University of Piraeus and NCSR Demokritos

Intelligent Agents and Multiagent Systems

MSc in Artificial Intelligence

Equilibria Computation in Zero-Sum Games

Authors
Tatiana Boura, MTN2210
Andreas Sideras, MTN2214

Supervisor
Georgios Vouros

February 7, 2023

Contents

List of Figures

1 Abstract

In the current assignment we examine two methods for the computation of equilibria in zero sum games. We demonstrate their theoretical foundations and we test them on several games. We illustrate their behaviour regarding their convergence and we compare them to each other. The first one is the Fictitious Play, a prominent model-based learning rule and the second one is Reinforcement Learning, in a model-free approach. The structure and content of this assignment is mostly inspired from the book *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations* [4].

2 Zero-Sum Games

In Game Theory, a Zero-Sum game is a class of games that represent situations of pure competition, where one player's gain must come at the expense of the other player's. In order to formally present these concepts, a few definitions will be provided and explained.

2.1 Games in normal form

Generally, agents under reasonable assumptions about preferences, will have utility functions whose expected values they want to maximize. So, if the agent knows or somehow predicts the outcomes of the environment and their probabilities, in the case of a stochastic environment, the action choice subjected to the maximization of the expected utility is a relatively simple task. In most of the cases, though, we are studying multi-agent environments where each agent aims to maximize his expected utility and his choice affects the environment and, consequently the utilities of the other agents. For the modelling of these complex interactions between agents we define the *normal form*, i.e. strategic form.

Definition 2.1.1. *(Normal-form game) A finite, n-person normal-form game is a tuple* (N, A, u)*, where:*

- N *is a finite set of n players, indexed by* i
- $A = A_1 \times \cdots \times A_n$*, where* A_i *is a finite set of actions available to player* i*. Each vector* $a = (a_1, \ldots, a_n) \in A$ *is called an action profile.*
- $u = (u_1, ..., u_n)$ *where* $u_i : A \to \mathbb{R}$ *is a real-valued utility (or payoff) function for player* i*.*

Now we are able to define zero-sum games as normal-form games:

Definition 2.1.2. *(Constant-sum game) A two-player normal-form game* $G = (\{1, 2\}, A_1 \times A_2, (u_1, u_2))$ *is constant-sum game if there exists a constant c, such that for each strategy profile* $a \in A_1 \times A_2$*, it is the case that* $u_1(a) + u_2(a) = c$*. A* ***zero-sum game*** *is the case when* $c = 0$*.*

A classic example of a two-player zero-sum game is the problem of *Selling damaged goods* that is a business transaction between a seller and a buyer. The seller wants to sell a broken product to an unsuspecting buyer. If this transaction happens, the seller wins (by getting money) and the buyer loses (by getting nothing of value for that money).

If player **Z** chooses L:

PLAYER **X** \ PLAYER **Y**	D	C
D	-3, -3, 6	1, -5, 4
C	-5, 1, 4	0, 0, 0

If player **Z** chooses R:

PLAYER **X** \ PLAYER **Y**	D	C
D	-2, -2, 4	2, -4, 2
C	-4, 2, 2	1, 1, -2

Figure 1: Example of three player zero sum game

We mentioned earlier that a zero sum game is a game of pure competition. In order for this to hold, it is important that the game involves exactly two agents. If more than two agents take part in the game, "any game can be turned into a zero-sum game if we add a dummy player whose actions do not impact the payoffs to the other agents, and whose own payoffs are chosen to make the payoffs in each outcome sum to zero" [4]. An example from [1] of a $3-player$ zero-sum game is the following :

Example 2.1.1. *Consider a zero-sum game involving three players X, Y, Z. In this game X and Y choose between two options, D and C, whilst Z chooses between L and R. In Figure 1 is presented the payoff matrix of three players, X, Y and Z for Z's action.*

2.1.1 Strategies in normal-form game

Each agent can select an action to play given their action profile. However, in order to win, a player must develop a strategy. The notion of a strategy is quite general, and it includes pure and mixed strategies. A **pure strategy** is a strategy where the player selects a single action and plays it. The choice of a pure strategy is called a pure-strategy profile. A **mixed strategy** is another type of strategy, where players randomly select an available action according to some probability distribution. A mixed strategy of a normal-form game is defined as follows:

Definition 2.1.3. *(Mixed strategy) Let (N, A, u) be a normal-form game, and for any set X let $\Pi(X)$ be the set of all probability distributions over X. Then the set of mixed strategies for player i is $S_i = \Pi(A_i)$.*

Another interpretation of a mixed-strategy is that of a population of randomly matched individuals in the role of each player of the game, some proportion of whom make each of a number of available choices.

Also, since we are studying normal-form games where agents are utility-driven, the expected utility of a mixed strategy is presented:

Definition 2.1.4. *(Expected utility of a Mixed Strategy) The expected utility of a mixed strategy u_i for player i with the mixed-strategy profile $s = (s_1, ..., s_n)$ given a normal-form game (N, A, u) is defined as $u_i(s) = \sum_{a \in A} u_i(a) \Pi_{j=1}^{n} s_j(a_j)$.*

Note that in a single-agent non-dynamic environment the agent can find the best strategy for him by choosing each action that maximizes the expected utility. When that environment becomes stochastic and/or multi-agent the choice of strategy is not straightforward. For these environments, other concepts are defined in order for an agent to choose their strategy.

In a multi-agent environment a strategy profile is a vector of strategies, containing one strategy for each agent.

2.2 Best Response and Nash equilibrium

A key concept for our project is the Nash equilibrium and for this reason we will dedicate this section to it.

2.2.1 Best Response

If an agent knew how the rest of the agents were going to play, the problem of picking a strategy would become simple, as the agent would choose a utility-maximizing action. So, we define $s_{-i} = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$, a strategy profile s without agent i's strategy ($s = (s_i, s_{-i})$). If the other agents except from i (denoted $-i$) were to commit to play s_{-i}, a utility-maximizing agent i would have to choose their **best response**.

Definition 2.2.1. *(Best response) Let i be a player. Then i's best response to the strategy profile s_{-i} is a mixed strategy $s_i^* \in S_i$ such that $u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i})\ \forall s_i \in S_i$.*

In a two-player normal-form game, let S_1 be the strategy set for player 1. Then strategy s_1^* is a best response for player 1 to player 2's strategy s_2 if $u_1(s_1^*, s_2) \geq u_1(s_1, s_2)\ \forall s_1 \in S_1$.

The best response characterizes a player's behavior as it gives a complete description of the strategies he will choose, given the strategies selected by other players. It does not have to be unique. However, when the support ($s_i > 0$) of a best response includes two or more actions, the agent must be indifferent among them, otherwise, the agent would prefer to reduce to zero the probability of playing at least one of the actions.

2.2.2 Nash equilibrium

In the most common case, the agent is oblivious about the other players' strategies. For this reason, the best response is a way to define a central idea of non-cooperative game theory, the Nash equilibrium, rather than a solution concept (solution concept is a formal rule for predicting how a game will be played).

