given the strategies chosen by others. Equilibria is important, because it can provide a framework for studying and predicting the best possible outcomes in games. This is of outmost importance especially in stochastic games, where the outcome is affected by randomness and thus equilibria can help predict in such environments the agents' interactions and outcome. In repeated games, equilibria can also be of benefit by predicting and choosing optimal strategy for agents over time.

However, computing equilibria is tricky and can prove complex, when more agents are introduced to the game. In this report, we will introduce how we can use fictitious play and reinforcement learning methods to compute equilibria in repeated zero-sum stochastic games. Along with the report, code implementation for a simple two-player zero-sum game using fictitious play and reinforcement learning strategies accordingly was also developed. The results of the evaluation of both methods are discussed and comparison is made on how Nash equilibria is computed for each method.

Overview

Multi-agent systems are defined by the cooperation of multiple agents interacting with each other in the same environment. Each agent interacts with their environment. If multiple agents interact on the same resources and influence the same part of the environment, then they are in a common sphere of effect. In a sphere of effect, agents are dependent on one another, then their behavior should follow some rules in order not to lead to unstable or contradiction. Each agent usually acts based on their own perception of the environment (state) and their own preferences and their goal is to choose an action that will maximize their utility function. However, in more complex environments where multiple agents are searching for a strategy that maximizes their profit, but their profit might damage the utility of someone else. These types of games are called zero-sum competitive games and will be discussed later. [3],[4]

In order to successfully describe the interactions of a multi-agent system in a mathematical way, the field of game theory comes into the scene. Game theory is a field where it contains algorithms that have been used for years for the general description and solution of interactions between multiagent systems. These multiagent systems typically find applications in Economics, Psychology, Biology and many more.[3],[4]

The main types of games studied are cooperative and competitive games. Competitive games refer to the existence of self-directed agents. These agents do not act in a heinous way to undermine the other agents, but only act in a way that will increase their own utility or payoff.

The payoff is a mathematical representation that gives a weighted number to every possible state of the agent, where the number represents how ideal their current state is.

As mentioned, the goal of the agent is to increase their payoff no matter what. This behavior, even involuntarily, affects the state of the other agents.

The study of interactions can be also formulated by two main representations: normal and non-normal form games. In our report we studied some of the most famous normal form games.

Normal form games: Normal form games are of limited amount of play time and can be descibed mathematicaly by the vector (N, A, u), where:

N is a finite set of n players, indexed by i

$A = A1 \times \cdots \times An$, where Ai is a finite set of actions available to player i .

Each vector $a = (a1, \dots, an) \in A$ is called an action profile of the game and

$u = (u1, \dots, un)$ where ui is a real-valued utility (or payoff) function for player i .

With the normal game formalism, the interactions of agents can easily be described through a payoff matrix, where payoff matrix is a helpful way of representing the state and the payoff for each involved agent.[6]

Repeated games

In normal form stage games (or one-shot games), the agent forms one strategy and act by it until the end of the game. In more realistic environments, the normal form is insufficient to capture the strategies and interactions among agents. Some of the limitations of the normal form include its inability to accommodate games of indefinite duration. A realistic game may involve repetitions and consist of a series of stages or states. Additionally, in a realistic game, agents may not have access to the opponent's payoff as assumed by the normal form algorithms. Furthermore, the normal form is inadequate for describing games of unlimited duration. To address these limitations, we are led to the description and study of a new category of games, known as repeated games.
Unlike one-shot games, repeated games are a form of game that repeats playing a stage game multiple times. Repeated games can be either type infinitely repeated or finitely repeated. Infinitely repeated games continue for an uncertain amount of time infinitely often, while finitely games are only played a known amount of times.[6]

Repeated games also add more complexity to the way an agent acts, since now the players can develop actions based on past experiences (memory) or adapt their strategy based on the opponent's moves. This new mode of play introduces new preferences and alters the players' decision-making strategy in the long-term. In short, agents are able to form a different strategy in each stage game. An example can be seen in Figure 1.1 where a visualization of the game Prisoner's Dilemma is seen as a branching decision tree. Each branch depicts the decision the agent can make as far as their actions go. Clearly, the decision space for this type of game is far richer that a simple stage game.[6]

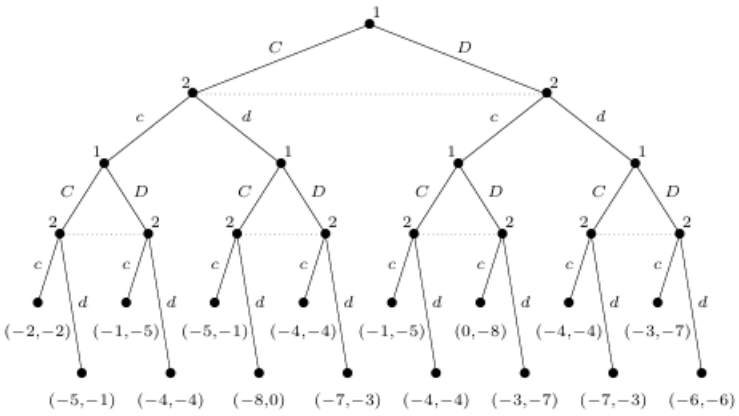


Figure 1.1: Extensive form of the finitely repeated game of Prisoner's Dilemma

If the agent chooses the same strategy in each stage game however, this is called a stationary strategy.

Zero Sum games

Zero-sum games are usually competitive types of games, where each player's win is another player's loss. One of these which we will also use in our experiments is Matching Pennies game. In Matching Pennies, each of the two-players have a penny and choose simultaneously whether to play Heads or Tails. They then simultaneously show their action to their opponent and if they match, player1 wins, otherwise player2 wins. This is a game of pure competition, since one agent's winning strategy gives maximum utility for them, but their opponent suffers the same amount of loss, while a tie gives zero reward. The payoff matrix is shown below:

	Heads	Tails
Heads	$1, -1$	$-1, 1$
Tails	$-1, 1$	$1, -1$

Figure 1.2 Mathing Pennies payoff matrix

Here it is evident that this two-player game is a constant sum game of zero value, since for every played strategy the sum of utilities is $uA + uB = 1 - 1 = 0$, where uA and uB are the utilities of the agents A and B [3],[5]

Nash equilibria in zero-sum games can be viewed graphically as a "saddle" in a high-dimensional space. At a saddle point, any deviation of the agent lowers his utility and increases the utility of the other agent. It is easy to visualize in the simple case in which each agent has two pure strategies. [3],[5]

We will show experiments of Nash equilibrium convergence of a repeated Matching Pennies in our experiments later.

Stochastic Games

As mentioned MDP is a type of game that describes the behavior of an agent in an uncertain, dynamic world. A stochastic game is a collection of normal-form games, in which the agents repeatedly play games from this collection, and the particular game played at any given iteration depends probabilistically on the previous game played and, on the actions, taken by all agents in that game. Based on the description, stochastic games are repeated games that generalize Markov decision processes (MDPs) to the multiagent setting. At each state all agents choose an action, and the transition function now depends on the joint action profile. As in other classes of games, the standard solution concept is the Nash equilibrium. [3]

Definition 6.2.1 (Stochastic game) *A* stochastic game *(also known as a* Markov game*) is a tuple* (Q, N, A, P, R)*, where:*

- Q *is a finite set of games;*
- N *is a finite set of* n *players;*
- $A = A_1 \times \cdots \times A_n$*, where* A_i *is a finite set of actions available to player* i*;*
- $P : Q \times A \times Q \mapsto [0, 1]$ *is the transition probability function;* $P(q, a, \hat{q})$ *is the probability of transitioning from state* q *to state* $\hat{q}$ *after action profile* a*; and*
- $R = r_1, \ldots, r_n$*, where* $r_i : Q \times A \mapsto \mathbb{R}$ *is a real-valued payoff function for player* i*.*

Figure 1.3: Stochastic Game Definition [3]

While in two-player games an equilibrium can be computed in polynomial time, for three-player games are way more complex and there are no known algorithms that are guaranteed to converge to an equilibrium in three-player stochastic games (even in the zero-sum case).[3]

Even in two-player games that have been proven to find a Nash equilibria, their computation time is polynomial as mentioned which makes the process of finding it very slow. The computation becomes way more complex, or even infeasible, for larger games. Another limitation is that prior knowledge of the payoff matrix for all players should be known and the game should be well defined in order to formulate it into a linear programming problem. For these reasons, using learning methods like Reinforcement Learning can help compute equilibria with less complexity. [3]

A simple stochastic game example is a variation of the famous Matching Pennies, where the stochastic element introduced is that each player now makes a random probabilistic choice that simulates the randomness of a coin tossed. This means they assign probabilities to each of their actions. The expected payoff of player1 with random probability p and q being the probability of player2:

$$E1\,(p,q)=p(1-q)-(1-p)q$$

Strategies

A strategy represents the choices an agent has in order to take an action. There exist two main strategies: pure strategies and mixed strategies. An action taken in a one stage game is called a pure strategy. On the other hand, if an agent takes an action with a random probability distribution on all their avaiable actions, we then have a mixed strategy.[5]

Best Response

In the domain of multi-agent systems the strategy chosen by an agent that maximizes its utility given the strategies of the other agents is called “Best Response”. For example in a two player game, strategy s1* is a best response for player 1 to player’s 2 startegy s2 if it maximizes player1’s utility:

$$u1(s1*, s2) \geq u1(s1, s2)$$

Equilibria

In a normal form game, the agent aims at learning an optimal strategy. An optimal strategy can be thought as the selection of actions that maximize the payoff of the agent. But what happens in more complex games, where multiple agents take part in the same environment, and they all try to maximize their payoff? For multiple agents, one optimal strategy cannot be defined, since each agent supports their one optimal strategy that may conflict with others. For this reason, in complex games like this, we define a subset of outcomes, called solution concepts, from multiple candidate strategies of interest. [3],[5]

Solution concepts capture different aspects of strategic behavior, stability, or optimality, providing a structured framework that help us understand when optimal outcome has been reached. One of the most fundamental solution concepts is the Pareto optimal equilibrium and Nash equilibrium. For our experiments, we are only interested in the concept of Nash equilibrium.[3],[5]

