Automated Curriculum Discovery for Emergent Reciprocity

Thomas Foster-Brown

Pembroke College

University of Oxford Trinity Term 2022

0 Acknowledgements

I have a number of people to thank for their helping hand in my completion of this report. I am incredibly grateful to Jakob Foerster for his expert knowledge and guiding hand. This project would have been impossible without his help.

I'd also like to thank Timon Willi and Chris Lu, whose weekly meetings kept me on track and provided me with invaluable perspectives on the problems I worked through in the course of this project.

My final thanks is to Oxford's Advanced Research Computing cluster. Without use of the incredible facility, a lot of the computational work in this project simply wouldn't have been possible.

Contents

1 Abstract

Learning in multi-agent settings is a frontier in research into reinforcement learning. Although we can use reinforcement learning to find optimal strategies in single-agent contexts, the introduction of multiple agents makes the search for optimal results far more complicated. As multiple agents learn, their learning changes not only their behaviour, but also the optimal behaviour for the other agent once it has reacted to this change of strategy.

The impact of multiple agents learning in games where agents can cooperate or betray has significant real-world parallels, such as countries at war, trade negotiations, animals competing for food and more [1, 2, 3]. How reinforcement learning algorithms interact with each other within these environments is a frontier for the intersection of computer science and game theory, as it can allow us to better understand how cooperation can be achieved to ensure the best possible situations for all parties in the long run. It is especially important given that the prevalence of reinforcement learning is only going to increase, so finding how agents can cooperate with both humans and other learning agents is a much needed topic for further research.

In this report, we investigate how these multi-agent systems evolve over time, and how we can reach stable long-term cooperation in an environment where multiple agents are learning simultaneously.

2 Related Work

Much has been researched on different strategies and how they interact in the game. Some are simple and are fixed or based only on the last turn, and some are more complex. Researchers such as Wakano and Yamamura have created more complex strategies that involve weighing up and thresholding past actions when deciding which move to make on a given turn [4].

Grofman and Poole used a "cooperation index" to investigate the link between the payout structure and the strategy of the other player [5]. They mathematically proved that strategies using partial Tit-for-Tat can induce pure cooperation in the opponent. We shall build on this in this report by analysing how multiple learning agents can induce stable cooperative behaviours in each other while using greedy exploration algorithms.

Much work has also been done on cooperation within Q-learning. Tan [6] analysed how cooperation

between two Q learning agents can lead to higher rewards in a hunter-prey grid world experiment. Our experiment differs from that, as agents in the IPD can only share information through their actions, rather than sharing sensory information like in Tan's experiments.

One of the difficulties in multiagent RL is that both agents have to account for noise in the other players actions due to their exploration. Zeng et al. [7] have shown that high levels of noise discourage cooperation in an evolutionary setting. In this report we shall research whether this is also the case for learning agents.

Foerster et al. use deep learning and modify the learning of an agent to be able to also account for the learning of the other agent [8]. They found this to be a successful method for inducing learning agents into a Tit-for-Tat strategy compared to naive deep learning. Our project differs from this, as I shall be focussing on tabular Q learning and will use greedy rather than stochastic functions for agents to choose their actions.

3 Project Aims

In this project, we investigate how learning agents learn in the Iterated Prisoner's Dilemma. In these games, the long-term strategy that rewards both agents the most is cooperation. However, learning agents struggle to learn this strategy due to the higher reward from defecting on a given turn, much like a prisoner might be tempted to betray his accomplice for less time in prison [9]. In these problems, we assume that agents act in a selfish way and try to maximise their own reward without regard for the opponent's reward.

We will investigate how different strategies evolve in the multi-agent setting, and how different hyperparameters affect the outcomes when learning agents play each other. We will also analyse how different sets of strategies are learnt and how they agents transition from one strategy to another.

The aim of this report is to find curricula that lead to long-term cooperative strategies between agents. A curriculum here refers to the idea that the environment an agent is in shapes its learning, so different reward structures can change results. We will be using tabular Q-Learning as the reinforcement learning method, which is an algorithm that allows an agent to assign a numerical value for each possible action in a given state.

3.1 Findings

In this report, we find that long-run cooperative strategies can be stable and that they are more common than more adversarial strategies, depending on the hyperparameters used for the learning agents. We will show that there are stable loops that exist in which agents periodically change their strategies yet still remain broadly cooperative.

We also find that analysis of the best response to each strategy is not sufficient to guarantee its stability, and we can quantify how the hyperparameters used in the learning agents affect these stabilities.

4 Background

Matrix games have been researched and theorised about for a long time. The Prisoner's dilemma, which is the focus of this report, was conceptualised by Flood and Dresher in 1950 [10], and was formalised later by Tucker [11]. The Iterated Prisoners Dilemma has been the subject of much research on Machine Learning. Axelrod [12] hosted a tournament in which experts in game theory submitted strategies to play against each other. The winner was Tit-for-Tat, a strategy which we shall be analysing in this paper in which an agent plays the same move that the opponent did on the last turn.

Reinforcement learning is a powerful tool that allows a learning agent to discover the best actions for a given state, even in complex systems. Using machine learning, an agent with no prior information on a system, nor any information on how states and actions are linked, can discover the actions with the highest reward for each given state. This means that a learner can learn to play different games optimally by exploring different states and actions, even learning to play video games in some instances [13]. Sandholm and Crites [14] used Q Learning, a type of reinforcement learning, to analyse how agents in the Iterated Prisoner's Dilemma learn in the presence of other learning agents. This is important as the presence of other learning agents makes the environment non-stationary from the perspective of both agents.

4.1 Matrix Games

Matrix games are defined as multi-player games involving two or more players who are able to choose from a discrete range of actions [15]. Both agents must choose an action on a given turn, and cannot know the other player's action until it has decided its own action. Players are then rewarded or punished according to both their own action and the action of the opponent. This is done through the use of a

reward matrix. An example of a Rewards table for a game of "Split or Steal" is shown below.

		Column Player	
		Split	Steal
Row Player	Split	Row Player earns £5000 Column Player Earns £5000	Row Player Earns £0 Column Player Earns £10,000
	Steal	Row Player earns £10,000 Column Player Earns £0	Row Player Earns £0 Column Player Earns £0

Table 1: Example Matrix Game - Split or Steal

As we see in the Rewards Matrix, if the Row Player chooses to "Split", its reward will depend on the action of the Column Player. It will earn either £5,000 if the column player splits, or £0 if the column player steals.

The crucial point of matrix games is the lack of knowledge of the opponents choices, as this often will heavily affect the outcome for a player. Players in matrix games aim to maximise their reward without considering the reward of the opponent, so agents learn selfishly.

4.2 The Prisoner's Dilemma

The game used in this report is the Prisoner's Dilemma, a matrix game. In this game, 2 agents play each other and each player has two possible moves. They can either cooperate with the other player, or they can defect. The Rewards Matrix for this Matrix Game is shown in Table 2. This is repeated for a number of timesteps with the same two players playing against each other.