Definition 2.2.2. *(Nash equilibrium) A strategy profile $s = (s_1, ..., s_n)$ is a Nash equilibrium if, for all agents i, s_i is a best response to s_{-i}.*

Intuitively, the idea of the Nash equilibrium is that a set of strategies, one for each player, would be stable if nobody has a unilateral incentive to deviate from their own chosen strategy. In a more simplistic way, if all players were to announce their strategies simultaneously, nobody would want to reconsider. A simple example is to imagine a scenario of two robbers splitting their robbery in half. If one of them was to denounce the other to the police, he would get the full spoils. However, he has no interest in doing so, because then the other one would denounce him as well and they would both end up behind bars. So splitting in half is the best solution for both of them.

2.2.3 Maxmin and Minmax strategies

The Nash equilibrium is not the only solution concept. Let us discuss one here.

Definition 2.2.3. *(Maxmin) The maxmin strategy for player i is $argmax_{s_i} min_{s_{-i}} u_i(s_i, s_{-i})$ and the maxmin value for player i is $max_{s_i} min_{s_{-i}} u_i(s_i, s_{-i})$.*

Intuitively, the maxmin strategy of agent i in an n-player, general-sum game is a strategy that maximizes i's worst-case payoff, when all the other agents happen to play the strategies which cause the greatest harm to agent i. The maxmin value of the game for player i is that minimum amount of payoff guaranteed by a maxmin strategy. the maxmin strategy is a possible choice for a conservative agent who wants to maximize his expected utility without having to make any assumptions about the other agents. The minmax strategy and minmax value can be considered the duality of maxmin. In two-player games the minmax strategy for player i against player $-i$ is a strategy that keeps the maximum payoff of $-i$ at a minimum, and the minmax value of player $-i$ is that minimum. One very important Theorem about minmax and maxmin values is the following:

Theorem 2.2.1. *(Minimax theorem[2]) In any finite, two-player, zero-sum game, in any Nash equilibrium each player receives a payoff that is equal to both his maxmin value and his minmax value.*

This theorem is important as it demonstrates that **maxmin strategies, minmax strategies and Nash equilibria coincide in two-player, zero-sum games**.

3 Repeated and Stochastic Games

Previously, we talked about games that were played one time. In this sections we will be presenting cases were a game is played multiple times.

3.1 Repeated Games

In repeated games, a given game is played multiple times by the same set of players. The repeated game is called the stage game. There are two types of repeated games : *finitely repeated games* and *infinitely repeated games*. In the first case, the game is repeated a finite and commonly-known number of times. In the second one the game is repeated infinitely often, or a finite but unknown number of times. Since in this project we implement some concrete examples of zero-sum games, the first case is within our interests.

3.1.1 Finitely Repeated Games

A finitely repeated game is a dynamic game in which a simultaneous stage game is played finitely many times, and the result of each stage is observed before the next one is played. This means that at each iteration the players do not know what the other player is playing, but afterwards they do. One example of a zero sum game is the Selling damaged goods game, whose payoff-matrix is seen in 2.
An example is the twice-played Selling damaged goods game in extensive form seen in Figure 3. In this example we assume that the payoff function of each agent is additive so in the deepest level we see the sum of payoffs of this two-stage game.

Seller \ Buyer	Buy	Pass
Sell	(1,-1)	(-1,1)
Keep	(-1,1)	(-1,1)

Figure 2: Payoff matrix of Selling damaged goods game.

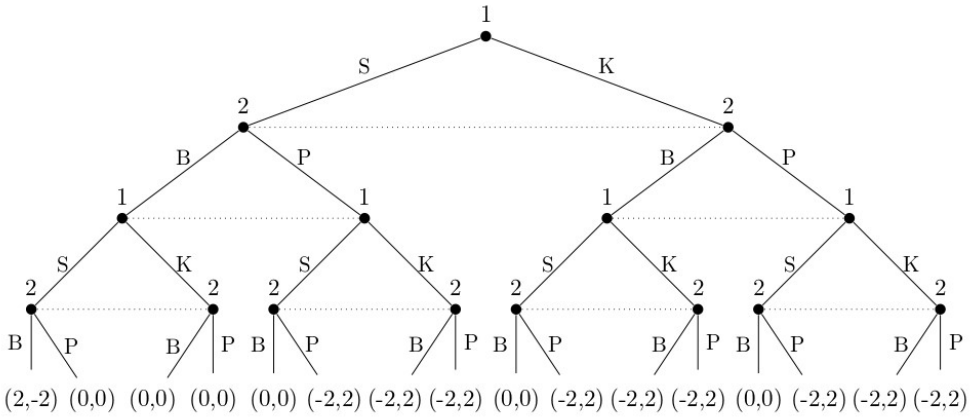


Figure 3: Twice-played Selling damaged goods in extensive form.

In a one-shot game the player ought to develop a strategy throughout the game. In the repeated game, however, the player can develop a different strategy throughout each stage game, making this concept richer strategy-wise, as the actions played at a stage game may depend on the history of play (behavioral strategy) or just the last state (Markov strategy). Surely, one strategy in the repeated game is to adopt the same strategy in each stage game. This strategy is called stationary strategy and is memory-less strategy.

3.2 Stochastic Games

3.2.1 Markov Decision Process

A Markov Decision Problems (MDP) is a model for decision making in an uncertain, dynamic world. The single agent starts out in some state, takes an action, and receives some immediate rewards. The state then transitions probabilistically to some other state and the process repeats. So, formally,

Definition 3.2.1. *(Markov Decision Problem) A Markov Decision Problems (MDP) is a tuple* (S, A, p, r) *where:*

- S *is a set of non-empty states*
- A *is a set of actions*
- $p : S \times A \times S \to R$ *indicates the probability of moving (transition probability) among states i.e.* $p(s, a, s')$ *is the probability of ending in state* s' *when taking action* a *in state* s*.*
- *The function* $r : S \times A \to R$ *returns the reward for each state-action pair.*

Note that in a MDP, the agent may or may not be aware of the environment dynamics in which he performs. The transition and reward functions could be unknown to him.

Solving known MDPs
In a MDP the agent aims to find an optimal policy π^* (function that maps a state to an action) in order to maximize their reward. One technique to calculate an optimal policy in a known MDP is a method called value iteration.

Value iteration defines a value function $V^\pi : S \to \mathbb{R}$ that specifies the value of following policy π starting in state s. Also, in the same way we define a state-action value function $Q^\pi : S \times A \to \mathbb{R}$ that shows the value of starting in a state s, choosing and taking an action a and the continuing according to chosen policy π. These two functions are related to one another via Bellman equations as follows:

$$Q^\pi(s,a) = r(s,a) + \gamma \sum_{\hat{s}} p(s,a,\hat{s}) V^\pi(\hat{s}) \tag{1}$$

$$V^\pi(s) = Q^\pi(s, \pi(s)) \tag{2}$$

For the optimal policy π^*, the second equation becomes $V^{\pi^*}(s) = max_a Q^{\pi^*}(s,a)$. The optimal policy is easily recovered from the solution to the Bellman equations. The calculation of Q and V can be done through Bellman equations as follows:

$$Q_{t+1}(s,a) \leftarrow r(s,a) + \gamma \sum_{\hat{s}} p(s,a,\hat{s}) V_t(\hat{s}) \tag{3}$$

$$V_t(s) \leftarrow max_a Q_t(s,a) \tag{4}$$

After iterating many times, Q and V converge on the Q^* and V^* values of an optimal policy π^*.