Nash Equilibrium

Without a doubt one of the most crucial concepts in game theory, and consequently in the analysis of multiagent systems, is that of equilibrium, particularly Nash equilibrium. Nash equilibrium refers to a state in which no agent can improve their outcome by unilaterally changing their strategy, assuming the other agents' strategies remain the same. For instance, if agents i and j are engaged in a Nash equilibrium with strategies s1 and s2 respectively, any deviation from these strategies will lead to suboptimal outcomes for the deviating agent. Despite its significance, the presence of Nash equilibrium doesn't guarantee an easy resolution to decision-making. There are instances where no Nash equilibrium exists for a given interaction scenario, and in other cases, multiple Nash equilibria may coexist. Nevertheless, Nash equilibrium remains a crucial concept in understanding and analyzing multiagent systems.[3],[5]

Nash equilibria can be categorized into two distinct types: strict and weak, each revealing different degrees of stability in strategic interactions. In a strict Nash equilibrium, every player's strategy represents a unique best response to the strategies chosen by the other players. This implies that no player has an incentive to unilaterally deviate from their strategy, as doing so would only result in a less favorable outcome for them. On the other hand, weak Nash equilibria allow for the possibility that at least one player has a best response to the strategies of others that differs from their equilibrium

strategy. Consequently, weak Nash equilibria are considered less stable than their strict counterparts, as they permit deviations that may lead to suboptimal outcomes for certain players.[3],[5]

It is important to note that mixed-strategy Nash equilibria always fall into the category of weak equilibria, as they involve players randomly selecting strategies with specific probabilities. In contrast, pure-strategy Nash equilibria can vary in their classification, depending on the nature of the game. Some pure-strategy Nash equilibria may qualify as strict, where each player's strategy is indeed their best response to the strategies chosen by others, while others may be categorized as weak if there exists at least one player with a non-equilibrium best response. This distinction between strict and weak Nash equilibria provides valuable insights into the stability and dynamics of strategic interactions in various game scenarios.[3],[5]

Let's assume the player i knows the strategy of their opponent j. Then, since the agent is utility-maximizing, their optimal strategy is easy to decide, since it will be the one that has the best utility given the opponent's strategy. This can be described as:

Player i's best strategy $s_i^* \in S_i$ given knowledge of all agents' -i strategies s-i is the one so that: $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})$, for all $s_i \in S_i$

This is the best response strategy.

In a more realistic game, the agent does not know the opponent's move, thus best response strategy cannot be applied. Instead, we can define an outcome of interest for such games using the solution concept Nash equilibrium.

Theorem 1.1: A strategy profile s = (s1,... ,sn) is a Nash equilibrium if, for all agents i, si is a best response to s−i. In other words, a game has a Nash equilibrium, if for a strategy s1 that agent i plays, agent j has no benefit from playing something different than strategy s2. And likewise, if agent j has played s2, i has no benefit of playing any strategy other than s1.

Overall, an agent can receive no incremental benefit from changing their current strategy, if other players remain constant in their strategies. This leads to a state of stability, since no agent has motive to deviate from their current strategy.

Theorem 1.2: A strict Nash equilibrium is defined, where for all agents i, the strategy si gives max utility $u_i(s_i, s_{-i}) > u_i(s'_i, s_{-i})$ than all other strategies $s_i \neq s'_i$.

Theorem 1.3: A weak Nash equilibrium is defined as a non strict Nash equilibrium, where for all agents i, the strategy si gives max utility $u_i(s_i, s_{-i}) \quad u_i(s'_i, s_{-i})$ than all other strategies $s_i \neq s'_i$.

It is also proved that a pure strategy Nash equilibrium can be weak or strict, but a mixed strategy Nash equilibrium is weak.[3]

Learning

Artificial intelligence focuses on engineering agents that are capable of operating in unknown environments and undergo changes while the agent is in the learning process. A diverse array of techniques has been developed, leading to the advancement of increasingly sophisticated learning rules. An extra layer of complexity is added in multi-agent systems where different agents coexist in the same environment. In that case, the presence of the other agents, alters the learning process and the environment for the protagonist agent. Each agent learning concurrently results in each learning rule creating a dynamic system which in turn results in generating intricate behaviors within

the system. In the domain of multiagent systems, the concepts of learning and teaching are intertwined, when deciding on a course of action, an agent must consider not only what it has learned from the past behaviors of other agents but also how it aims to influence their behaviors.[3]

Fictitious Play

Fictitious play is one of the earliest learning rules and it was initially proposed as an iterative method for computing Nash equilibria in zero-sum games. In spite of being less effective for this purpose, its intuitive update rule led to its interpretation as a simplistic learning model. Fictitious play is classified as model-based learning, where the learner maintains beliefs about the opponent's strategy.[3]

In fictitious play, the agent focuses on their own payoff matrix, disregarding the payoffs of the opponent and they are only concerned with their own actions. Additionally, in this approach, in each stage game, the agent maintains statistics about the opponent's actions, assuming that the opponent follows a mixed strategy based on the empirical distribution of their previous actions. This assumption of stationarity arises because the agent aims to learn the strategy for the stage game, rather than the opponent's stochastic game strategy.[3],[6]

For every action a in opponents set A, it maintains $w(a)$: the number of times or frequency that action has been played. The empirical distribution is provided by:

$$P(a) = \frac{w(a)}{\sum_{a' \in A} w(a')}.$$

Figure 1.4: Empirical probability of opponent actions a

$P(a)$ is the empirical average and it is the probability of the opponent playing action a based on the opponent's past actions.[3]

The performance of fictitious play heavily depends on the initial beliefs of the players. These initial beliefs, akin to action counts observed prior to the game's commencement, significantly influence the learning trajectory. It's crucial to select nonempty prior beliefs for each agent; a prior belief vector of all zeros is invalid as it doesn't represent a meaningful mixed strategy. Fictitious play presents a paradox in that each agent assumes the opponent follows a stationary policy, yet no agent consistently adheres to a stationary policy except when the process naturally converges to one.[3],[6]

The structure of such techniques is straightforward.

Initialize beliefs about the opponent's strategy
repeat
  Play a best response to the assessed strategy of the opponent
  Observe the opponent's actual play and update beliefs accordingly

Figure 1.5: Fictitious play algorithm[3]

Fictitious play uses best response method in selecting its next action, as seen above.

Best response, agent i receives a mixed strategy $si*$, which gives maximum utility based on their opponents current strategy profile. If for every agent i has chosen their best response and it is unchangeable in time, then the strategy profile of all agents has reached a Nash equilibrium.[3],[6]

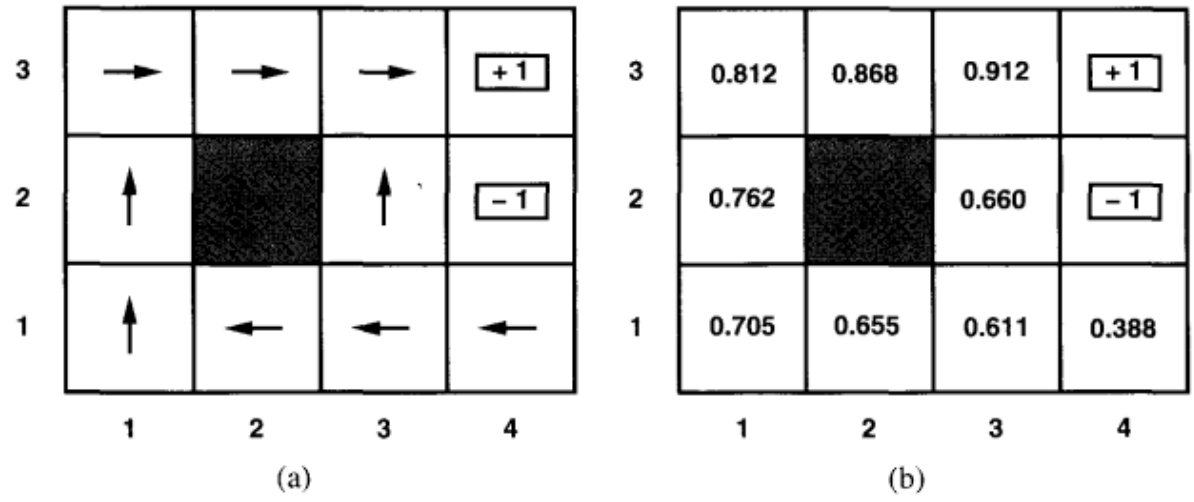
Theorem 1: If there exists a pure Nash equilibrium in a stage game, then Fictitious play can converge to it in the repeated game.

Theorem 2: If the empirical distribution of each player's strategies converges in fictitious play, then it converges to a Nash equilibrium

Theorem 3: If the game is zero-sum, then the empirical frequencies of play will converge in fictitious play

Theorem 1 provides an easy way to determine from first glance if an equilibrium exists in a game. Our experiments later focus on repeated games and show that even in non one-shot games, there is still noticeable convergence to empirical distribution of the actions. Based on Theorem 2, we will conduct our experiments in the following chapter.

Markov Decision Process



Markov decision process is the process of determining an optimal policy in a stochastic environment with a well defined transition model. Decision problems are frequently categorized into

"Markov" and "non-Markov" scenarios. Specifically, the Markov property is deemed to be upheld if the transition probabilities from any given state are contingent solely upon the current state and not influenced by past events. Studying algorithms designed to compute optimal policies within Markov decision problems offer invaluable insights into navigating complex environments with uncertain outcomes.[1]

Reinforcement learning

Reinforcement learning is a learning technique in which an agent learns in an environment by trial and error using reward as feedback from its actions. Reinforcement learning is similar to MDP and in the case of multi agent systems, stochastic games can be implemented. To further explore the principles of reinforcement learning, it is essential to study the learning task:

Accessibility of the Environment: Environments can be categorized as either accessible or inaccessible. In an accessible environment, states are directly correlated with percepts, simplifying the agent's task. Conversely, in an inaccessible environment, the agent must internally maintain a state representation to comprehend and navigate the environment effectively.

Initial Knowledge and Learning: Agents may commence their journey with prior knowledge of the environment and the consequences of their actions. Alternatively, they might need to acquire this knowledge gradually through learning, encompassing both the environment model and utility information.

Distribution of Rewards: Rewards can manifest in different ways, either confined to terminal states or distributed across all states. Moreover, rewards can serve as direct components of the utility function, such as points in a game or monetary gains in a financial setting, or they can serve as subtle indicators guiding the agent's actions.