	Player 2 Cooperates	Player 2 Defects
Player 1 Cooperates	Reward - Reward Player 1 earns R = 3 points Player 2 earns R = 3 points	Sucker - Temptation Player 1 earns S = 0 points Player 2 Earns T = 5 points
Player 1 Defects	Temptation - Sucker Player 1 earns T = 5 points Player 2 Earns S = 0 points	Punishment-Punishment Player 1 earns P = 1 point Player 2 earns P = 1 point

Table 2: An example rewards system for the prisoner's dilemma

We can see here that the total reward for both players is maximised when both players choose to cooperate, giving a total reward of $2R = 6$. However, a player maximises their reward by defecting against a cooperative opponent. We see that if both players choose to defect, the total reward is the lowest total of any combination of choices.

In the Prisoner's Dilemma, the optimal action is always to defect. This is because a player will maximise

their reward by defecting, whether or not the opponent chooses to defect. If the opponent cooperates, the player will earn 5 points instead of 3 by defecting, and if the opponent defects, the player will earn 1 point instead of 0. The fundamental problem of the prisoners dilemma is that both players are therefore incentivised to defect, but they would both attain higher rewards mutually if they both cooperate.

The payout matrix of the prisoner's dilemma can be changed. The matrix in Table 2 shows one of many possible payout structures. To be a true prisoner's dilemma, the following condition must be met [16]:

$$T > R > P > S \tag{1}$$

with T, R, P, and S as defined in Table 2. Rule 1 means that the reward for defecting must always be greater than the reward for cooperating, and that the cooperation of the opponent must always lead to a higher reward for the player.

Weak forms of the dilemma also exist, but we do not consider them in this report. In these weak forms, $T \geq R$ or $P \geq S$, instead of the strict inequalities shown in Rule 1. These forms do not have the same property that states that it is always better to defect on a single turn, instead that defecting has to be at least as good as cooperating [16].

4.3 The Iterated Prisoners Dilemma

This brings us to the Iterated Prisoner's Dilemma, henceforth referred to as the IPD. This is the same base game as the Prisoner's Dilemma, however instead of playing only once, players play the game against each other multiple times. This fundamentally changes the game, as there is now more of an incentive for players to cooperate, as each player has the ability to punish defection by themselves defecting on the next turn or thereafter.

The payoff matrix used in the IPD must abide by Rule 1, and must also abide by the following constraint:

$$2R > T + S \tag{2}$$

This means that the highest possible reward must be gained by both agents cooperating, rather than

alternating between cooperation and defection [17].

Another aspect to consider is the number of iterations. This has to be unknown to the player because a rational player will always defect on the last turn, as there is no future reward to consider. The rational player, knowing this, will also defect on the penultimate turn, knowing that their opponent will also defect on the last turn. This effect cascades through each turn and means that the game has a simple answer of "always defect". We solve this by having a random horizon for the problem.

4.4 The Markov Decision Process

To model the game, we define a Markov Decision Process, or MDP [18]. An MDP is defined as a tuple $M = \langle S, A, P_a, R_a \rangle$, where S is a finite set of possible states, A is a finite set of possible actions. P is the State-Transition function, which defines the probability that when an agent takes action a in state s, it will progress to another state s', $P_a(s, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$. This is demonstrated in Figure 1.

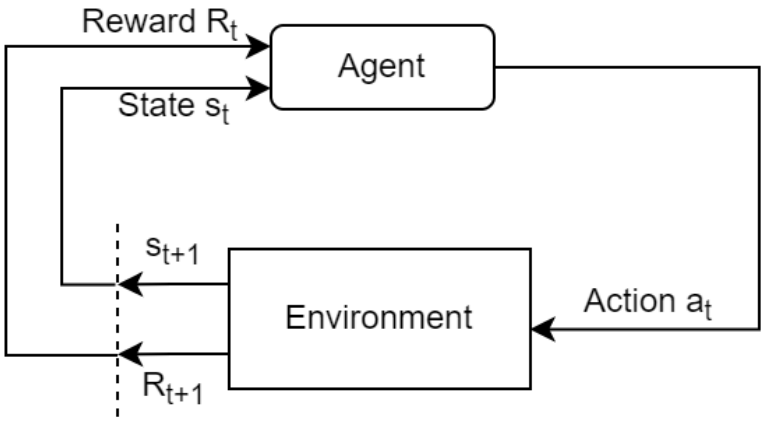


Figure 1: Single agent MDP Loop

R is defined as the reward assigned to an agent immediately upon moving from state a state s to another state s' after taking action a.

The aim of an MDP is to create a policy, Π, which maps states to actions. These can be probabilistic; therefore, given a certain state, an agent will select its next action probabilistically given its current state and its policy.

To define the MDP for the IPD, we must define all of these terms. When a learning agent plays against another agent with a set policy, we can define this as a single agent stationary MDP. In this case, the state is defined as some history of past actions made in the game. The actions are to defect or cooperate. R is given by a rewards matrix such as in Table 2. P is simple, as the next state only depends on the

agent's actions when the opponent is fixed with no noise.

When two learning agents interact, an MDP is no longer sufficient and does not describe the system adequately. Instead, we can define a Multi-Agent Markov Decision Process, or MMDP [19]. This is defined as M = $\langle S, \alpha, \{A_i\}_{i \in \alpha}, P_a, R_a \rangle$. The values are defined as above, plus the difference that α is a finite list of agents, and A_i is the set of actions available to an agent i.

In the MMDP, multiple agents select actions simultaneously, which then proceed to form a joint action. The state transition matrix then uses this joint action to move from state s to s'. This is demonstrated in Figure 2. The state is consistent between both agents, but the agents do not necessarily have the same reward as each other.

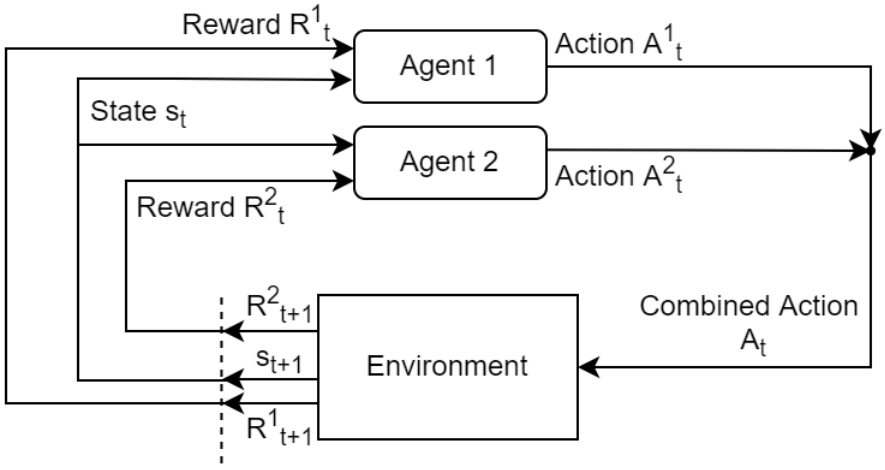


Figure 2: Two agent MDP Loop

5 Methods

5.1 Tabular Q-Learning

To attempt to solve the MMDP found in the Iterated Prisoner's Dilemma, we will use Tabular Q-learning. Tabular Q-learning is a reinforcement learning algorithm in which a value is learnt in a table for each state-action pair. These are called Q-values and describe the return of each state-action pair. Q-learning aims to find the optimal action-value function $Q*$ [20], which is defined as:

$$Q^*(s,a) = \mathbb{E}(r_{t+1} + \gamma \max_a(Q(s',a'))) \tag{3}$$

This means that $Q*$ is the expected value of the immediate reward upon completing action a in state s, added to a discounted value for the highest value action available in the next state. γ is the discount factor. This controls how heavily weighted future rewards are. When it is 0, the learner only considers the next action, and when it is higher, it weighs future rewards more highly. s_t and a_t are the state and action chosen at a given point.