3.2.2 Stochastic Games

Stochastic games are a generalization of both MDPs and repeated games we reviewed earlier. An MDP is simply a stochastic game with only one player whereas while a repeated game is a stochastic game in which there is only one stage game. Let us formally define a **Stochastic Game**:

Definition 3.2.2. *(Stochastic Game) A stochastic game (or 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*
- *$P : Q \times A \times Q \to [0,1]$ is the transition probability function where $P(q,a,\hat{q})$ is the probability of transitioning from state q to state $\hat{q}$ after action profile a*
- *$R = r_1, \ldots, r_n$, where $r_i : Q \times A \to R$ is a real-valued payoff function for player i*

In this definition we have assumed that the strategy space of the agents is the same in all games, and thus that the difference between the games is only in the payoff function.

4 Teaching and Learning

In many Artificial Intelligent environments the main goal of the agent is to learn about their environment and act according to their goals. However, in a multi-agent environment the learning process of the agent depends also on the other agents, who may act as "teachers". Also, the learning-agent might take the role of the teacher when needed. There are a few matters that make this concept of agents' teaching and learning complex, such that the interaction between the teachers and the learners, the learning itself etc. For this reason a few notions were introduced in order to develop such an environment.

4.1 Fictitious Play

One of the earliest learning rules to be studied is Fictitious Play (FP). It is a "belief based" learning rule which means that players form and update beliefs about the opponent's strategy (*model based learning*) and behave rationally with respect to these beliefs. At first it was proposed in order to compute Nash equilibria in zero-sum games and, as it may not be the most effective method for doing that, it depends on an intuitive update rule so it is mostly viewed as a simplistic model of learning. Its convergence will be discussed later on.

4.1.1 Algorithm and characteristics

In the FP two players $i = 1, 2$ play the game G a number of n times. Each player observes the actions of the other agent in each stage game and after each stage game they update their beliefs about the other agent. More specifically, the algorithm is presented in Algorithm 1.

Algorithm 1 FP algorithm

```
Initialize beliefs about the opponent's strategy
while played less than n times do
    Play a best response to the assessed strategy of the opponent
    Observe the opponent's actual play and update beliefs accordingly
end while
```

In FP the agent cares only about their own payoff matrix. Knowing the opponents payoffs is indifferent to them, as they are only interested in their actions, which they know. So in a sense, the agent is myopic. Also, since in this learning method the agent keeps statistics about the other agent's actions in each stage game they assume that the opponents plays a mixed stationary strategy given by the empirical distribution of the opponent's previous actions. They assume that the strategy is stationary, since in this learning framework the agent keens to learn the stage game strategy and not the stochastic game strategy of the opponent.

So, for the set A of the opponent's actions, and for every $a \in A$, where $w(a)$ is number of times that the opponent has played action a, then the agent computes the probability of a in the opponent's mixed strategy as $P(a) = \frac{w(a)}{\sum_{a' \in A} w(a')}$.

As the reader may have noticed already, FP has two major drawbacks. The first one is that it assumes a stationary strategy of the opponent. If the opponents changes strategy between the stage games, then computing a probability distribution on the opponents

choice of actions won't reflect their strategy. The second drawback is that FP is very sensitive to the initialization of its prior beliefs, since a completely wrong initialization about the opponent's strategy, in a finite repeated game, would require much more iterations in order to converge to the right strategy.

It is proven that FP converges to Nash equilibrium in **two-player zero-sum games** in [3].

4.2 Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback (reward) from its own actions and experiences. Most RL problems can be formulated through MDPs. However, MDPs describe a single-agent environment and, since in this project we study multi-agent environments, we will be using the generalization of MDPs: stochastic games. In the RL approach we used, unlike FP, RL does not explicitly model the opponent's strategy, but tries to maximize its worst case scenario expected payoff.

4.2.1 Learning in unknown MDPs

Let us consider a single-agent environment where the agent participates in an MDP. In section 3.2 we defined an algorithm to compute the Q-values when the MDP is known. However, in many problems the agent is not aware of the rewards or the transition probabilities, but it can be proven that if the agent always knows what state they are in and the reward received in each iteration, they can still converge to the correct Q-values (see Algorithm 2, Theorem 4.2.1).

Algorithm 2 Q-learning algorithm

Initialize the Q-function and V values (arbitrarily, for example)
repeat
  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):
  $Q_{t+1}(s_t, a_t) \leftarrow (1 - a)Q_t(s_t, a_t) + a_t(r(s, a) + \gamma V_t(s_{t+1}))$
  $V_{t+1}(s) = max_a Q_t(s, a)$
until Convergence

Theorem 4.2.1. *Q-learning guarantees that the Q and V values converge to those of the optimal policy π^*, provided that each state-action pair is sampled an infinite number of times, and that the time-dependent learning rate a_t is $0 \leq a_t < 1$ and $\sum_0^\infty a_t$ converges to ∞ as well as $\sum_0^\infty a_t^2 < \infty$.*

One could note that the Theorem does not mention neither the rate of convergence nor it gives any assurance about the accumulation of optimal future discounted rewards by the agent. One possible scenario is that by the time the agent converges to an optimal policy it has paid a high cost, which cannot be regained by exploiting the policy going forward.

4.2.2 RL in zero-sum stochastic games

Let us remind that stochastic games are a generalization of MDPs when the environment is multi-agent. In the case of zero-sum stochastic games, where two players are involved, one approach to solve the problem is to consider that each agent ignores the existence of the other. Now, we may define $Q_i^\pi : S \to \mathbb{R}$ to be the value for agent i when both agents follow strategy profile π starting in state s and choosing action a. With this formation, Q-learning algorithm can now be applied.

In a multi-agent environment the agent is not looking for an optimal strategy, but rather a strategy that performs well against their opponents. For example, we could require that the agent uses a learning rule that is Hannan consistent i.e. against any set of opponents it yields a payoff that is no less than the payoff the agent could have obtained, by playing any one of his pure strategies throughout. It can be proven that Q-learning is Hannan-consistent for an agent in a stochastic game against opponents playing stationary policies. For more complex opponent's strategies throughout the different stage games such thing cannot be proven.

So, let us try another approach, one where the agent is aware of what actions their opponent selected at each point in its history. Then, we can define a new Q-function, $Q_i^\pi : S \times A \to \mathbb{R}$, defined over states and action profiles. Note that in a two player stochastic game $A = A_1 \times A_2$ and the formula of updating Q would be

$$Q_{i,t+1}(s_t, a_t, o_t) = (1 - a_t)Q_{i,t}(s_t, a_t, o_t) + a_t(r_i(s_t, a_t, o_t) + \gamma V_t(s_{t+1})) \tag{5}$$

In two player zero sum games, the strategy profile where each agent plays his maxmin strategy is a Nash equilibrium (Section 2.2.3). So, the payoff of the first agent is the value of the game and it helps us formulate the value function for Q-learning,

$$V_t(s) = \max_{\Pi_i} \min_{o} Q_{i,t}(s, \Pi_i(s), o) \tag{6}$$

The minimax-Q algorithm is seen in Algorithm 3. The algorithm guarantees to converge to the value of the game in the limit of infinite samples of each state and action profile pair. If the opponent is irrational and plays a strategy different from its optimal, the algorithm will not be able to exploit it in most games. The parameters α, ϵ and γ need to be specified by the user. The learning rate, however, needs to be updated as seen in 3. There are multiple ways of doing that, but the most straightforward and the one we chose in our implementation is : after a certain amount of iterations the learning rate is multiplied by a decay factor that is positive and less than one.