Passive vs. Active Learning: Agents can adopt either passive or active learning approaches. Passive learners observe the environment without intervention, focusing solely on learning state utilities. In contrast, active learners leverage acquired knowledge to take actions and explore uncharted territories, often employing problem generation to drive exploration strategies.

Given that agents are rewarded based on utilities, two primary design approaches merit consideration. The first involves learning utility functions, where agents strive to acquire knowledge of utility across various states or state histories. By comprehending the utility associated with different states, agents can make informed decisions aimed at maximizing expected utility. This approach underscores the significance of understanding the long-term consequences of actions for achieving optimal outcomes. Alternatively, agents can opt for Action-Value Function Learning, commonly known as Q-learning. This strategy focuses on learning an action-value function to estimate the expected utility of taking specific actions in given states. Through iterative updates of action values during interactions with the environment, agents enhance their decision-making abilities to optimize overall utility. Each design paradigm presents unique advantages and trade-offs, catering to diverse learning contexts and goals. The selection between learning utility functions and action-value functions hinges on factors such as environment complexity, data availability, and desired decision-making granularity.[1]

Reinforcement learning in zero-sum stochastic games

Hannan-consistent for an agent in a stochastic game against opponents playing stationary policies. However, against opponents using more complex strategies, such as Q-learning itself, we do not obtain such a guarantee. The above approach, assuming away the opponent, seems unmotivated. Instead, if the agent is aware of what actions its opponent selected at each point in its history, we can use a modified Q-function, Qπ i : S×A 7→ R, defined over states and action profiles, where A = A1 × A2. The formula to update Q is simple to modify and would be the following for a two-player game.[3]

Now that the actions range over both our agent's actions and that of its competitor, how can we calculate the value of a state? Recall that for (two-player) zero-sum games, the policy profile where each agent plays its maxmin strategy forms a Nash equilibrium. The payoff to the first agent (and thus the negative of the payoff to the value of a second agent) is called the the game, and it forms the basis for our revised value function for Q-learning.[3]

Q-Learning

The core idea behind Q-Learning is to estimate the unknown transition probabilities by leveraging the actual distribution of states encountered during gameplay. However, this approach leaves significant flexibility in determining the action selection sequence within the algorithm. It is important to note that while this theorem establishes the validity of the method, it does not provide insights into the convergence rate. Moreover, it offers no guarantees regarding the agent's ability to accumulate optimal future discounted rewards. Depending on the discount factor, the agent may incur high costs before converging to the optimal policy, making it challenging to recoup these losses in the future. This concern is mitigated when learning occurs during training sessions, with the agent being deployed into the real world only after sufficient convergence.[2],[3]

Initialize the Q-function and V values (arbitrarily, for example)
repeat *until convergence*
  Observe the current state s_t.
  Select action a_t and and take it.
  Observe the reward $r(s_t, a_t)$
  Perform the following updates (and do not update any other Q-values):
  $Q_{t+1}(s_t, a_t) \leftarrow (1 - \alpha)Q_t(s_t, a_t) + \alpha_t(r(s_t, a_t) + \beta V_t(s_{t+1}))$
  $V_{t+1}(s) \leftarrow \max_a Q_t(s, a)$

Figure 2.1 Q-Learning algorithm

Minimax Q-Learning algorithm

Similar to the foundational Q-learning algorithm, the described minimax-Q learning algorithm is ensured to converge as the number of samples for each state-action pair approaches infinity. This convergence guarantees the agent a payoff that is at least equivalent to that of its maxmin strategy. However, it deviates from Hannan consistency, which implies that if the opponent adopts a suboptimal strategy, minimax-Q may struggle to exploit it effectively across various games. While it offers robust convergence properties, this approach may exhibit limitations in scenarios

where adversaries employ suboptimal strategies, thereby impacting the agent's ability to capitalize on potential advantages in gameplay dynamics.[2],[3]

The minimax-Q algorithm not only prescribes updates for Q and V values but also outlines a strategy for updating the Π strategy. While the algorithm provides guidance on these updates, certain parameters remain flexible, such as the adjustment of the learning parameter, α. One approach to updating α involves employing a decay rate, where α is reduced to α * decay after each Q-value update, with decay being a value less than 1. Alternatively, inspired by techniques from the Q-learning literature, another option is to maintain separate α values for each state-action pair. In this scenario, a common method is to set α = 1/k, where k represents the total number of updates for that specific Q-value, including the current update. For instance, when encountering a reward for the first time in state s with action profile a, the Q-value is updated to reflect the observed reward plus the discounted value of the subsequent state (α = 1). Subsequent encounters with the same state-action pair result in updates where the new Q-value is a blend of half of the previous value and half of the new reward and discounted successor state value. This adaptive adjustment of α ensures that learning rates vary across different state-action pairs, allowing the algorithm to dynamically adapt its learning process based on the frequency of encounters with specific scenarios.[2],[3]

// Initialize:
forall $s \in S, a \in A,$ *and* $o \in O$ **do**
  $Q(s, a, o) \leftarrow 1$
forall s *in* S **do**
  $V(s) \leftarrow 1$
forall $s \in S$ *and* $a \in A$ **do**
  $\Pi(s, a) \leftarrow 1/|A|$
$\alpha \leftarrow 1.0$
// Take an action:
when in state s, with probability $explor$ choose an action uniformly at random,
and with probability $(1 - explor)$ choose action a with probability $\Pi(s, a)$
// Learn:
after receiving reward rew for moving from state s to s' via action a and
opponent's action o
$Q(s, a, o) \leftarrow (1 - \alpha) * Q(s, a, o) + \alpha * (rew + \gamma * V(s'))$
$\Pi(s, \cdot) \leftarrow \arg\max_{\Pi'(s,\cdot)}(\min_{o'} \sum_{a'}(\Pi(s, a') * Q(s, a', o')))$
// The above can be done, for example, by linear programming
$V(s) \leftarrow \min_{o'}(\sum_{a'}(\Pi(s, a') * Q(s, a', o')))$
Update α

Figure 2.2 MiniMax Q-learning algorithm [3]

In Q-Learning, if the agent knows its current state and their reward, they are guaranteed to converge to Q, V values of the optimal policy.

Q-learning can be used to solve the multi-agent Nash equilibrium with the historical information of itself and its opponent. For (two-player) zero-sum games, the policy profile where each agent plays its maxmin strategy forms a Nash equilibrium. The payoff to the first agent (and thus the negative of the payoff to the second agent) is called the value of the game.

Computing equilibria: There is no formula in computing a Nash equilibrium. It is empiricaly computed by the convergence of the mixed strategies of each agent in a finite repeated game.

In the following sections, we have developed some experiments in Nash equilibrium computation in different types of repeated games. In these experiments, we monitor the mixed strategies of each agent. Our results show if there is convergence to each agent's empirical action probabilities and check if it reached a Nash equilibrium based on different types of games and different agent learning methods with fictitious play or MiniMax Qlearning.[3]

Experiments

For our experiments, we attempted to run various repeated game types with different payoff matrices for two-player fictitious play agents, two-player MiniMax Q-learning agents and a combination of a twoplayer fictitious play and MiniMax Q-learning agents. The games we experimented were Matching Pennies, Battle of Sexes, Prisoner's Dillema and Rock, Papers scissors. The main difference between these games are based on how cooperative and competitive they are and how many different types of equilibria they have with each other. As far as the configuration goes, the default training on the MiniMax Q-Learning agent was with an ε-greedy method with exploration ε = 0.3, learning rate a = 1 and discount factor γ = 0.9, where the learning rate is not updated at each iteration, but in every 100 episodes. We will also present training scenarios with a change of these parameters. Finally, we also experimented with the number of players, in case this was also a factor to the convergence.

1. Matching Pennies

	Heads	Tails
Heads	$1, -1$	$-1, 1$
Tails	$-1, 1$	$1, -1$

Figure 3.1: Payoff matrix for Matching pennies 2-player game.

As seen by the player's payoff matrix in Figure 3.1, the Matching Pennies is a zero-sum game. There is no pure Nash equilibrium, but there exists one mixed Nash equilibrium where the mixed strategy of each player is to play Heads or Tails with probability (0.5, 0.5). We will show this with the experiment.

For our first experiment, we tried implementing fictitious play and Q-learning in the popular Matching Pennies game. The version we implemented it on was for a two-player repeated game. We initialize each player's beliefs as (1.5, 2) and (2, 1.5) meaning that player1 played Heads 2 times and Tails 1.5 times and player 2 played Heads 1.5 times and Tails 2 times. After some iterations, the player empirical action count is seen below:

Round	1's action	2's action	1's beliefs	2's beliefs
0			(1.5,2)	(2,1.5)
1	T	T	(1.5,3)	(2,2.5)
2	T	H	(2.5,3)	(2,3.5)
3	T	H	(3.5,3)	(2,4.5)
4	H	H	(4.5,3)	(3,4.5)
5	H	H	(5.5,3)	(4,4.5)
6	H	H	(6.5,3)	(5,4.5)
7	H	T	(6.5,4)	(6,4.5)
⋮	⋮	⋮	⋮	⋮

Figure 3.2: Empirical action profile of 2-players Matching Pennies.

FP vs FP: From the above Figure 3.2, we can see that the stage strategy of the players does not actually converge. In fact, in each stage game each agent changes their strategies back and forth, leading to no convergence on a steady state and therefore no existence of pure strategy. However, even if that is the case, the empirical distribution of the stage game strategies over multiple iterations does converge to the (unique) mixed strategy profile (0.5, 0.5). This is apparent in our graph Figure 3.3, as seen that the probability of both actions for each agent converges to the steady value 0.5, even if the stage action goes back and forth.