The agent learns by updating the Q-values for each action as it takes it. A benefit of Q-learning is that the learning agent requires no prior knowledge of the environment in which it exists, it only has to explore to find optimal strategies for the game it is playing. As the agent explores, it updates its Q table, which is done using:

$$Q^{new}(s_t,a_t) = Q(s_t,a_t) + \alpha[r_t + \gamma \max_a(Q(s_{t+1},a)) - Q(s_t,a_t)] \tag{4}$$

where α is the learning rate, which dictates how heavily the agents should weigh new experiences against its existing knowledge [21]. This variable dictates the extent to which the Q-value for a state-action pair is updated, with higher values of α causing the old values to be overwritten more quickly with new data.

This algorithm aims to approximate the optimal action-value function Q*. This is the function that has a perfect numerical value for each state-action pair.

Agents can choose their moves using a variety of methods. In this report we will focus on two.

5.1.1 Convergence of Q to Q*

The aim of Q-learning is to use Q to approximate the expected value of $Q*$ using Equation 4. This therefore solves the MMDP by helping each agents to learn optimal policies through their Q-values. We can define TD-error as follows:

$$\text{TD-error} = r_t + \gamma \max_a(Q(s_{t+1}, a)) - Q(s_t, a_t) \tag{5}$$

The TD-error, or Temporal Difference error, describes the difference between the measured discounted return during a step and the current estimation of the Q value for a state-action pair. Equation 4 therefore adds this error to the current value of Q(s,a). Therefore, we can see that when $\gamma = 0$, ΔQ is proportional to the difference between the reward and the estimation of the reward. Using a continuous approximation when α is small, we can therefore say:

$$\frac{dQ(s_i, a_i)}{dt} = \alpha[r_t - Q(s_i, a_i)] \tag{6}$$

Where $Q(s_i, a_i)$ is a particular state-action pair. Therefore:

$$Q(s_i, a_i) = r_t(1 - e^{-\alpha t}) \tag{7}$$

which tells us that the Q value will exponentially converge to the reward, which is the same as $Q*$ when $\gamma = 0$. However, when $\gamma > 0$, convergence cannot be described simply due to the presence of the term $\gamma \max_a(Q(s_{t+1}, a))$. If this term has already converged to the optimal value, that is, the agent is entering a well-explored state, the convergence will be exponential, as the term will then be constant. If not, convergence will be different from an exponential, as the value of the next best action will either be overestimated or underestimated, resulting in either faster or slower changes in Q than the exponential calculated above. Despite this, convergence is still guaranteed for a single agent [20].

These convergence results do not hold for multi-agent settings, as the other agents responses may change as it learns, therefore the actions they take in a given state might change. This then causes the Q*-values for the state-action pairs of the first agent to change too.

5.1.2 Epsilon-Greedy

Agents must use an algorithm to choose their action based on its Q-values. One such algorithm is ϵ-greedy [22]. This is described in Equation 8. Epsilon refers to the probability of a random action being taken. The function balances *exploration*, ie. random actions, with *exploitation*, which refers to choosing the action with the highest known Q-value. The action a_t on a given turn is chosen as follows

$$a_t = \begin{cases} \max_a(Q(s_t, a)) & \text{with probability } (1 - \epsilon) \\ \text{random } a & \text{with probability } \epsilon \end{cases} \tag{8}$$

This is useful, as it allows the learning agent to both use the current information it has to make optimal decisions and be able to explore and find more optimal solutions than the ones it already has.

5.1.3 Boltzmann Exploration

While epsilon greedy exploration chooses either the highest-valued action or a random action, Boltzmann exploration means that the Q-values dictate the probability of the action being chosen. This is also known as stochastic exploration. This is described by Equation 9 below [14].

$$p(a_i) = \frac{exp(Q(a_i)/\tau)}{\Sigma_{j=1}^{n} exp(Q(a_j)/\tau)} \tag{9}$$

This ensures that when one Q value is larger, it is more likely to be chosen, so it takes advantage of known data. Other actions also have a chance of being chosen, which ensures that there is still exploration.

Here, τ refers to a temperature and, as it increases, random exploration will increase. Different problems will have different values of τ leading to better results, and often τ is annealed, or gradually reduced, over time to increase exploitation and decrease exploration when more data is known.

5.1.4 Online Vs Offline Learning

There are 2 methods of learning when using Q-learning. The first is online learning. This is when the Q table is updated after every step using Equation 4. The second is offline learning, in which the learning happens at the end of the episode, with the Q-values constant during the episode, and the sequence of events known by both agents as they update their Q tables. [23]

For single agent learning, the difference between these methods is minor, and learners will perform just as well reaching Q* with either method. However, when learners are in a multiagent setting, online learning can have different outcomes. For example, two agents may have 2 possible actions with 2 Q-values very close together, with one Q value in equilibrium over the other. When online learning is used, the 2 agents could both choose an unlikely action a few times in a row, which might cause the lower Q value to overtake the higher one, causing the policy to change mid-episode. However, when using offline learning, it would require many more of these random actions to change the policy, as the average reward from the episode will have a lower variance than a few actions.

5.1.5 Tabular Q-learning - Example

We will start with an easy problem to demonstrate Q-learning. An agent starts on the grid shown in Table 3 at point S. The episode ends when it reaches point G, or after 50 turns, whichever is sooner. The agent is rewarded with 10 points when it lands on point G, and -1 point when it lands on any other space.

				G
S				

Table 3: Q-learning environment

	↑	
←	X	→
	↓	

Table 4: Possible movements of the agent

In this example, we will use epsilon-greedy exploration to learn the optimal Q-values, with $\epsilon = 10\%$. We will also use a learning rate of $\alpha = 1$ and a discount factor of $\gamma = 1$. When we initialise the Q learner, it has no knowledge of the system, so all the Q-values for each state are 0.

Table 5 shows the Q values for each cell after a single turn. The agent explores spaces and is rewarded -1 on each turn, so updates Q for these state action pairs with a new value of -1.