Algorithm 3 minimax Q-learning algorithm

Require: the set of states S , the set of the agent's actions A, the set of the oponent's actions O, the learning rate α, the explor probability ϵ and the discounting factor γ

for $\forall$ s $\in$ S, $\forall$ a $\in$ A and $\forall$ o $\in$ O **do** ▷ Initialize
  $Q(s, a, o) \leftarrow 1$
end for
for $\forall$ s $\in$ S **do**
  $V(s) \leftarrow 1$
end for
for $\forall$ s $\in$ S and a $\in$ A **do**
  $\Pi(s, a) \leftarrow \frac{1}{|A|}$
end for
$\alpha \leftarrow 1.0$
When in state s, with probability ϵ choose an action uniformly at random and with probability $1 - \epsilon$ choose action a with probability $\Pi(s, a)$ ▷ Take an action
after receiving reward r for moving from state s to s' via action a and opponent's action o : ▷ Learn
$Q(s, a, o) \leftarrow (1 - \alpha) * Q(s, a, o) + \alpha * (r + \gamma * V(s'))$
$\Pi(s, \cdot) \leftarrow argmax_{\Pi'(s,\cdot)}(min_{o'} \sum_{a'}(\Pi(s, a') * Q(s, a', o')))$
$V(s) \leftarrow min_{o'} \sum_{a'}(\Pi(s, a') * Q(s, a', o'))$
Update α

5 Experiments

In the current chapter, we will try to illustrate how these two approaches of computing equilibria in zero sum games behave. We have implemented the FP and RL algorithms and applied them on four different games. We examined whether these algorithms converge to the same equilibria, their corresponding convergence rates and their dependence on their parameters or their initial configuration. We will start demonstrating the comparison between FP and RL for each game. Then we will provide some conclusions about the results.

5.1 Matching Pennies

The first game under investigation is the famous Matching Pennies game. In the Figure 4, we display the payoff matrix of the game.

	H	T
H	(1,-1)	(-1,1)
T	(-1,1)	(1,-1)

Figure 4: Payoff matrix of Matching Pennies game

In this game, there is no pure strategy Nash equilibrium. Namely, that means that if the agents are aware of their opponent's action they have no incentive of changing their decision. For instance, considering the action profile (H, H), it does not form a Nash equilibrium as the column player would like to change his strategy and play T in order to get a payoff of 1. However, there exists a Nash equilibrium in mixed strategies. Let the column player choose action H with probability p and action T with probability $1 - p$. The row player should be indifferent between his actions as, otherwise, he would be better off changing his strategy to a pure one, according to which he only played the preferred action. So,

$$\begin{aligned} u_1(H) &= u_1(T) \\ 1 \cdot p + (-1) \cdot (1 - p) &= (-1) \cdot p + 1 \cdot (1 - p) \\ p &= \frac{1}{2} \end{aligned}$$

The above computations show that the column player plays the mixed strategy of $(H, T) = (0.5, 0.5)$. The same applies to the row player. Both players have an expected payoff equal to zero, following the equations below:

$$\mathbb{E}[agent's\ 1\ payoff] = \sum_{a \in A} s_1(a) \cdot \overline{u}_1(a)$$

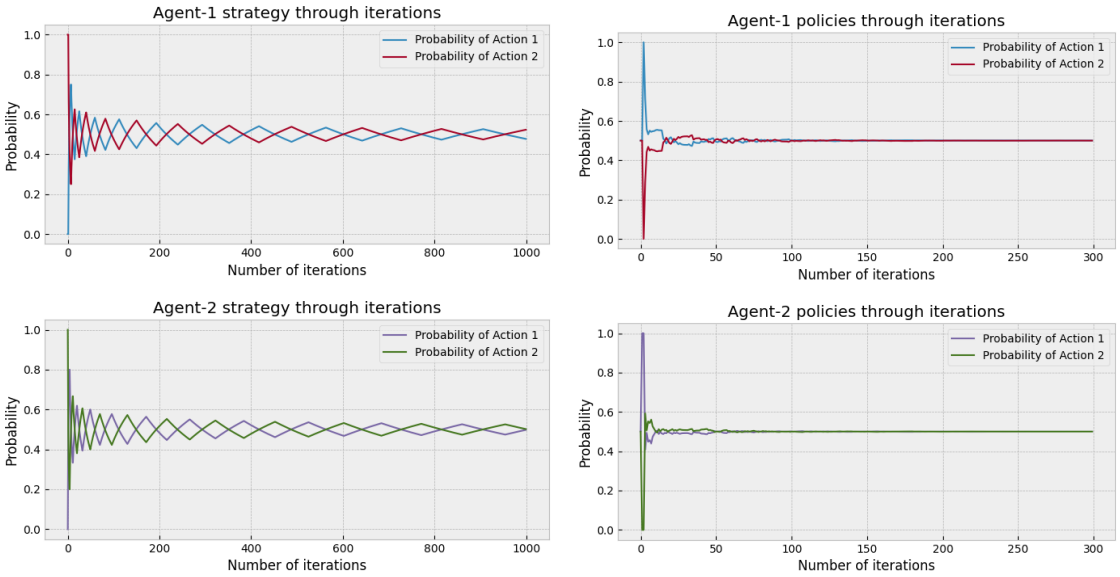
where the expected utility of agent i for an action j is,

$$\overline{u}_i(a_j) = s_{-i}(a_j) \cdot u(a_j, a_1) + s_{-i}(a_2) \cdot u(a_j, a_2)$$

and we denote with s_{-i} the strategy of the opponent.

We used the FP and RL algorithms in order to compute the mixed strategies equilibria. Both algorithms ran for 1000 iterations. As regards FP, we initialized the agents'

beliefs to be (0,1) meaning that both agents think their opponent played one time tails and zero times head in the past. On the other hand, we gave the following parameters to the RL algorithms: $\epsilon = 0.3$, $\alpha = 1.0$ and $\gamma = 0.9$. The figures in 5 illustrate the number of iterations they took to converge.