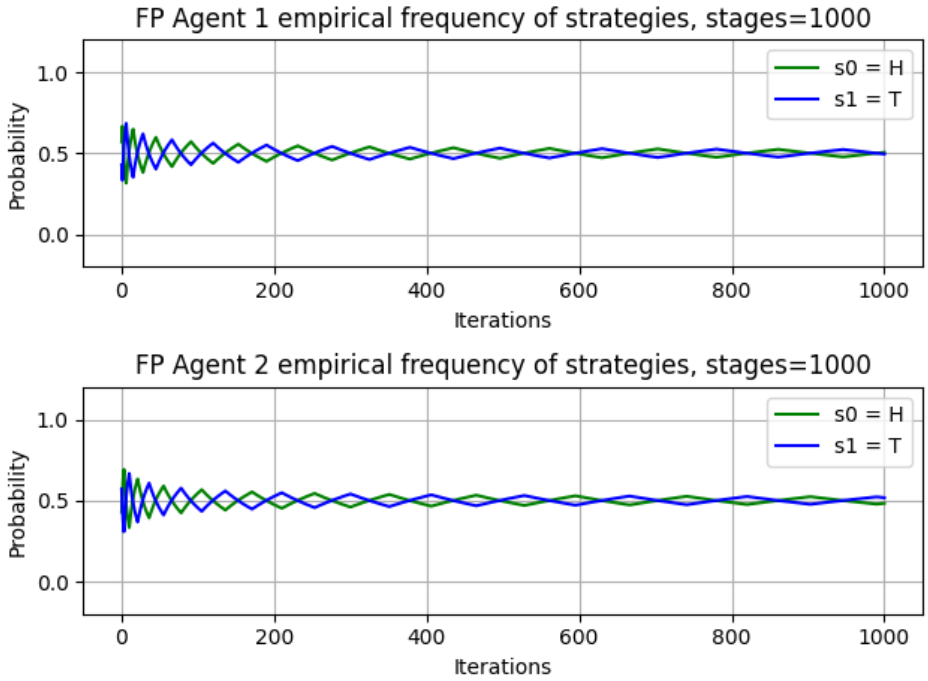


Figure 3.3: Matching Pennies (1.5, 2) and (2, 1.5) initial beliefs for 100 iterations. Mixed Nash convergence at (0.5, 0.5).

Another interesting thing we noticed, is the speed of convergence in fictitious play, where the mixed Nash equilibrium seems to be found at around the 500th iteration where it starts to smooth out to the 0.5 values. However, there is still clear oscillation even after 1000 iterations.

Another thing is that fictitious play agents can be greatly affected by their initial beliefs. For example, in Figure 3.4 we run the same experiment but this time initial beliefs are (100, 0.1) and (0.1, 100). The figure shows an apparent decline in the rate of convergence, even after 1000 iterations.

Still, it is intuitive from the figure that for more iterations, the fictitious play still converges to an equilibrium. This is to be expected since fictitious play converges for all zero-sum games.

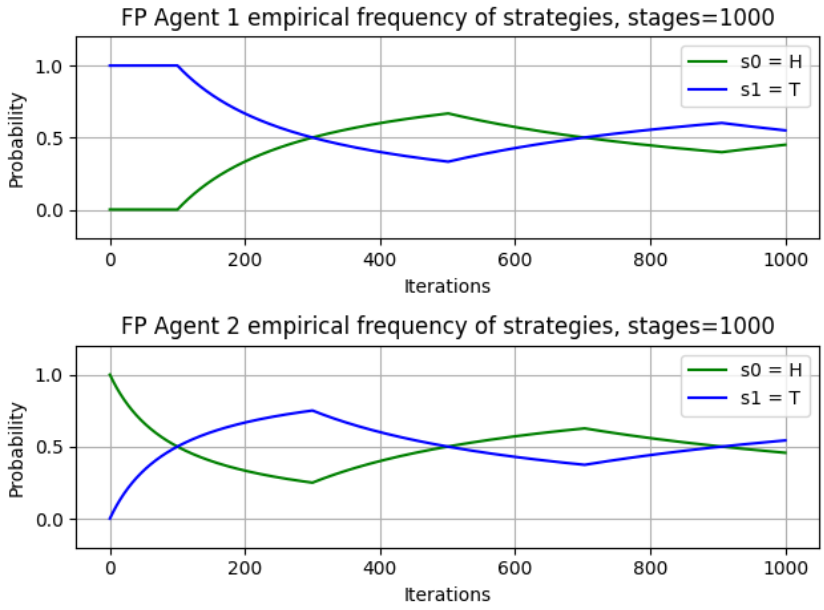


Figure 3.4. Matching Pennies with different initial beliefs slow the rate of convergence to an equilibrium

However, the convergence of the fictitious play agents is as expected, since Matching Pennies is a zero-sum game and from Theorem 2 and 3 it is expected to converge to Nash equilibrium.

But it is not certain the same can be said for RL agent. For that we apply our next experiment, which is training 2 MiniMax Q-Learning agents with default parameters to play against each other.

RL vs RL: Interesting results are also seen with two Minimax Q-learning agents. As seen in Figure 3.5, using RL parameters ε=0.3, learning rate α = 1 and γ = 0.9 and initial policies (beliefs) (0.5, 0.5):

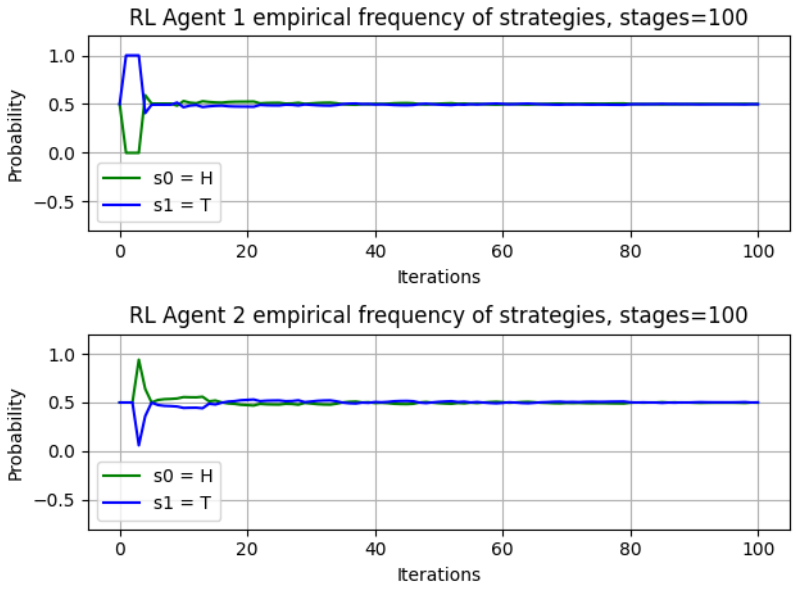
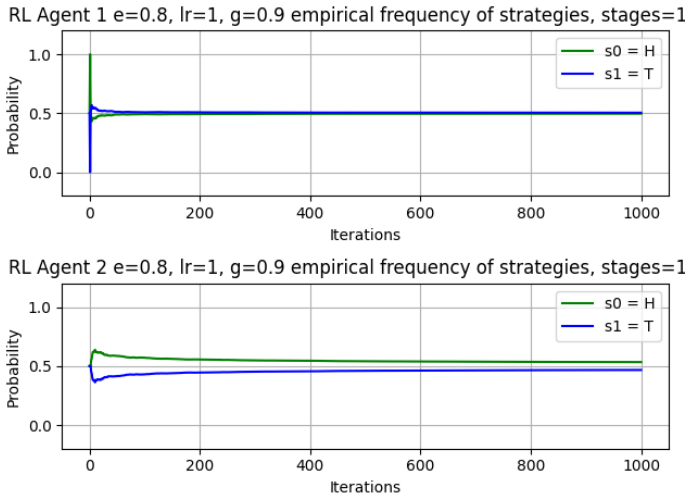


Figure 3.5: Matching Pennies for RL agents. Mixed Nash convergence at (0.5, 0.5).

It is shown, RL agent seem to not oscillate like FP agents and also generally converge to their mixed Nash equilibrium (0.5, 0.5) much faster than FP (around 18 iterations). Also changing the ε and learning rate values did not alter the convergence greatly, since it was able to learn in approximately the same amount of time and converge to the same equilibrium, in contrast to the sensitivity that fictitious play agents displayed in Figure 3.4. Some attempts at changing the exploration rate ε, learning rate a and discount factor γ are shown in Figure 3.6.



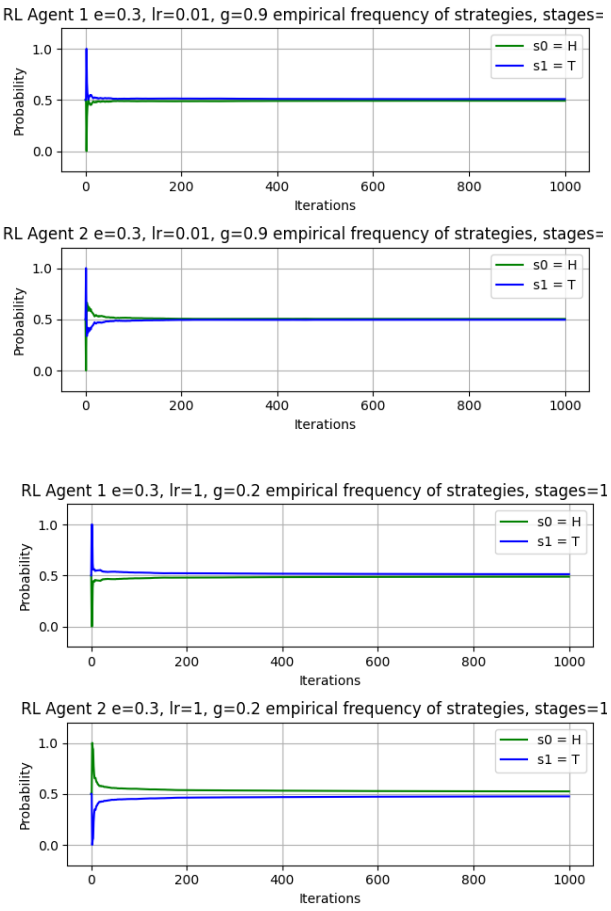


Figure 3.6: RL vs RL for different (a) epsilon value = 0.8 (b) learning rate = 0.01 and (c) gamma = 0.2 Matching Pennies.

It is shown, RL agent seem to not oscillate like FP agents and also generally converge to their mixed Nash equilibrium (0.5, 0.5) much faster than FP (around 18 iterations). Also changing the ε, learning rate and discount factor initial values can affect the rate of convergence slightly as seen in Figure 3.6, in contrast to the sensitivity that fictitious play agents displayed in Figure 3.4. Changing the initial policies to (1.0, 0) and (0, 1.0) also does not affect the convergence to the same Nash equilibrium seen in Figure 3.7.