[-1 -1 0 0]	[-1 -1 0 0]	[-1 0 -1 0]	[-1 -1 -1 0]	[0 0 0 -1]
[-1 0 0 0]	[0 0 0 0]	[-1 -1 0 0]	[-1 0 0 0]	N/A
[-1 0 0 0]	[0 0 0 0]	[0 0 0 0]	[0 0 0 0]	[0 0 0 0]
[-1 0 0 0]	[0 0 0 0]	[0 0 0 0]	[0 0 0 0]	[0 0 0 0]
[-1 0 0 -1]	[0 0 0 0]	[0 0 0 0]	[0 0 0 0]	[0 0 0 0]

Table 5: Q-values for learner after 1 episode, 20 steps long

In the next episodes, the agent explores more and further fills in the Q table. Table 6 shows the Q-values after 3 episodes, when the agent first reaches the goal. We can see that in the state immediately below

the goal, the Q value for the "Up" action is 10, as this is the reward that it was given upon taking this action. Now, any movement that takes the agent from another cell into this cell will give that action a value of 9, as Equation 4 shows us that the Q value will be equal to $Q_{new} = -1 + 10$, as the reward for that turn would be -1, and the maximum Q in that state is 10, as per Table 6.

[-1 -1 0 0]	[-1 -1 -1 0]	[-1 -1 -1 0]	[-1 -1 -1 -1]	[0 0 0 -1]
[-1 0 0 0]	[-1 -1 -1 0]	[-1 -1 -1 0]	[-1 0 0 0]	Goal
[-1 0 0 0]	[-1 0 -1 0]	[-1 -1 0 0]	[-1 -1 0 0]	**[10 0 0 0]**
[-1 -1 0 0]	[-1 -1 -1 0]	[-1 -1 0 0]	[-1 0 0 0]	[0 0 0 0]
[-1 -1 0 -1]	[-1 -1 0 0]	[-1 0 0 0]	[0 0 0 0]	[0 0 0 0]

Table 6: Q-values for learner after 3 episodes

Now that we have more Q-values around the goal, more episodes will begin filling in the table with the optimal Q-values using Equation 4. The Optimal Q-values are learnt after a large number of episodes, as the agent must randomly explore each possible state-action pair once the next state has already learnt its optimal action. These optimal Q-values are shown in Table 7

[5 6 6 5]	[6 7 7 5]	[7 8 8 6]	[8 9 9 7]	[9 9 10 8]
[5 7 5 6]	[6 8 6 6]	[7 9 7 7]	[8 10 8 8]	Goal
[6 6 4 5]	[7 7 5 5]	[8 8 6 6]	[9 9 7 7]	[10 9 8 8]
[5 5 3 4]	[6 6 4 4]	[7 7 5 5]	[8 8 6 6]	[9 8 7 7]
[4 4 3 3]	[5 5 4 3]	[6 6 5 4]	[7 7 6 5]	[8 7 7 6]

Table 7: Ideal Q-values for grid shown in Table 3

If we focus on the cell immediately below the goal, we can analyse why these ideal Q-values have emerged. The Q value for the "Up" action is 10, as it is immediately awarded 10 points with no future rewards. The Q value for the "Right" action is 9, as it loses 1 point that turn and remains in the same location, able to attain the reward of 10 on the next turn. The Q-values for the "Down" and "Right" actions are 8, as they lose a point on that turn, and the cells they land on are further away from the goal, so the maximum Q-values at those cells are 9. This can also be considered by the idea that moving left will require 2 extra movements to get to the goal, so $Q_{down/left} = -2 + 10$ for the cell we are analysing.

5.1.6 Tabular Q-learning in the IPD

Given what we know, we can now apply tabular Q-learning to the IPD. As discussed in Section 5.1, we know that we need to define the MDP. The actions in this case are either cooperate of defect. The states can be defined by some history of past actions by the player and the opponent. One such method of defining this history is "One-step history". This defines 5 states: Initial, CC, CD, DC, and DD. The

initial state is the state at the start of the episode when no plays have been made. The other states refer to what each agent did in the previous turn. For example, DC means that the player defected and the opponent cooperated on the previous turn.

Other horizons can also be defined, such as a 2 step history, where in addition to the states listed above, there also exist states such as "CCCC", which means both agents cooperated on the previous 2 turns. Longer horizons give more information, and as the horizon grows, the number of states increases with order 2^{2n}, so learning will take much longer as there are more states to explore.

5.2 Strategies in the IPD

There are a number of defined strategies discussed in other papers [24] researching the iterated dilemma. I will outline some of these strategies and some of their benefits and weaknesses in the following sections.

5.2.1 Always Cooperate

The first strategy we should consider is continued cooperation, regardless of what the agent or the opponent has done in the past. When both agents always cooperate, they will always maximise their collective reward. Using the rewards matrix in Table 2, we can see that if both agents always cooperate, they will always earn 3 points each, totalling 6 points. This is a good long-term situation for both players.

The issue with playing this strategy is that a learning opponent will quickly learn that it can defect and earn 5 points instead of 3. Given that agents in this game are selfish and don't consider any rewards that aren't their own, they will then adopt a strategy of defection. This means that the cooperative agent will earn 0 points for each round. We can therefore conclude that the best response to always cooperating is always defecting.

We can prove this by analysing the discounted returns, which are the returns once the discounted future returns have been considered. The discounted returns for cooperating when playing against this agent are calculated in Equation 10. The returns for defecting follow a similar equation, shown in Equation 11.

$$\begin{aligned} V_{coop} &= R + \gamma V_{Coop} \\ \therefore V_{coop} &= \frac{R}{1-\gamma} \end{aligned} \tag{10}$$

$$V_{defect} = \frac{T}{1-\gamma} \tag{11}$$

Returns depend on the rewards matrix used. Here, we define the returns using the values S, P, R, and T. We can see that for all values of gamma, the discounted returns will be higher when always defecting. The discounted rewards depend on the payout matrix used, but since $T > R$, it will always be optimal to defect here.

5.2.2 Always defect

Another possible strategy is to always defect. In single step games, always defect will always score higher than or equal to its opponent, as the only possible outcomes are DC, where the agent attains a higher reward than the opponent, and DD, where it and its opponent score equally. However, in tournaments with multiple strategies playing against each other, more cooperative strategies will win, as always defecting can never enter the cooperative steady state, which allows higher rewards than the DD equilibrium.

As with the strategy of always cooperating, there is an optimal strategy here, which is to always defect. This Defect-Defect equilibrium is not ideal for either agent, as they both earn 1 point for a total of 2, which is less than in the Cooperate-Cooperate state where they both get 3 points. Neither agent will want to try and cooperate however, as they know that on that turn the opponent will, resulting in going from 1 point to 0.

We can see from the discounted rewards in Equation 12 that the optimal play is always to defect, as $P > S$.

$$V = \begin{cases} \frac{S}{1-\gamma} & \text{Always cooperate} \\ \frac{P}{1-\gamma} & \text{Always defect} \end{cases} \tag{12}$$

5.2.3 Tit-for-Tat

Tit-for-tat (hereinafter referred to as TFT) is a much more balanced strategy than the two already mentioned. In TFT, the agent cooperates on the first turn. On subsequent turns, its action will be the opponents action from the previous turn. This means that it will punish the opponent's defections and reward the opponents cooperation.

This strategy also adapts to the previous 2. When playing against the cooperative agent, both agents will cooperate and exploit the more rewarding long-term equilibrium of Cooperate-Cooperate, and when playing against the agent that always defects, it will not allow itself to be exploited (except on the first turn) and will end up in the Defect-Defect equilibrium. This can be seen in Figure 3.