(a) Fictitious Play

(b) Reinforcement Learning

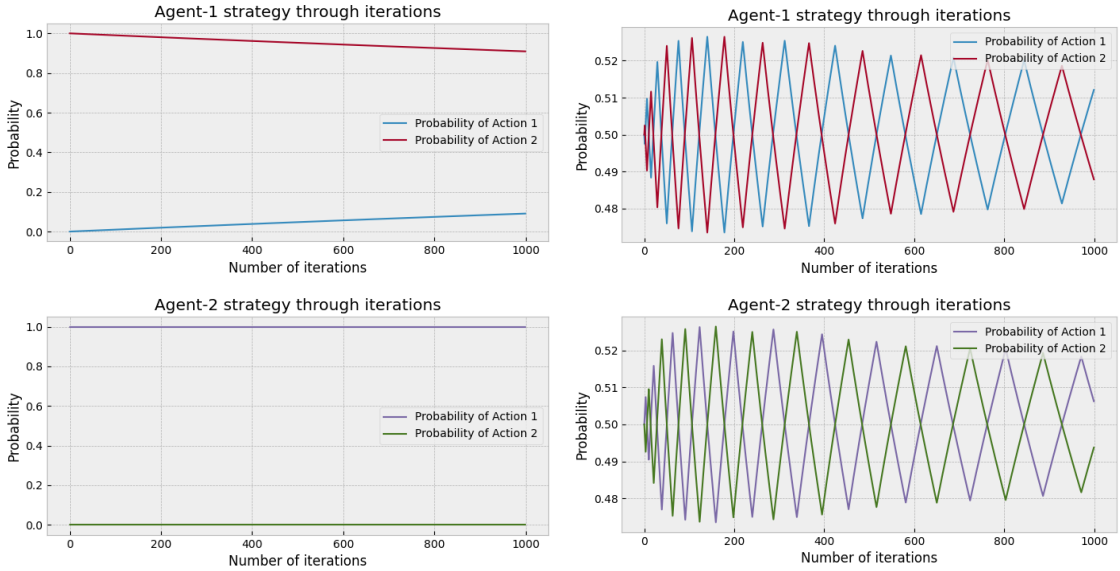
Figure 5: Convergence comparison (Matching Pennies)

As one can notice, the RL algorithm converges fairly quickly (∼50-100 iterations) for each agent. On the contrary, for the FP approach the convergence is slower.

Nonetheless, the convergence of both algorithms is dependent on their corresponding parameters. As seen in 6 in FP the initialization of the agents' beliefs plays an important role on the algorithm's behavior: In 6a where the initial belief is that the column player has played 10000 times tails and zero times head the row player has played zero times tails and 10000 times head in the past, the FP algorithm needs approximately 100000 iterations to converge to the mixed strategies (not included in the figure for clarity reasons). Moreover, in 6b the initial beliefs for both players are that the opponent has played 100 times head and 100 times tails in the past and in this case, since the initial beliefs agree with the mixed strategy of each player, the algorithm converges almost immediately.

On the other hand, it seems that the RL approach is quite more robust to its initial beliefs. As we can see in Figure 7 even when we initialize the policies in a such way that we introduce some kind of preferences over the agents' actions, the algorithm manages to converge within the same amount of iterations with much less oscillations rather than FP.

However, the RL algorithm's convergence is not that sensitive to the parameters in this relatively simple game. In Figure 8 are presented three cases in which we changed only one parameter of the algorithm ($\epsilon = 0.9$, $\alpha = 0.5$ and $\gamma = 0.5$). The convergence is a little slower than when the initial parameters where used in the cases where we set $\epsilon = 0.9$ and $\alpha = 0.5$ respectively. In the case where we changed γ to 0.5 the convergence was a bit faster. Nonetheless, in all three cases RL has converged to the mixed strategies until the 100^{th} iteration.



(a) Initial beliefs (10000,0) and (0,10000) (b) Initial beliefs (100,100) and (100,100)

Figure 6: Strong initial beliefs in Fictitious Play (Matching Pennies)

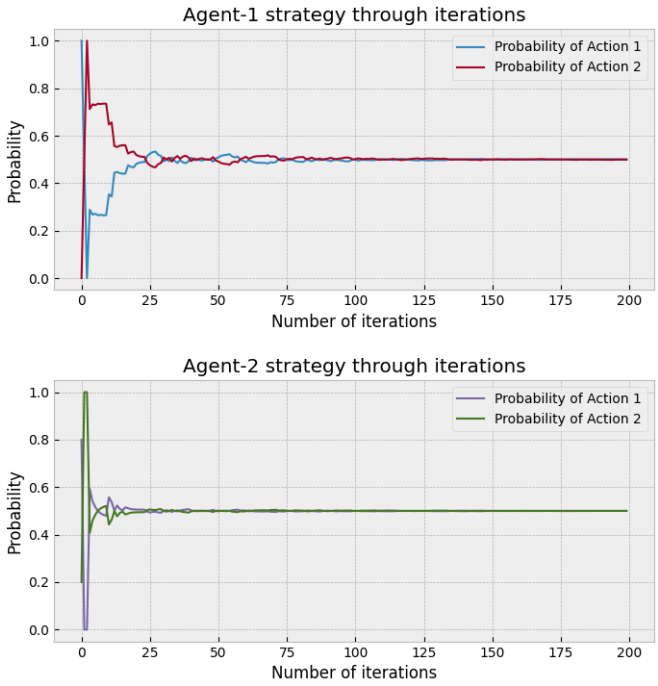
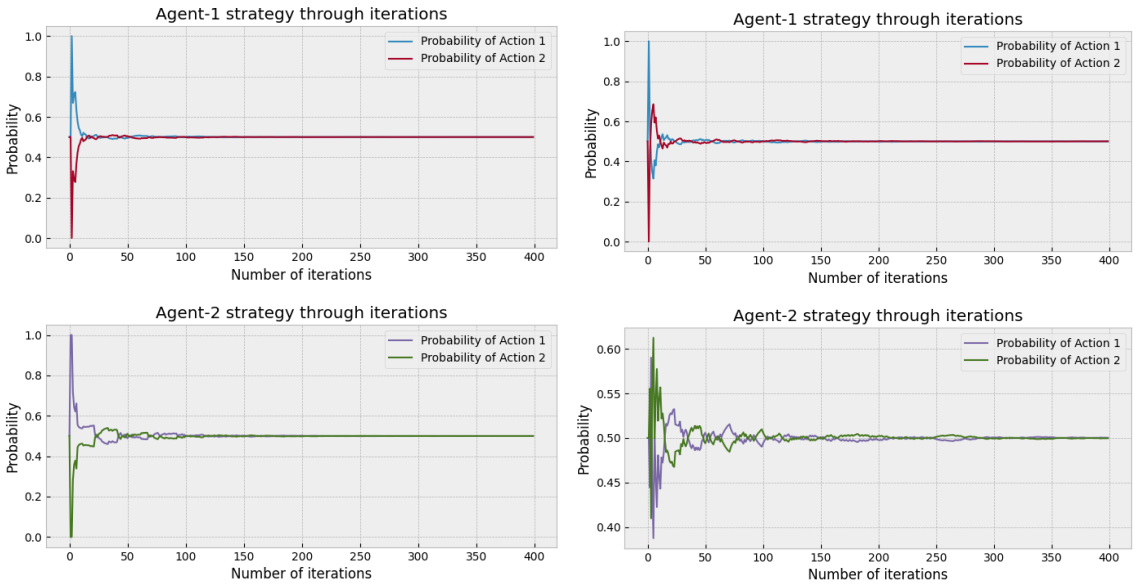
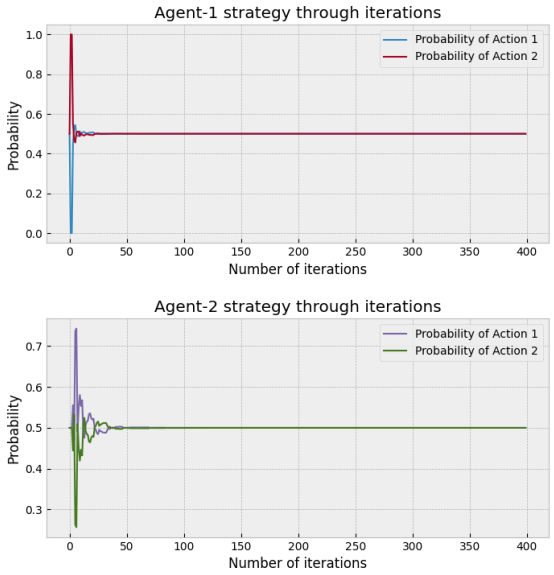