However, one disadvantage against FP agents is that RL is much more unstable in the initial episodes and oscillates a lot more, whereas FP agents are smoother in convergence and have an obvious pattern.

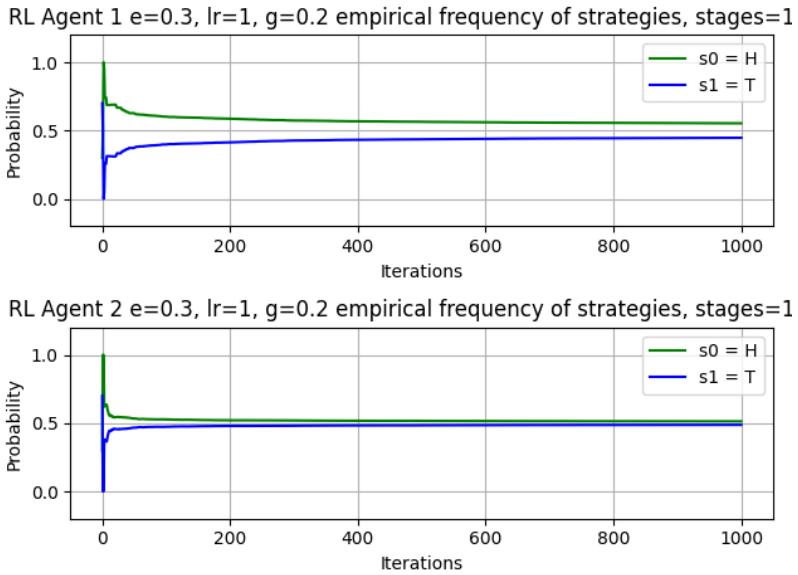


Figure 3.7: RL vs RL different policies (0.3, 0.7) Matching Pennies.

RL vs FP: In order to see if the strategies of different learning methods affect one another the same way, we experimented on games with one agent being RL agent and another FP. What we want to notice is any effect in the final convergence to a Nash equilibrium – if any!

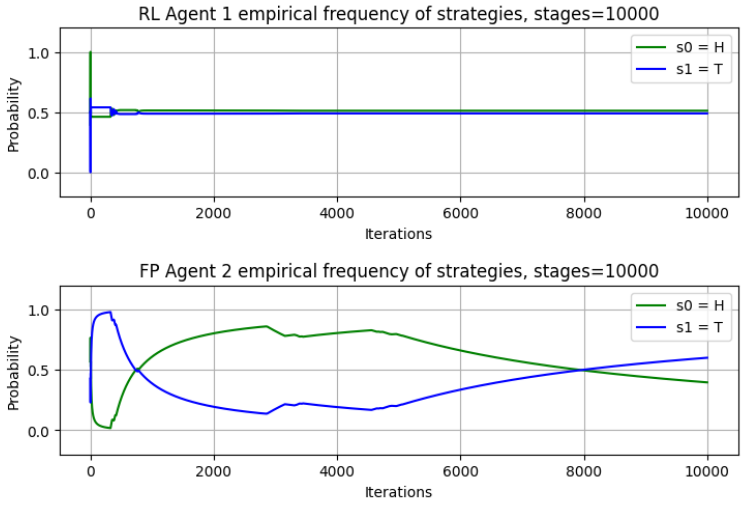


Figure 3.8: RL vs FP Default parameters and beliefs, no convergence Matching Pennies

Even for 10000 iterations, FP shows no clear indication of converging to the mixed Nash equilibrium. While RL agent also struggles to converge at first, it manages to find the mixed strategy (0.5, 0.5) much faster than the FP agent. Still, there is no apparent mixed Nash equilibrium convergence on (0.5, 0.5), even if both agents playing by one of their own kind did converge.

Another interesting thing we noticed is that the pattern of convergence changes without change in the initial parameters or beliefs. An example of running the same game with the same configurations but not deterministic outcome is seen in Figure 3.9.

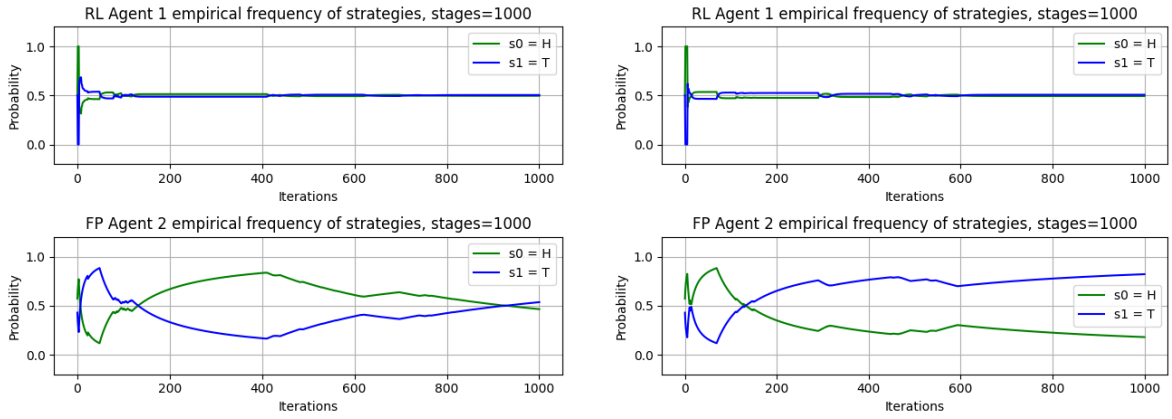


Figure 3.9: RL vs FP different convergence pattern with same beliefs. Still no convergence to mixed Nash equilibrium Matching Pennies

2. Prisoner's Dilemma

	B	S
B	-1, -1	-3, 0
S	0, -3	-2, -2

Figure 3.10: Prisoner's Dilemma payoff matrix

Going to a less competitive game and rather more compromising, the Prisoner's Dilemma game. Its payoff matrix can be seen in Figure 3.10. Each player should choose either to stay silent and not betray their opponent or choose to betray them. There is one pure Nash equilibria in the strategy (Betray, Betray), and in this strategy profile no agent has the motive to change their action.

FP vs FP:

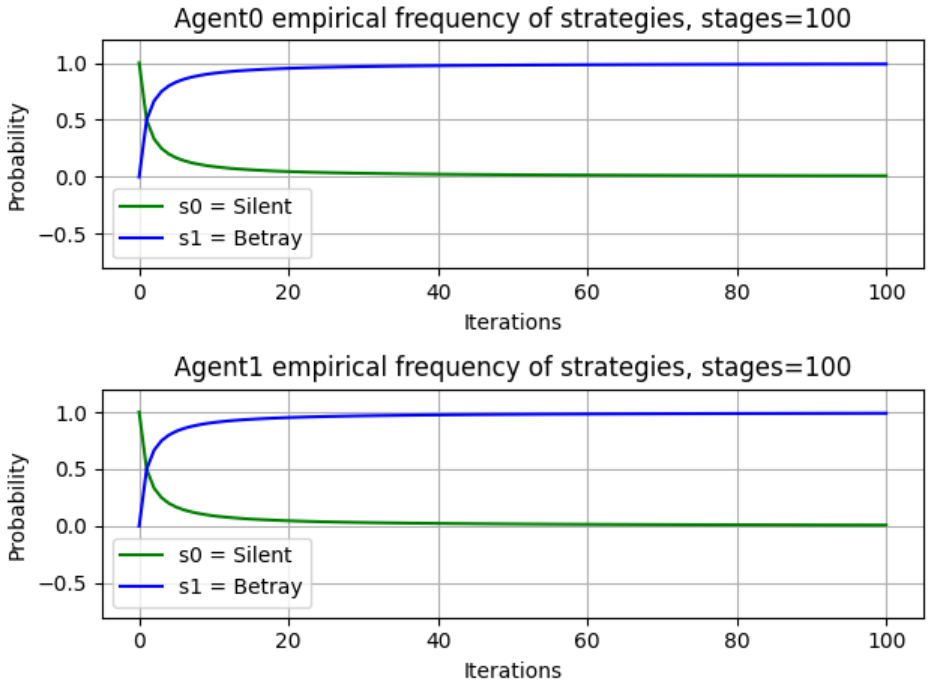


Figure 3.11: FP vs FP initial beliefs (1, 0) (1, 0) Prisoner's Dilemma.

What is taken from the above is that agents manage to converge to the unique pure Nash equilibrium (Betray, Betray) in a very small amount of time. The convergence is also much smoother compared to the Matching Pennies game, which was purely competitive.

Running again with stronger initial beliefs (1000, 0) (0, 1) gives the result in Figure 3.12.

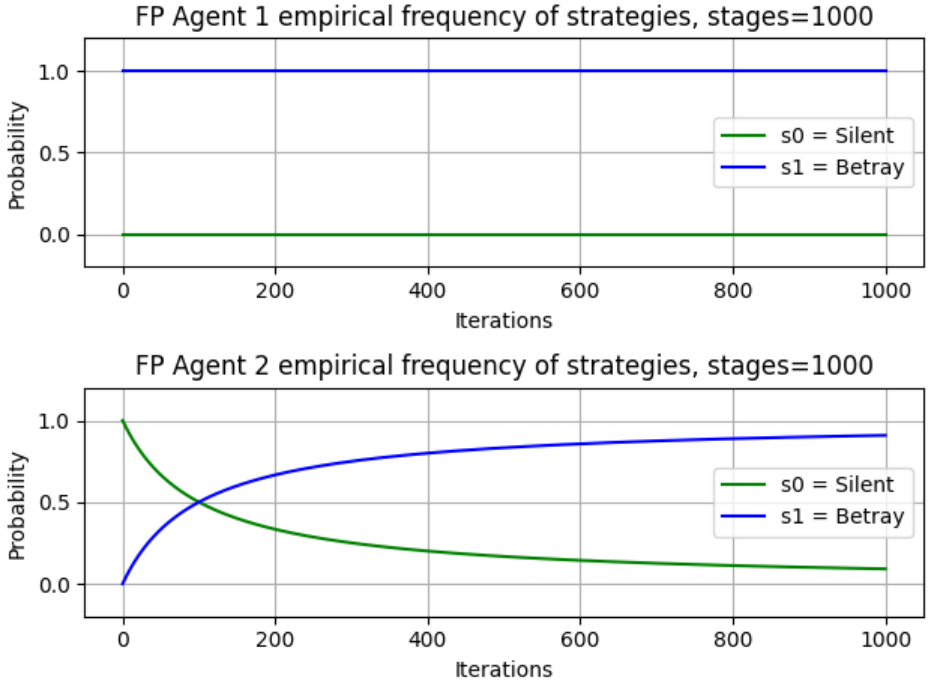


Figure 3.12: FP vs FP Strong initial beliefs (100, 0) and (0, 1) Prisoner's Dilemma

It is apparent that the game still converges to a Nash equilibrium, but the sensitivity of FP players is still evident by the fact that the convergence rate is slower (noticeable in player 2). Nevertheless, the convergence to the Nash equilibrium is almost immediate.