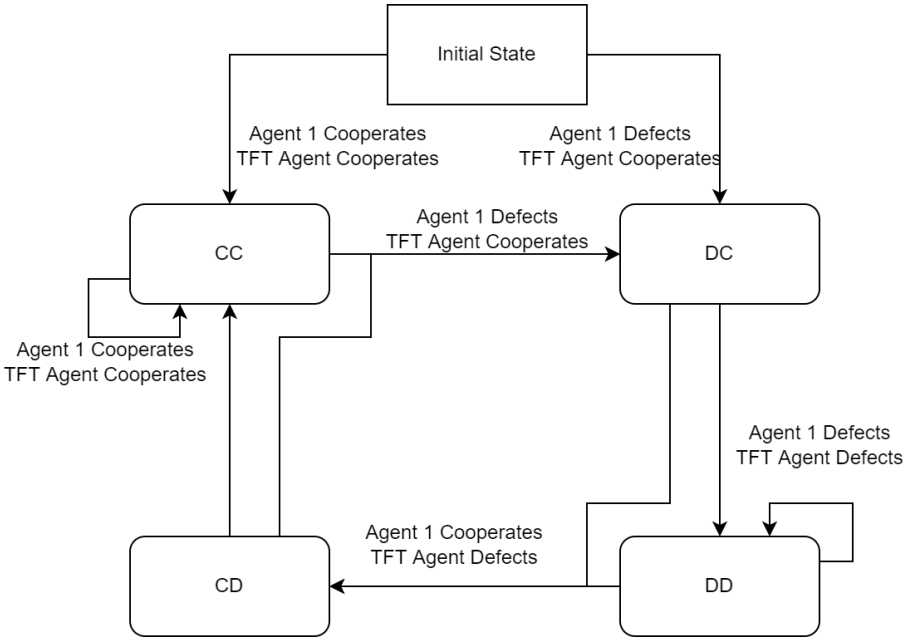


Figure 3: Markov decision chain for an agent playing against a TFT opponent

Figure 3 shows us the MDP for an agent playing against a Tit-for-Tat opponent. We can see that the first state after each agents first action will always be either CC or DC, as the TFT agent will always cooperate on the first turn. We can also see that the only two stable states are CC and DD, since the DC and CD states can lead to only other states and not themselves. This is an important point, as it shows that the TFT agent will not allow itself to be continually exploited by remaining in the CD state and receiving the sucker's payoff of 0 each turn.

When two TFT agents play each other, they will both start by cooperating, so will both continue to cooperate on subsequent turns, meaning TFT agents when playing against each other will end up in the best mutually beneficial long term equilibrium.

There are optimal strategies against TFT, and they depend on the value of γ used in equation 4. When the discount factor is low, the optimal play involves more defection, as the agent does not weigh future rewards heavily. This means that we are likely to reach a suboptimal long-term equilibrium in favour of the short term reward from defection. When the discount factor is higher, we care more about future rewards, so cooperation is incentivised. The discounted returns V are shown in Equation 13 [14].

$$V_{TFT} = \begin{cases} \frac{R}{1-\gamma} & \text{Always cooperate} \\ \frac{T+\gamma S}{1-\gamma^2} & \text{Alternate between defecting and cooperating} \\ T + \frac{\gamma P}{1-\gamma} & \text{Always defect} \end{cases} \tag{13}$$

The optimal strategies when using the [5,3,1,0] payout structure depend on γ and are as follows:

$$\text{Optimal Strategy Vs TFT} = \begin{cases} \text{Always defect} & \gamma < \frac{1}{4} \\ \text{Alternate between cooperation and defection} & \frac{1}{4} \leq \gamma < \frac{2}{3} \\ \text{Always defect} & \gamma \geq \frac{2}{3} \end{cases} \tag{14}$$

5.2.4 Grim Trigger

When playing Grim Trigger, sometimes referred to as "Spiteful" in other publications, an agent will cooperate initially, but once it its opponent defects it will then subsequently always defect [25]. Using one-step history, this means the agent will only cooperate in the Initial and CC states, and will defect in the CD,DC and DD states.

This strategy can be viewed as harsh, as it doesn't allow any forgiveness, even if an opponent tries to cooperate to return to a CC equilibrium. It is however an effective strategy at ensuring the opponent doesn't defect, as the threat of indefinite removal of cooperation is a powerful deterrent.

Figure 4 visually shows us the possible state transitions when an agent plays against a Grim Trigger opponent. The most important thing to note is that once the DC state has been reached, it is not possible to enter the CC or DC state again, so the agents are fixed in the CD and DD states.

The discounted returns when playing against Grim Trigger are the same as when you always defect or always cooperate when playing against TFT, as defined in Equation 13. Like TFT the optimal strategy depends on the value of γ used. This is shown in Equation 15.

$$\text{Optimal Strategy Vs Grim Trigger} = \begin{cases} \text{Always defect} & \gamma < \frac{1}{2} \\ \text{Always cooperate} & \gamma \geq \frac{1}{2} \end{cases} \tag{15}$$

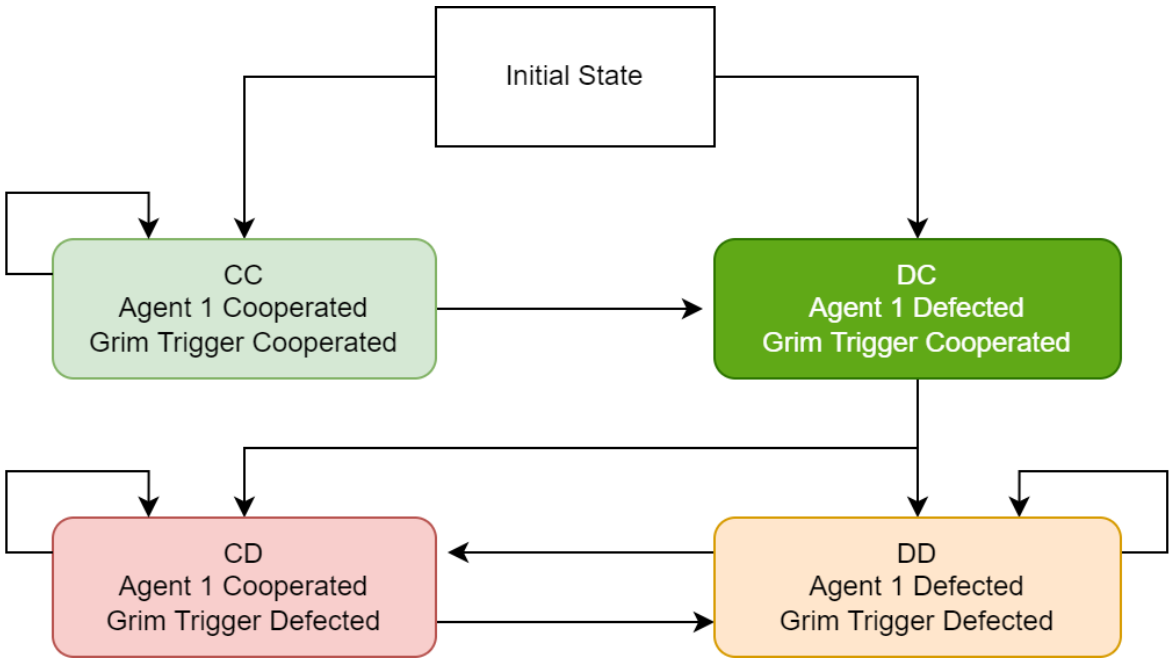


Figure 4: Grim Trigger strategy diagram

5.2.5 Pavlov

Pavlov is another strategy which can be defined with a 1-step history. In Pavlov, the agent will defect only in the DC and CD states. This means that Pavlov agents will be cooperative until an opponent defects. Once the opponent defects, the Pavlov agent will then defect on the next turn. If both agents defect, the Pavlov will cooperate to attempt to restore cooperation.