Figure 7: Initial policies in RL: row player (1.0, 0.0) and column player (0.8, 0.2)



(a) $\epsilon = 0.9$

(b) $\alpha = 0.5$



(c) $\gamma = 0.5$

Figure 8: Different Parameters RL (Matching Pennies)

5.2 Custom game 1

The second game we chose is a zero sum game with the payoff matrix displayed in Figure 9. Again, in this game there exists no pure Nash Equilibrium as in each action profile

	C	D
A	(3,-3)	(1,-1)
B	(2,-2)	(4,-4)

Figure 9: Payoff matrix of Custom game 1

at least one player has the motive to change their strategy and choose another action to increase their payoff. Contrarily, there is a Nash equilibrium in mixed strategies. Let us compute it as we did in the previous section. We suppose that the column player chooses action C with probability p and action D with probability $1 - p$. Since the row player must be indifferent between his actions (A, B):

$$\begin{aligned} u_1(A) &= u_1(B) \\ 3 \cdot p + 1 \cdot (1 - p) &= 2 \cdot p + 4 \cdot (1 - p) \\ p &= \frac{3}{4} \end{aligned}$$

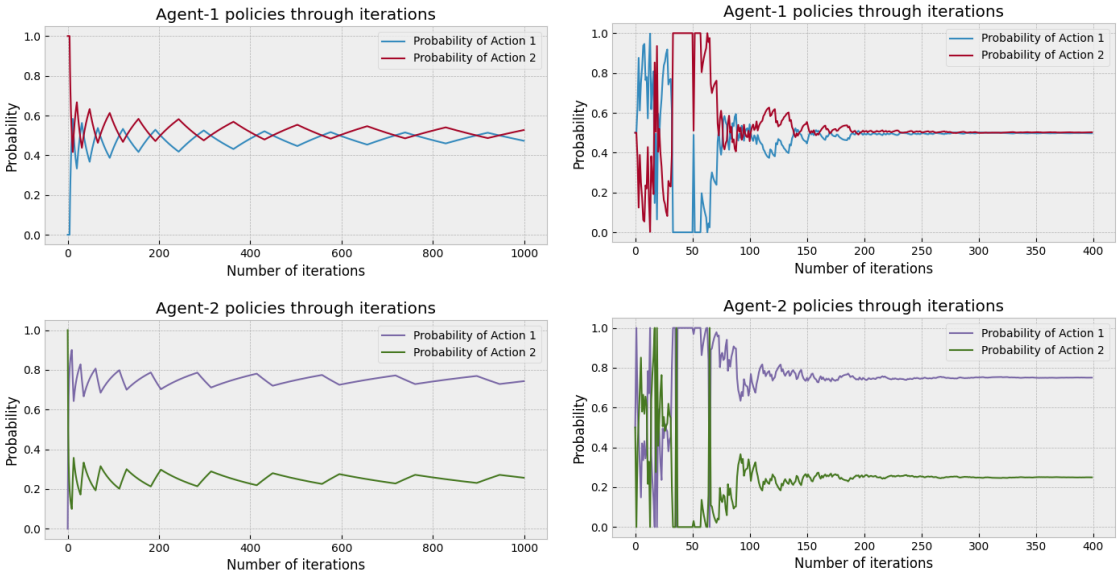
which leads us to the conclusion that the mixed strategy of the column player is $(0.75, 0.25)$. Likewise, let us assume that the row player chooses action A with probability q and action B with probability $1 - q$. As the column player is indifferent between their actions,

$$\begin{aligned} u_2(C) &= u_2(D) \\ (-3) \cdot q + (-2) \cdot (1 - q) &= (-1) \cdot q + (-4) \cdot (1 - q) \\ p &= \frac{1}{2} \end{aligned}$$

the mixed strategy of the row player is $(0.5, 0.5)$. The expected payoff of row player is equal to 2.5, whereas the expected payoff of column player is equal to -2.5.

The FP and RL algorithms were used in order to compute the mixed strategies equilibria. The number of iterations that both algorithms ran is, again, 1000 and in FP the initial agents' beliefs are $(0, 1)$. Also, the RL parameters are no different from "Matching Pennies": $\epsilon = 0.3$, $\alpha = 1.0$ and $\gamma = 0.9$. The figures in 10 illustrate the number of iterations they took to converge.

In this game, the RL algorithm converges to each mixed strategy in between 100 to 200 iterations and the FP algorithm needs more than 1000 iterations to converge and we denote big osculations in its convergence rate. In Figure 11 we present two figures that illustrate the convergence of FP with different initial beliefs. In 11a, the initial belief is that, in the past, the column player has played 10000 times C and zero times D and the row player has played zero times A and 10000 times B. In this case the algorithm converges after 400000 iterations (again not illustrated for clarity reasons). In 11b the initial beliefs agree with the strategy of row player and are different from the strategy of column player (yet not very far from the truth). So, in this case, the algorithm converges to the strategy of the row player quickly and lot long after it converges to the strategy of the column player.

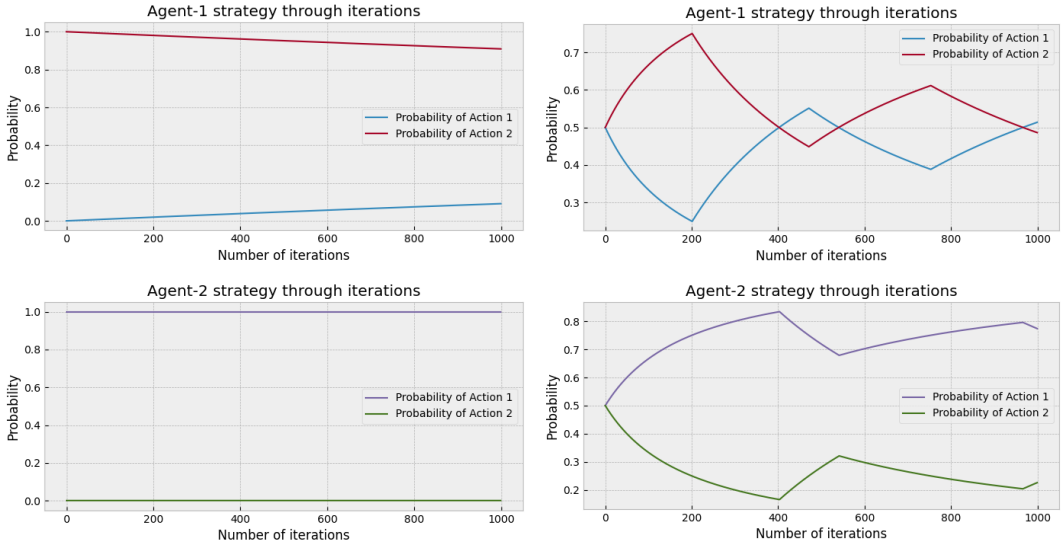


(a) Fictitious Play

(b) Reinforcement Learning

Figure 10: Convergence comparison (Custom game 1)

Regarding the parameters of RL we noticed the same pattern as in the matching pennies case. The convergence rate stayed pretty much unchanged.