RL vs RL:

In this game, we noticed that the RL agent is way more unstable in converging than FP. The agent oscillates their strategies quite often and for 1000 iterations seems still to not be able to converge to the pure strategy as seen in Figure 3.13. The parameters used again are the default ones (ε=0.3, a=1, υ=0.9) and initial policies (0.5, 0.5).

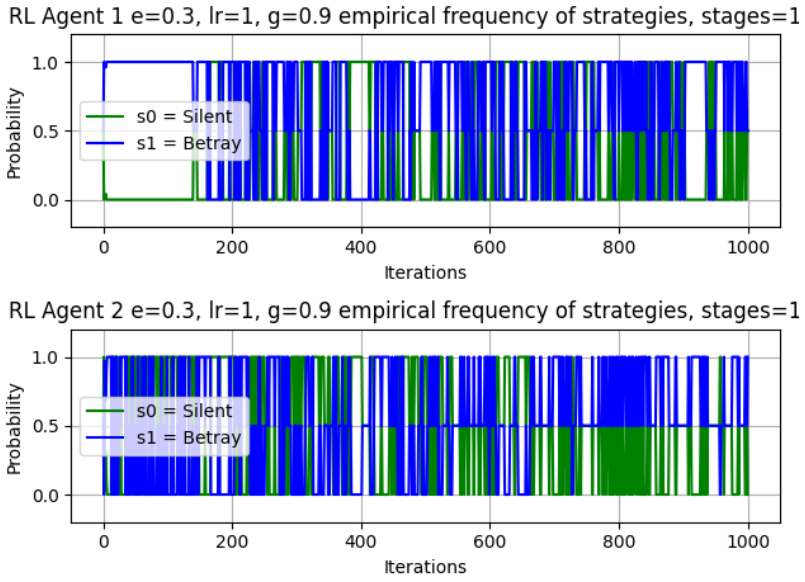


Figure 3.12: RL vs RL initial default params no convergence Prisoner's Dilemma

However, changing its parameters to ε=0.8, a=1 and γ=0.2 gives an immediate convergence. Also, a more stable result was noticed with just changing the exploration variable ε=0.8, seen in Figure 3.14.

Even if initially the RL agents cannot converge, from our experiments it is apparent that by tuning their parameters they eventually manage to converge to the pure Nash equilibrium.

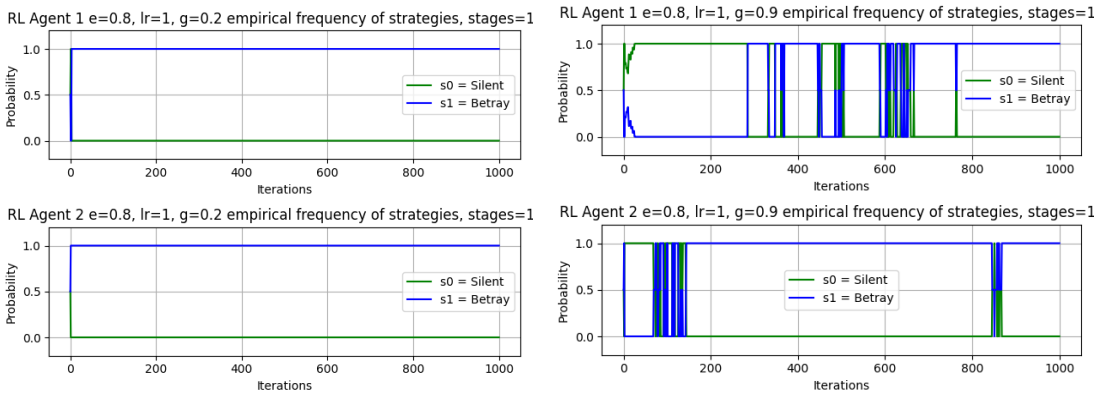


Figure 3.14: RL vs RL initial beliefs (0.5 0.5) Prisoner's Dilemma

RL vs FP:

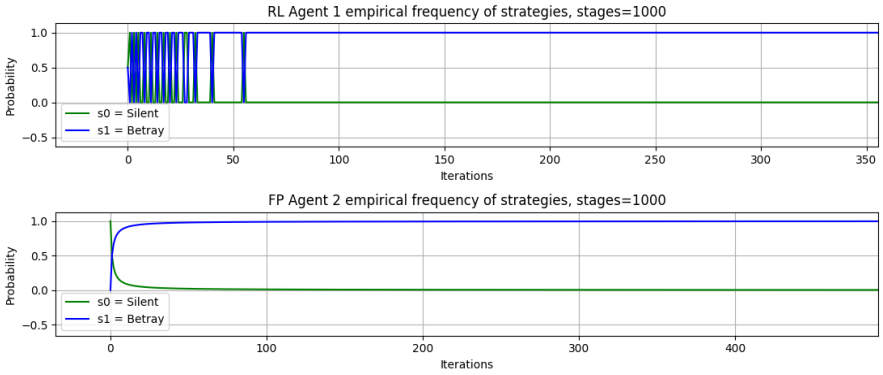


Figure 3.15: RL vs FP initial beliefs (0.5 0.5) (1 0) default params Prisoner's Dilemma

The RL agent is using its default initial parameters.

From the Figure 3.15, the rate of convergence is much faster for RL agent with its default params and contrary to our initial beliefs that showed RL could only converge to the Nash equilibrium with tinkering its parameters, now both row and column players are able to converge almost immediately to the 1 pure Nash (Betray, Betray).

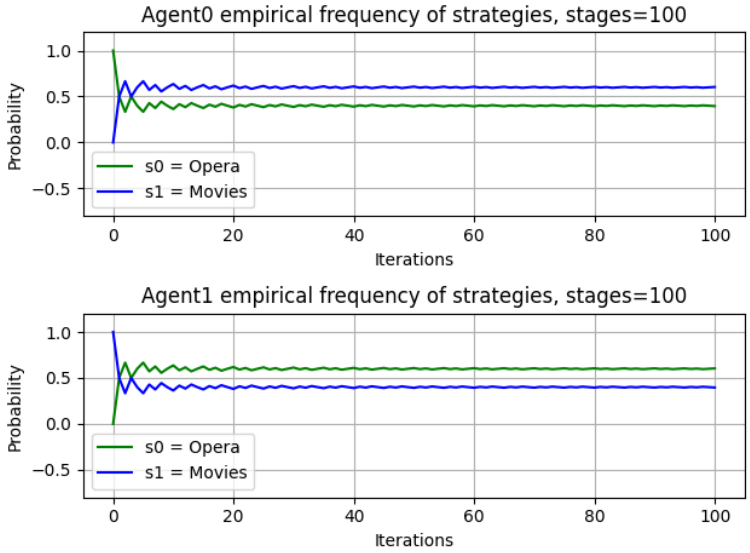
3. Battle of Sexes

Battle of sexes is again a compromising game its payoff matrix of which can be seen in Figure 3.16. This game has 2 pure and 1 mixed Nash equilibria. The 2 pure are in the options (Opera, Opera) or (Movies, Movies). There is also a chance, depending on how the agent is initialized, that we might converge to the mixed Nash equilibria (3/5, 2/5). Its variety in Nash equilibria make this game ideal for our experiments, especially to test the sensitivity of the learning algorithms to different beliefs in relation to the final equilibria computed.

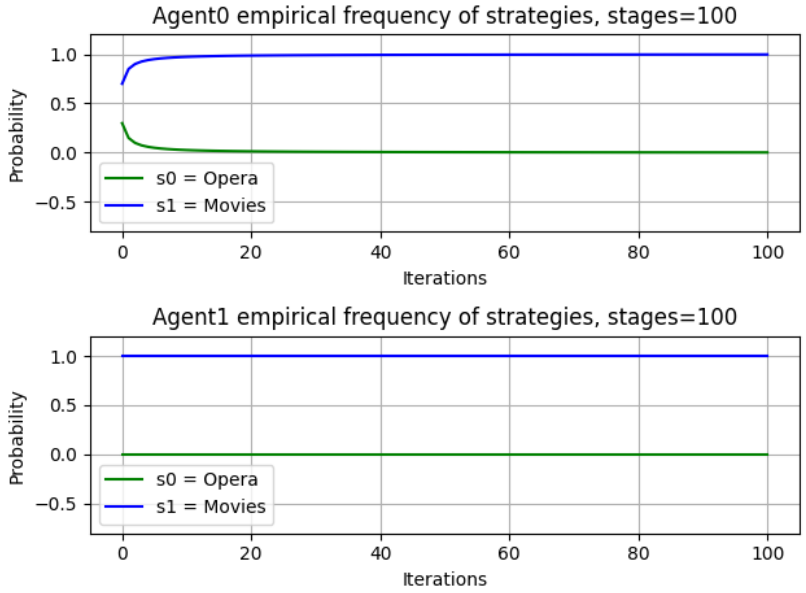
FP vs FP:

By the figure we can see that again changing the initial beliefs tends to affect the FP agents to go towards a different Nash equilibrium. For example, initial strong beliefs (0, 100) (0.3, 0.7) converge to one of the pure Nash equilibrium (Movies, Movies) (Figure 3.17), wereas changing the initial beliefs to (0, 1) (1,0) leads to the convergence to the mixed Nash equilibrium (Figure 3.16).

Both row and column player converge quite fast and smooth.



3.16: Battle of Sexes mixed Nash convergence, initial beliefs (0, 1) (1, 0)



3.17: Battle of Sexes pure Nash convergence, initial beliefs (0, 100) (0.3, 0.7)

RL vs RL:

In the RL vs RL case, change of the values ε, a and γ lead to different convergence rate as seen in previous games so for the sake of not repeating ourselves, we will only present the default parameter run in Figure 3.18.