$$V_{Pavlov} = \begin{cases} \frac{R}{1-\gamma} & \text{Always cooperate} \\ \frac{T+\gamma P}{1-\gamma^2} & \text{Always defect} \end{cases} \tag{16}$$

This means that the optimal strategy against a Pavlov opponent is to always defect and to alternate between the rewards of T and P.

6 Experimental Setup

The experiments conducted in this paper will be divided into three main parts, which are as follows:

6.1 Verification that Q Learners Learn the Best Response to Other Strategies

The first experiment is to verify that a Q-learning agent learns the best-response to other strategies. In this section, we will initialise Q-learning agents with empty Q tables. We shall be using 1-step history for all of these experiments. The algorithm used in this report shall be epsilon-greedy, as most other research in this area has used stochastic methods and the use of greedy agents provides a novel avenue for research.

We shall vary the hyperparameters of the learner and see what effect this has on learning. The aim in this section is to confirm that the learners learn the best response for each strategy as defined in Section 5.2.

Further to this, we shall see the effect of added noise in the decision process of the opponent. For this part of the experiment, Q-learning will be performed across a range of values of $\epsilon_{opponent}$ and the resulting Q values, and associated strategies strategies, shall be analysed. This will give us a basis with which we can analyse the effect of noisy exploration when both agents are learning.

6.2 Stability of Strategies in Multiagent Self-play

The second set of experiments is designed to test which strategies are stable in self-play. In this section, we will initialise two learning agents with a set of Q values that are associated with the strategy we would like to test. For example, to test the stability of the "Always Defect" strategy, $Q(s_i, D) = 100$ and $Q(s_i, C) = 0$ across all states s_i could be used as initial values for both agents. Both agents will be exploring using the Epsilon-Greedy algorithm, so they will then adopt these strategies, but with added noise for the purpose of exploration.

Q-learning will then be performed, with both agents learning simultaneously, until either a steady state is reached, or the agents learn different strategies.

We shall be investigating the effect of the hyperparameters of the two agents on this stability, so that we know the values of the hyperparameters that we should use for the final experiment. Both agents will use the same value of α and ϵ as each other, and shall be performing online learning, in which their Q table

is updated after every step.

6.3 Evolution of Strategies Between Learning Agents

The final experiment is to investigate how different strategies evolve when two learning agents are initialised with no knowledge of the game, and no strategy. This means that $Q(s, a) = 0$ for all state-action pairs. For each step, we shall log the combined strategies of both agents before and after updating the Q table. We can therefore observe which strategies the agents can reach when initialised at 0. When extended for a large number of steps, we can also observe any stable strategies that are reached, as well as any stable loops of strategies that occur.

We will also plot the average reward for the agents against time over a large number of initialisations. If the average reward decreases with learning, this means defection is dominant, and if the average reward increases, then the agents have learned more cooperative long term strategies.

7 Results

7.1 Verifying that Q-Learning learns optimal strategies

As discussed in the previous section, there are optimal strategies when playing against static agents. The first experiment in this paper will to verify that our Q learning agent learns the correct strategies against these agents.

7.1.1 Learner Vs Always Cooperate

In this example, the opponent will always cooperate regardless of the actions the agent has taken. We have used the values $\alpha = 1$ and $\epsilon = 0.5$, as the value of each state-action pair is constant with no randomness, so having these parameters be high causes the agent to reach $Q*$ faster.

After a small number of 10 step episodes, the Q table when $\gamma = 0.9$ is shown in Table 8. As expected it learned that defecting has a higher Q-value then cooperating. As expected, it learned that the Q value of defection is 50, which fits with the value expected in Equation 11. The Q value for cooperating is 48, because the highest Q value in the next state is for defecting, so, using Equation 3, we can calculate $Q*(C) = R + \gamma \times Q*(D) = 48$.

The Q values for CD and DD remain 0, as the opponent never defects, so the agent never reaches these states and thus they remain at their initial value of 0. The CC, DC and initial states all have the same Q values because the state doesn't affect the actions of the opponent, as it will always cooperate.

	Cooperate	Defect
CC	48	50
CD	0	0
DC	48	50
DD	0	0
Initial	48	50

Table 8: Q values learned playing against the always cooperate strategy

	Cooperate	Defect
CC	0	0
CD	9	10
DC	0	0
DD	9	10
Initial	9	10

Table 9: Q values learned playing against the always defect strategy

7.1.2 Learner Vs Always Defect

The analysis for this section is much the same as in Section 7.1.1. The Q values in this case are lower, as instead of 3 or 5 points, the agent can only earn 0 or 1 point playing against an always defect opponent.

The Q values are shown in Table 9. These results align with the maximum expected return found in Equation 12.

7.1.3 Learner Vs TFT

As mentioned in Section 5.2.3, the optimal strategy against a Tit-for-Tat opponent is dependant on γ. In Table 10, we can see how the Q values change as γ increases.

	C	D
CC	4.05	5.25
CD	4.05	5.25
DC	1.05	1.25
DD	1.05	1.25
Initial	4.05	5.25

(a) $\gamma = 0.2$

	C	D
CC	6.33	6.67
CD	6.33	6.67
DC	3.33	2.67
DD	3.33	2.67
Initial	6.33	6.67

(b) $\gamma = 0.5$

	C	D
CC	30	29.3
CD	30	29.3
DC	27	25.3
DD	27	25.3
Initial	30	29.3

(c) $\gamma = 0.9$

Table 10: Learned Q values playing against TFT agent

When $\gamma = 0.2$, we can see that in every state $Q(s_i, D) > Q(s_i, C)$, so when random actions chosen under epsilon greedy are ignored, the learner will always defect, as predicted in Equation 14. Similarly, we can see that when $\gamma = 0.9$, $Q(s_i, C) > Q(s_i, D)$ in every state, so in any state the agent will choose to cooperate, also fulfilling our expectation in Equation 14.

When $\gamma = 0.5$, we can see that the Q values for cooperation are higher than defection only in the DC and DD states. This means that the agent will defect initially, and the TFT opponent will cooperate. This will put the agent into the DC state, in which it will cooperate and the opponent will defect, leading to the CD state where it will defect again. The agent and its opponent will then alternate between CD and DC, fulfilling our Equation 14 expectations.