(a) Initial beliefs (10000,0) and (0,10000)

(b) Initial beliefs (100,100) and (100,100)

Figure 11: Strong initial beliefs in Fictitious Play (Custom game 1)

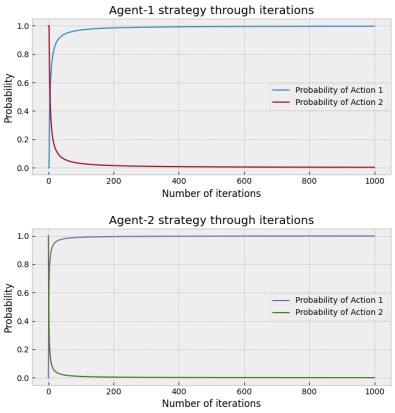
5.3 Custom game 2

The payoff matrix of the third game we chose is displayed in Figure 12. In this game

	C	D
A	(2,-2)	(2,-2)
B	(1,-1)	(3,-3)

Figure 12: Payoff matrix of Custom game 2

there exist both pure Nash and mixed strategy equilibria. The pure strategy equilibrium is the action profile (A, C) as in that case no agent has the incentive of changing their strategy. The payout for the row player is equal to 2 and for the column player it is -2. The equilibria are computed in the same way as before and the row player plays the pure strategy of action A, and the column player plays the mixed strategy of $(0.5, 0.5)$. The payoff of row player is 2 and the expected payoff of column player is equal to -2.



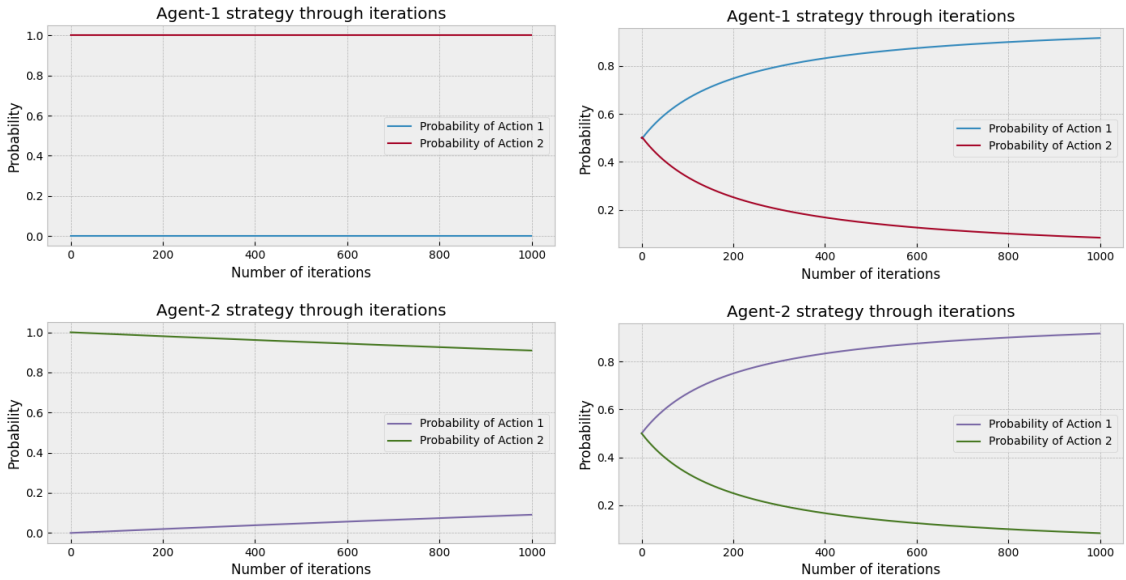
(a) Fictitious Play

(b) Reinforcement Learning

Figure 13: Convergence comparison (Matching Pennies)

In this game, as seen in Figure 13 each algorithm converges a Nash equilibrium. In FP the convergence to the pure Nash equilibrium is almost immediate for the initial beliefs $(0, 1)$ and $(0, 1)$. Using the RL approach the algorithm converges fast to the pure

strategy of the row player and after 250 iterations to the pure strategy of the column player. As noted previously, the FP algorithm converges very fast for the aforementioned initial beliefs, but, as seen in 14, the convergence is slower when initializing the beliefs to $(100, 100)$ and $(100, 100)$. Also, for initial beliefs $(0, 10000)$ and $(0, 10000)$, that are in a way the opposite strategies, the algorithm converges to the pure strategies after 1000000 iterations. However, it should be noted that for initial beliefs $(100000, 100000)$ and $(1, 100)$ that highly suggest that the column player plays a mixed strategy of $(0.5, 0.5)$, the FP algorithm converges to the mixed Nash Equilibrium. The same thing happens if we initialize the learning rate α to 0.1 in the RL approach.



(a) Initial beliefs (0,10000) and (0,10000)

(b) Initial beliefs (100,100) and (100,100)

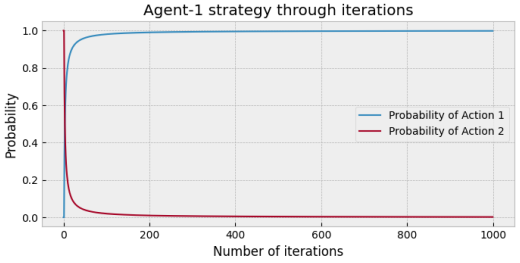
Figure 14: Strong initial beliefs in Fictitious Play (Custom game 2)

5.4 Selling damaged goods

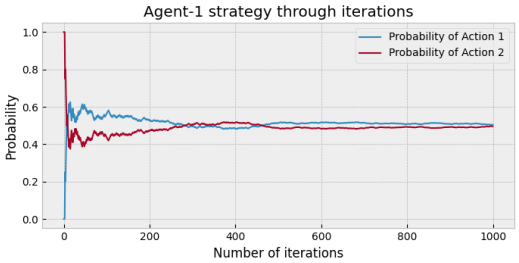
Last game to be studied is the "Selling damaged goods" game. Its payoff matrix can be viewed in 2. In this game exist two Nash equilibria in pure strategies. These equilibria are $(Sell, Pass)$ and $(Keep, Pass)$. There, also, exists an equilibrium in the pure strategy of $Pass$ of the column player and the mixed strategy $(0.5, 0.5)$ of the row player. In each equilibrium, the row player's expected payoff is -1 and column player's expected payoff is 1.