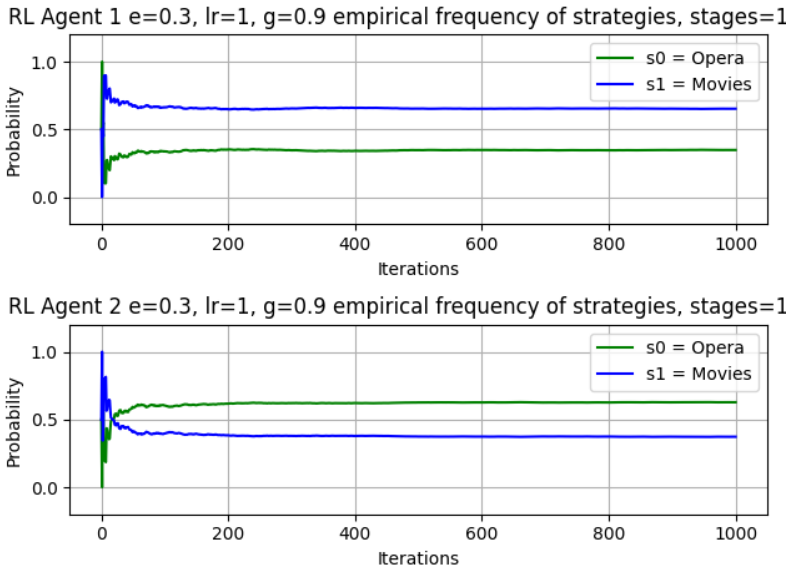


Figure 3.18: RL vs RL Initial default parameters Battle of Sexes

As it seems, the game converges to the mixed Nash equilibrium. Change in initial policies also only remains to compute the mixed Nash as seen in Figure 3.19 where we tried to change the policies to (0.3, 0.8) and to strong policies (100, 0). This is interesting, since FP agents seem to be able to find all Nash equilibrium based on their initial beliefs, while RL stays stable.

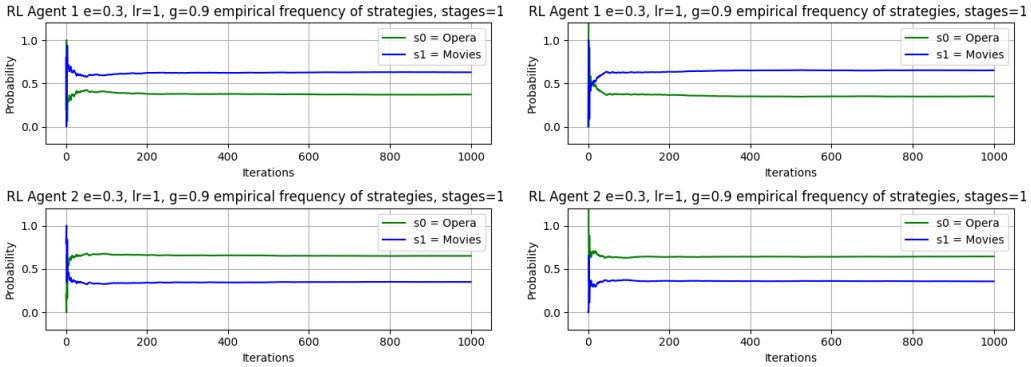


Figure 3.19: RL vs RL different policies (a) (0.2 0.8) (b) (100 0) Battle of Sexes

RL vs FP:

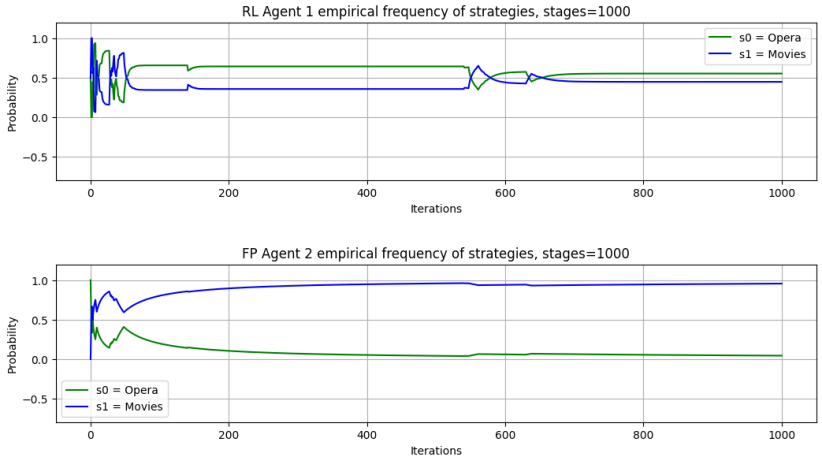


Figure 3.20: RL vs FP default initial beliefs Battle of Sexes

For the mixed agents game, there is no apparent convergence for both players, even if FP player manages to converge to the pure strategy Opera and is not motivated to change their action, the row player (RL agent) converges to a mixed strategy (3/5, 2/5) which is similar to the mixed Nash. So both players try for differentequilibria, but do not manage to converge to any of them.

4. Left and Right

Going to a fully cooperative game, the Left and Right game payoff matrix can be seen in Figure 3.21.

	R	L
R	1, 1	0, 0
L	0, 0	1, 1

Figure 3.21: Left and Right payoff matrix

Left and Right display 2 pure Nash equilibrium at the strategies (R, R) and (L, L), since no player has the motive to change their strategy and get maximum reward from these positions.

FP vs FP:

As seen from Figure 3.22, for initial mixed beliefs (0.5 0.5) we manage to converge to the pure Nash equilibrium (L, L). The convergence is almost immediate and smooth. Change of beliefs might lead to some initial oscillations for the starting stages, but the agents always end up converging to either one of the 2 pure Nash equilibria.

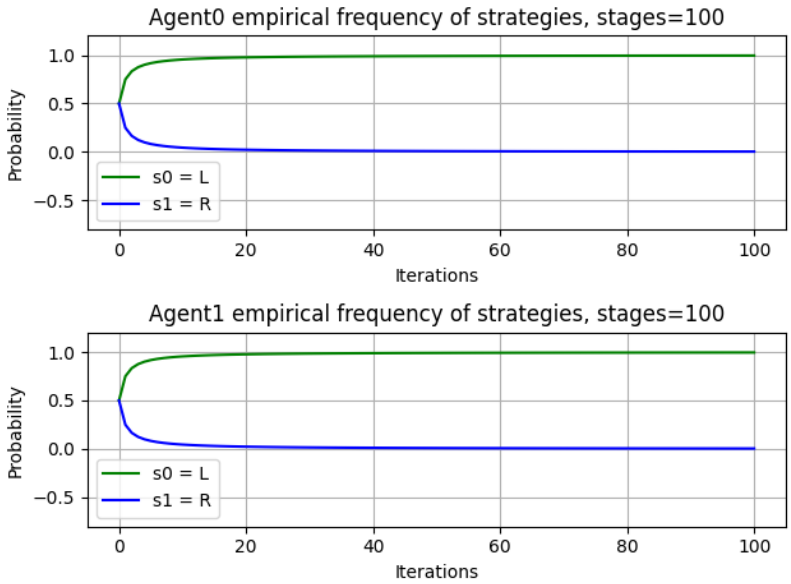


Figure 3.22: FP vs FP initial beliefs (0.5, 0.5) Left and Right

RL vs RL:

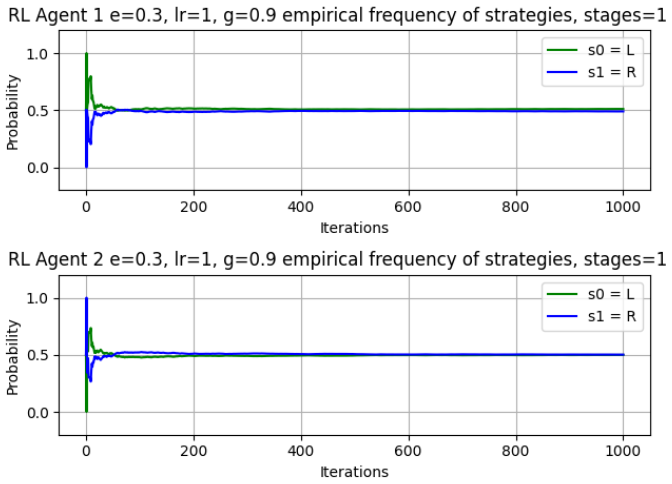


Figure 3.23: RL vs RL initial default parameters Left and Right

The result for RL vs RL is shown in Figure 3.23. In contrast to FP vs FP agents, RL struggles to converge (around 50 iterations while FP took 1). There is also a problem in computing the 2 pure Nash equilibria, since RL seems to only be able to find the mixed strategy (0.5 0.5). The reason RL might fail might be our adoption of Minimax strategy. Minimax strategy is a pessimistic strategy that suspects that their opponent is always going to play against the

agent, thus always making the agent play cautiously. However, Left and Right is a fully cooperative game and the RL agent's strategy does not seem to work here.

Let us note here that changing either of the initial parameters does not change the result, but for the sake of not tiring our readers we will not include the additional results.

RL vs FP:

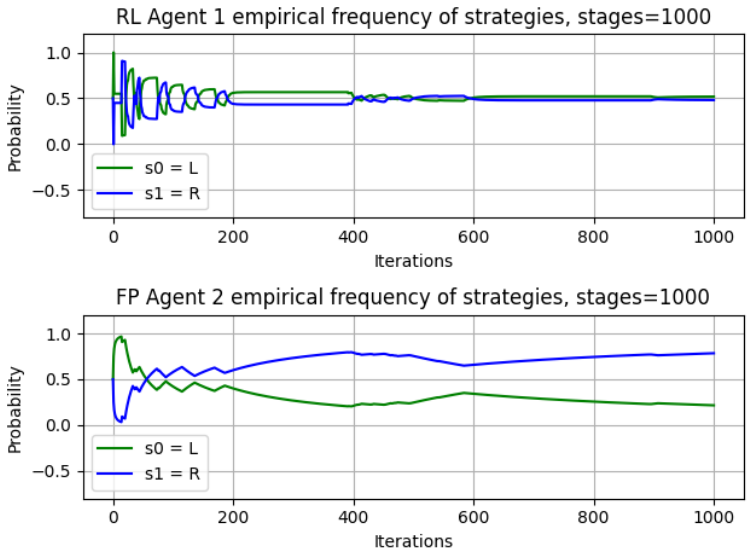


Figure 3.24: RL vs FP initial default paarmeters and beliefs (0.5 0.5) Left and Right

The rate of convergence of the FP player looks to be much slower than the previous FP vs FP results. Also, again, RL agent seems to not be able to find the pure Nash equilibrium, while FP is slowly converging. Still, no total pure Nash equilibrium is found.