7.1.4 Learner Vs Grim Trigger

Like in Section 7.1.3 above, we expect to see different strategies develop in the Q table when different values of γ are used. The results for $\gamma = 0.2$ and $\gamma = 0.8$ are shown in Table 11. As expected, we can see that when γ is low, the agent will initially defect to take advantage of the higher initial reward. It will then keep defecting, as it has learned that it gets a higher reward when it defects, as the grim trigger agent will now always defect.

When γ is high, the agent cooperates as the long-term reward is higher. In the states where the grim trigger has already started defecting however it will defect, as it is now in the same situation as the agent

	C	D
CC	4.05	5.25
CD	0.25	1.25
DC	0.25	1.25
DD	0.25	1.25
Initial	4.05	5.25

(a) $\gamma = 0.2$

	C	D
CC	30	14
CD	9	10
DC	9	10
DD	9	10
Initial	30	14

(b) $\gamma = 0.9$

Table 11: Learned Q values playing against Grim Trigger agent

playing against "always defect" in Table 9, as it knows that cooperating is pointless as the opponent will not start cooperating again.

7.2 Playing Against Noisy Agents

In all of the aforementioned examples, the action of the opponent is solely determined by its strategy combined with the previous actions of the agent. We will now investigate what occurs when randomness from the opponent is introduced.

7.2.1 Playing Against Noisy TFT

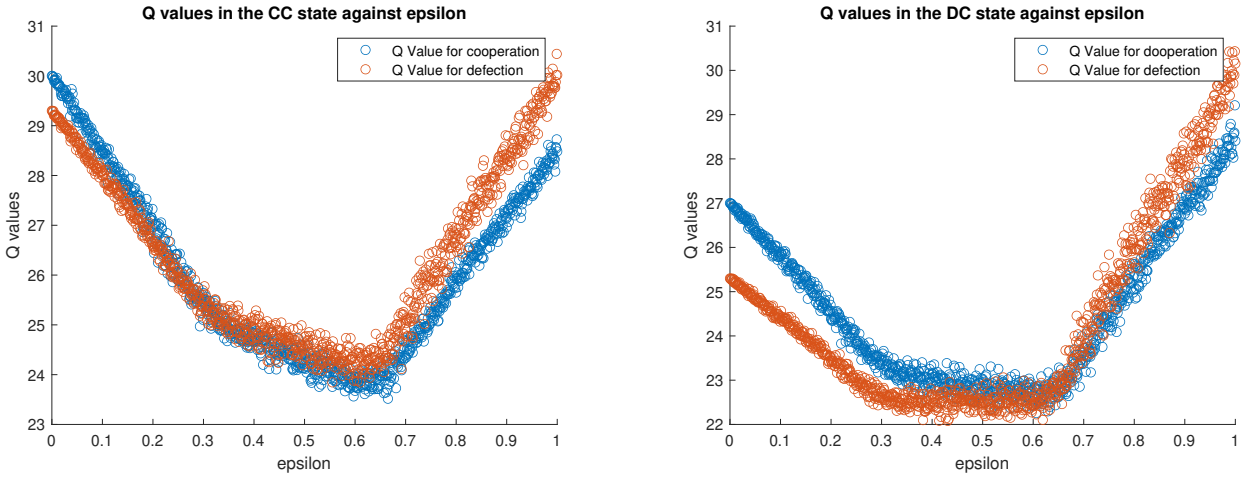
When playing against a noisy TFT opponent, the optimal decisions depend not only on γ, but also on the probability that the agent acts randomly instead of following its usual strategy. In this section, ϵ_2 is the chance that the opponent acts randomly.

When ϵ_2 is 0, the agent acts like the agent in the previous section; it is deterministic. As the degree of randomness increases, Q* changes.

As epsilon increases, the expected return of cooperation decreases, as there is now an $\epsilon/2$ chance that despite the agent's cooperation, the opponent will still defect. This means that the expected reward will be $3(1-\epsilon_2/2)+0(\epsilon_2/2)$ when cooperating. Similarly, the expected reward from defection is $5(1-\epsilon_2/2)+1(\epsilon_2/2)$. It is clear that the value of cooperation and defection both decrease, with defection decreasing in value faster, as the difference between the rewards of 5 and 1 is greater than the difference between the rewards of 3 and 0.

When $\epsilon = 1$, it is clear that there is no benefit to cooperation. Much like when the agent is playing against "always defect" and "always cooperate" strategies, the actions of the agent have no impact on the actions of the other agent as it will only act randomly. In this situation, there is a 50% chance of

defection and a 50% chance of cooperation, so when the agent chooses to cooperate the expected return is $3 \times \frac{1}{2} + 0 \times \frac{1}{2} = 1.5$, whereas the expected reward when defecting is $5 \times \frac{1}{2} + 1 \times \frac{1}{2} = 3$.



(a) CC State

(b) DC State

Figure 5: How the Q values for cooperation and defection vary against noisy TFT agent

Figure 5 shows how the Q values of defection and cooperation vary with ϵ in the CC and DC states for an experiment with $\gamma = 0.9$. As theorised above, the agent learns that when ϵ is high, the discounted return for defection is higher than that for cooperation. The initial state, CC and CD states have identical Q values like in Table 10. The DC and DD states are also identical. I will therefore omit these repeated diagrams from this report.

We can also see that the results are noisy, rather than a smooth line. This is because the agent is trying to approximate Q* using learned rewards, but in this case the rewards depend on the random actions of the opponent, so an "unlucky" run for the agent will lead to an underestimation of $Q*$

We can see from Figure 5 that the Q values vary in a piece-wise linear way. The reason for this can be seen in Equation 3. We get the "kinks" in the distribution of Q values when the next best action changes between cooperation and defection. We can see that this kink occurs in Figure 5a at $\epsilon = 0.3$ when the cooperation and defection lines meet.

We can see from the diagrams that the optimal strategy depends on the value of *epsilon*. These strategies are as follows:

$$\text{Optimal Strategy Vs Noisy TFT} = \begin{cases} \text{Always cooperate} & \epsilon < 0.3 \\ \text{Alternate between cooperation and defection} & 0.3 \leq \epsilon < 0.65 \\ \text{Always defect} & \epsilon \geq 0.65 \end{cases} \tag{17}$$

These results confirm that cooperation is only optimal when randomness is low, as the agent was able to keep the higher rewards of the CC equilibrium. As randomness increases, the benefit of cooperation is reduced as the agent will be betrayed more and more often. When ϵ is around 0.5, it makes sense to defect in the CC state, as the expected return for defecting is significantly higher than than for cooperating, so the future benefit is outweighed by the current reward.

When ϵ is high, it makes sense to always defect. As ϵ increases, so does the expected return. This is because once the agent has chosen an always defect strategy, a higher ϵ means that the opponent is less likely to punish the agent's defection.

This variance of the results for Q can be estimated mathematically. For example, in the CC state, the expected return for cooperation is shown in Equation 18. Given that all state-action pairs have this variance in each measurement, it is clear that estimating Q* could be an issue.