In the "Selling damaged goods" game a strong correlation between FP's convergence and its initial beliefs emerges. In Figure 15, that shows the convergence of the FP algorithm with different prior beliefs, one can notice two things. First of all, in 15c and 15d where the prior beliefs are biased towards to an equilibrium, the algorithm converges to that exact equilibrium. Secondly, in 15a and 15b, where we assume we do not have any knowledge about the players' past actions, the algorithm does not always converge to the same equilibrium. We can see that in a game of multiple equilibria FP converges to one of them and this convergence is highly correlated with its initial beliefs.

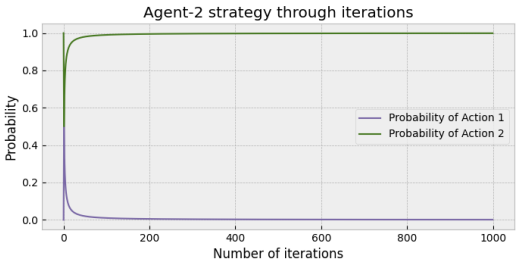
On the other hand, RL always converges to the mixed Nash equilibrium regardless of the

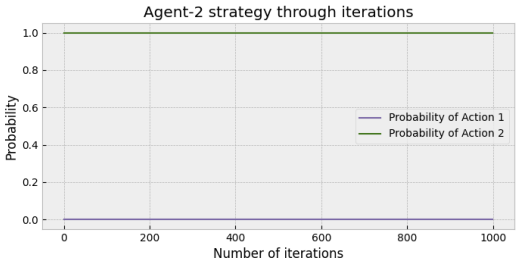


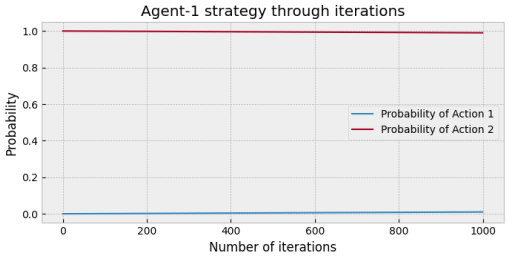
(a) Initial beliefs (0,1) and (0,1)



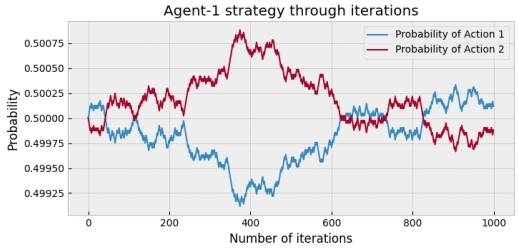
(b) Initial beliefs (0,1) and (0,1)

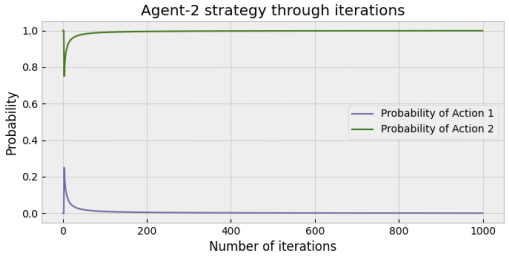






(c) Initial beliefs (0,1) and (0,100000)





(d) Initial beliefs (0,1) and (1000,1000)

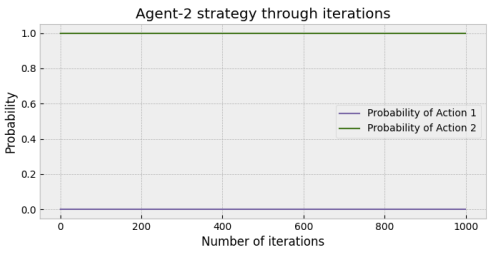
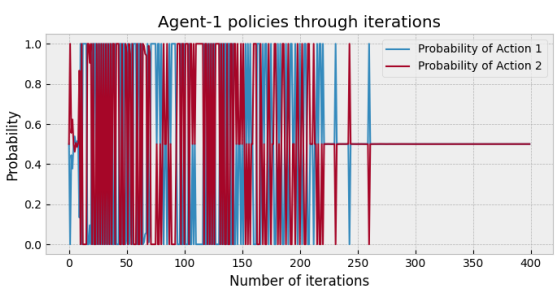
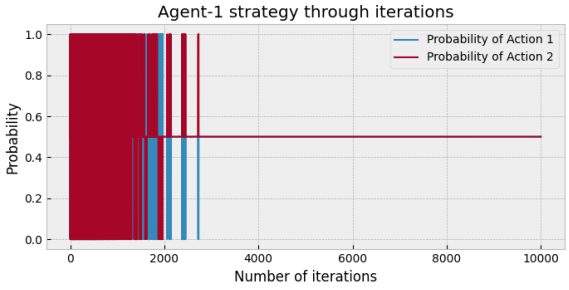


Figure 15: Different initial beliefs in Fictitious Play (Selling Damaged Goods)

initial values of its parameters. In Figure 16 are displayed two executions of the algorithm with different learning rates. It is clear that, in this case, the different initialization of the learning rate a results to faster or slower convergence.



(a) Starting with $a = 1.0$



(b) Starting with $a = 0.1$

Figure 16: Convergence of Reinforcement Learning (Selling Damaged Goods)

6 Conclusion

After the examination of both FP and RL for the computation of Nash Equilibria on multiple games, we would like to conclude some points in this section. To begin with, it was quite clear that the convergence rate, as well as the strategy to which the FP converged, were highly correlated to its prior beliefs. According to the experiments we performed, in cases when both pure and mixed strategies exist, FP showed a slight preference to pure strategies. However, we saw that not only when we executed the algorithm again with the same settings, but also when we altered its initial beliefs to some extreme values, FP was able to find different equilibria strategies. For example, in the "Selling damaged goods" game after running the algorithm 100000 times, 33404 of them converged to the mixed strategy equilibrium, 66559 to one of the pure equilibria and 37 to the other pure equilibrium. Another interesting thing is when we tested the assumption where FP is based on (that both players play a stationary strategy throughout the state games, section 4.1.1), and we forced one of the agents to respond uniformly at random in each stage game. By doing so, the other agent was unable to converge to an equilibrium strategy, but the first one did. Conversely, RL proved to be more robust to its initial configuration. Regardless its starting point, RL converged to the same equilibria between different runs. In the "Selling damaged goods", for example, after executing the algorithm 1000 times, all of them converged to the mixed strategy equilibrium. The convergence rate, though, depends on the parameters ϵ, γ and α as we demonstrated before and was significantly higher that FP.

References

[1] Jiawei Li and Graham Kendall. On nash equilibrium and evolutionarily stable states that are not characterised by the folk theorem. *PLoS ONE*, 10:e0136032, 08 2015.

[2] J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.

[3] Julia Robinson. An iterative method of solving a game. *Annals of Mathematics*, 54(2):296–301, 1951.

[4] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, USA, 2008.