5. Three-player game

In order to observe whether more players play a big role in defining the strategies of the different players, we came up with a three-player Rock-Paper-Scissor experiment. The payoff matrix is shown in Figure 3.25.

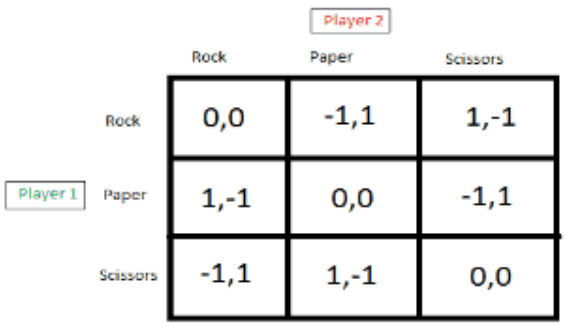


Figure 3.25: Rock-Paper-Scissor payoff matrix.

We tested both for two player environments and also for the three-player version for FP vs FP agents.

Rock-Paper-Scissor has 1 unique mixed Nash equilibrium with probability distribution (1/3, 1/3, 1/3).

Two-player FP vs FP:

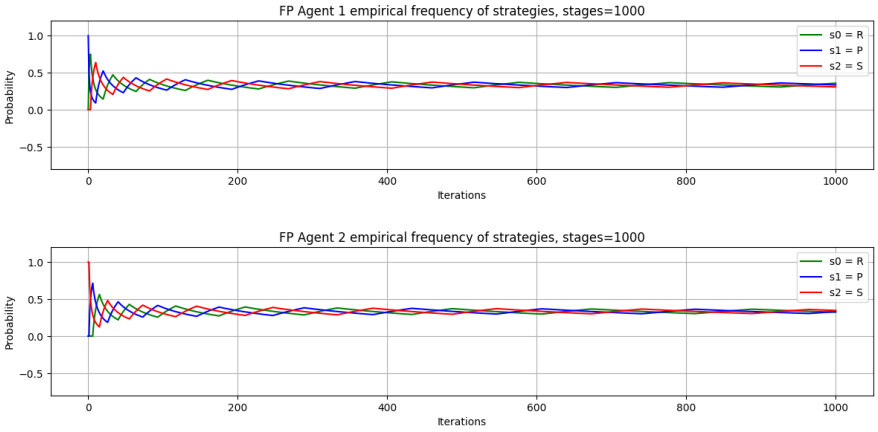


Figure 3.26: FP vs FP Initial default beliefs (1 0 0) (0 1 0) (0 0 1) RPS

Three-player FP vs FP:

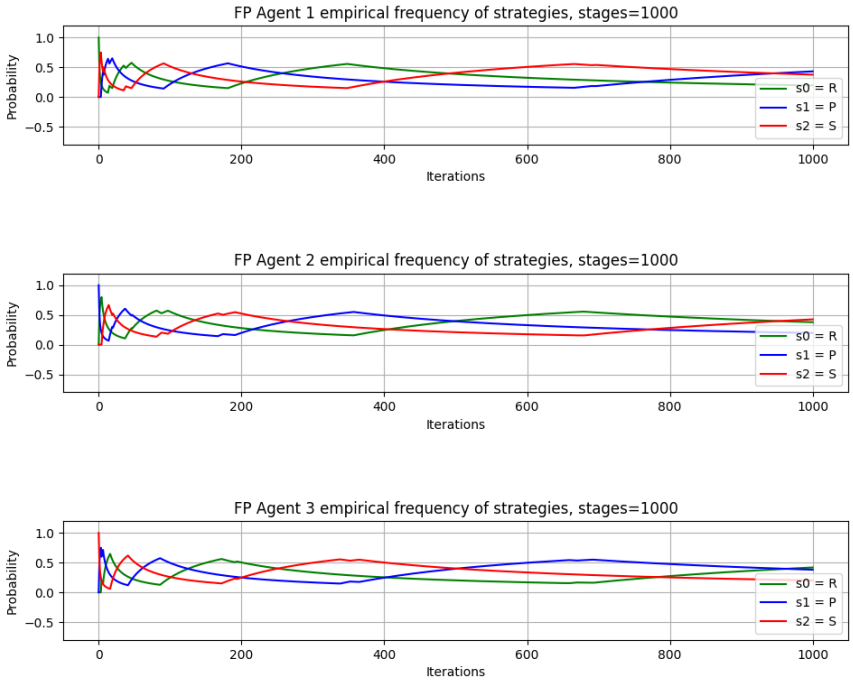


Figure 3.27: 3-player FP vs FP Initial default beliefs (1 0 0) (0 1 0) (0 0 1) RPS

Comparing the 2-player vs 3-player RPS game seen in Figures 3.26 and 3.27 respectively, we can see that the addition of a third player does not affect the convergence to a Nash equilibrium. RPS is a

zero-sum game, so Fictitious play converges eventually to the mixed Nash, even if with the 3-player version it looks like it takes a bit more time to do so.

As seen in previous two-player experiments, change in initial beliefs also change drastically the convergence rate. In Figure 3.28 the initial belief changes from (1 0 0) to (100 0 0) and it seems the rate of convergence gets much slower, but nevertheless the pattern of distribution seems to eventually end up in the Nash equilibrium (1/3 1/3 1/3) again:

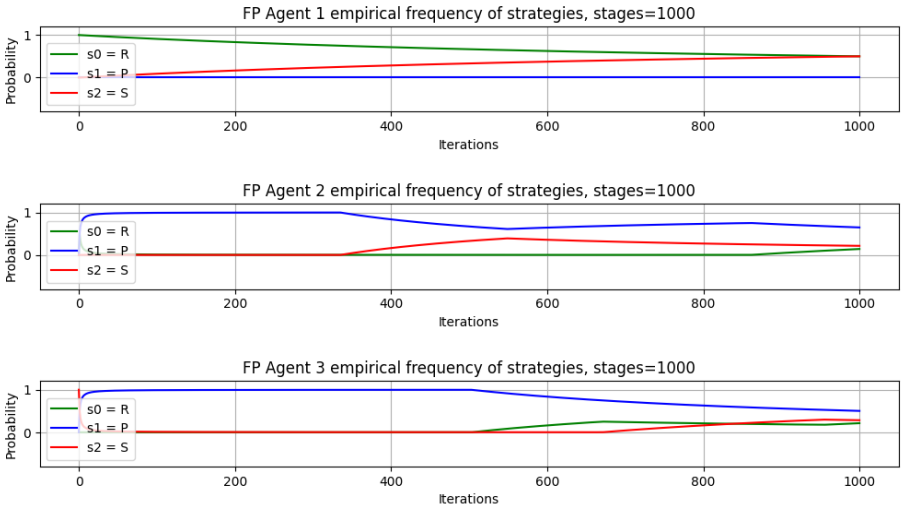


Figure 3.28: 3-player FP vs FP strong beliefs (100 0 0) (0 1 0) (0 0 1) RPS

Conclusion

From our experiments for computing Nash equilibria in repeated games for both fictitious play and minimax Q-learning agents, we noticed some things from the results. First, fictitious play always converges for zero-sum games, however it takes a lot more time for the algorithm to converge than that of a minimax-Q-learning agent. In general, the convergence of RL agents was smoother and quicker most of the time. Another noticeable thing is that RL agents were less affected in finding a Nash equilibrium by their initial policies than FP agents, who in a biased way converged to the Nash equilibrium closer to their initial beliefs. RL agents always converged to the same mixed equilibrium even with different initial params. However, different combinations of initial parameters might slow down its convergence. FP agents were greatly affected by change in their initial belief distribution, so that it could give biased results, slow down total convergence time or even increase oscillations.

As far as FP and RL mixed agents go, we noticed that FP was greatly affected by the different strategy of their RL opponent and usually displayed different convergence patterns than if the FP agent was playing with another FP agent. Initial belief effect still holds for FP in these types of games. Nevertheless, even with FP and RL mixed games, both agents managed to converge to a Nash equilibrium. Finally in our 3-player experiment, we noticed that the behavior of the agents remained the same and all agents still converged to the same mixed Nash equilibrium that the two-players had found, with the only difference being the increase in oscillations.

All in all, RL agents are quicker, more robust and less prone to be affected by factors such as their initial policies or their opponent's strategy. Their robustness also makes them great at

converging in even stochastic environments. With that in mind, RL agents are still not ideal, since most of the time fail at finding the Nash equilibria at games that have pure Nash equilibria. That could be the nature of the pessimistic MiniMax Q-learning algorithm we decided to use in our experiments. On the other hand, FP agents are greatly biased in their Nash equilibria computation but display less oscillations at the starting iterations in comparison to RL but converge slower in more competitive games. Factors such as their opponent's strategy and beliefs can also make them slower to converge with more oscillations.

Bibliography

1. Artificial Intelligence, A Modern Approach, Stuart J. Russell and Peter Norvig, Prentice Hall, Englewood Cliffs, New Jersey 07632
2. Multiagent Systems, A Modern Approach to Distributed Modern Approach to Artificial Intelligence, Gerhard Weiss, The MIT Press, Cambridge, Massachusetts, 1999
3. MULTIAGENT SYSTEMS, Algorithmic, Game-Theoretic and Logical Foundations, Yoav Shoham, Kevin Leyton-Brown, 2008
4. An Introduction to MultiAgent Systems, Michael Wooldridge, John Wiley & Sons LTD, 2002
5. MULTIAGENT INTERACTIONS, slides from Intelligent agents and multiagent systems course, Artificial Intelligence Msc
6. Learning & Teaching, slides from Intelligent agents and multiagent systems course, Artificial Intelligence Msc