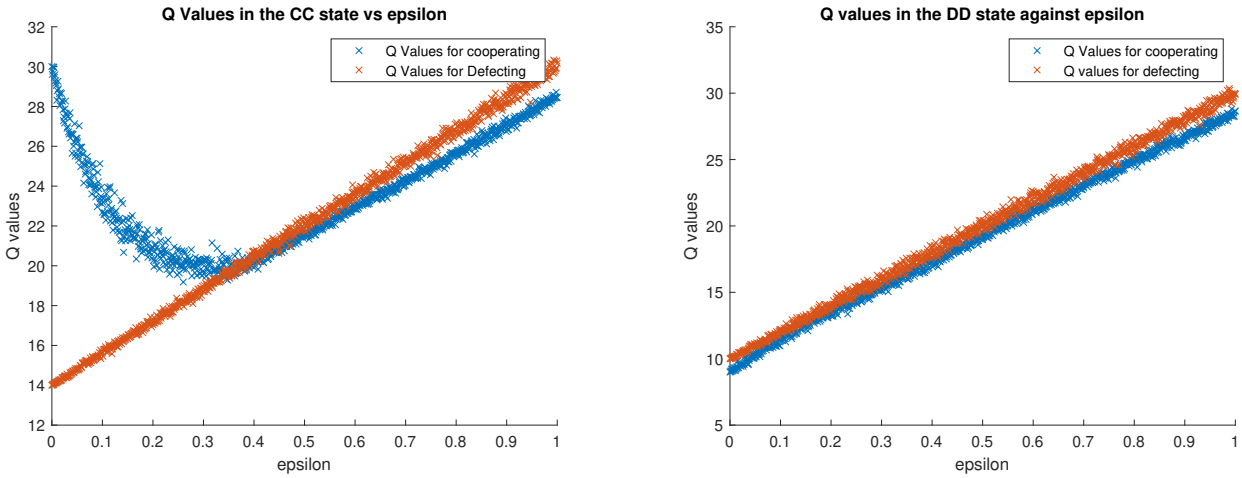
$$\begin{aligned} \mathbb{E}(R_{CC,C}) &= 3(1-\epsilon/2) + 0(\epsilon/2) \\ &= 3 - \frac{3}{2}\epsilon \end{aligned} \tag{18}$$

$$\begin{aligned} Var(R_{CC,C}) &= \mathbb{E}(R^2_{CC-C}) - \mathbb{E}(R_{CC-C})^2 \\ &= (9(1-\epsilon/2) + 0(\epsilon/2)) - (3 - \frac{3}{2}\epsilon)^2 \\ &= \frac{9}{2}\epsilon - \frac{9}{4}\epsilon^2 \end{aligned} \tag{19}$$

The solution to this is to use a lower learning rate. If a high learning rate of 1 is used, the Q table will switch between the reward of 3 and 0 as the opponent acts randomly. However, when a lower learning rate is used, the agent is able to effectively consider more data in its Q value. In Figure 5, a learning rate of 1% is used, which means the noise has been reduced. Lower learning rates can further reduce this noise at the expense of time taken to reach $Q*$.

7.2.2 Playing Against Noisy Grim Trigger

We can see from Figure 6 that as the opponent's ϵ increases, the discounted reward from cooperating in the CC state decreases. This is because the odds of the grim trigger agent defecting randomly increases, which then "activates" the trigger, meaning it will then defect on the next turn too.



(a) CC State

(b) DD State

Figure 6: How the Q values for cooperation and defection vary against noisy grim trigger opponent

The reward for defection in the DD state rises with epsilon, as the expected reward increases linearly with ϵ, as when ϵ is higher, the opponent has a higher chance of cooperating, meaning the agent will earn $T = 5$ points instead of $P = 1$.

Beyond a value of about $\epsilon = 1/3$, it makes no sense to cooperate due to the fact that the chance of the grim trigger activating itself is above $1/6$, so the future reward of cooperating is no longer high enough to outweigh the short term reward of defecting.

We can therefore see that the optimal strategy against grim trigger is to also play grim trigger with $\epsilon < 1/3$, and to always defect otherwise. This is useful, as this means that it is a strategy which is the optimal strategy against itself, which is important when it comes to creating stable strategies.

7.3 Multi-agent Learning with Greedy Agents

We have seen that when an agent is playing against a fixed strategy, the Q values are deterministic, and a learner can learn to play the best response against that strategy. With 2 learners, this problem becomes more difficult, as the actions of an agent affect not only the next action of an agent, but also what it learns. For example, when two agents are in the CC state, if an agent defects, it learns that defecting has

a higher reward. The other agent also learns that the expected return from cooperating might be lower, as it will earn 0 points instead of 3.

The degree of randomness is important in a multi-agent setting. This is because both agents learn through exploration, but each time an agent explores randomly, it is doing something that the other agent is not expecting, and will influence the learning of the other agent. It therefore makes sense to use low values of *epsilon* in multi-agent contexts, as less exploration will lead to more stability due to the non-stationarity of the system from the perspective of both agents.

In this section, we shall be analysing stability of agents when using the Epsilon-Greedy algorithm. This is so that the analysis in the previous section will apply to our learning agents. Using greedy agents also means that each agent has a distinct strategy based on which Q value is higher for each state. This means we can do more interesting analyses of when strategies change and how this evolves.

When considering different strategies in multi-agent settings it is useful to know whether those strategies are stable. Much like in the earlier example, agents will learn optimal responses to the strategy of the opponent. Unlike the earlier examples, when an agent learns the best response to another agent, that agents actions are now impacted by the change in strategy. Both agents are constantly adapting to the new strategies that each of them learns.

7.3.1 Stability of Cooperation

Much the same as when playing against a static agent, learners will quickly learn to defect when their opponent always cooperates. In this Section and henceforth we will use a γ of 0.95, so that agents will heavily weight future rewards. When $\gamma = 0.95$, the Q* values according to Equation 10 are 60 for cooperation and 100 for defection, respectively. We can initialise both learners with a Q value for cooperation of 100, and 60 for defection so that they have Q values of the same order of magnitude as their optimal Q values and so that their initial strategy using ϵ-greedy is to always cooperate. In Figure 7 we can see how the Q values evolve for each agent in the CC state. Initially the agents are earning 150 points per episode, as the episode is 50 steps and the agents earn 3 points per step when they both cooperate. Around episode 80, the Agent 1 has defected enough times for it to have learned that the Q value for defection is higher than that for cooperation, so it starts defecting in the CC state, which we know from the fact that its return is around 200, which is the same as getting the reward of 5 for defecting in CC and the reward of 3 for defecting in DC equally for 50 steps.

Figure 7: Reward per episode for 2 learning agents

At around episode 110, we can see that Agent 2 learns that it should also start defecting in the CC state, as it would earn 1 rather than 0. We can therefore see that in this zone, both agents defect in CC and cooperate in DD, earning an average of around 2 points, to get a total episode reward of 100. The agents defect in DD as this state has not yet been thoroughly explored, so the Q values for DD in both agents are still higher for cooperation as that is how they were initialised.

We can see how the Q values in the CC state evolve for one agent. It is difficult to analyse the evolution of these values at the beginning, as there are lots of sharp changes of gradients that occur either when the other agent adopts a new strategy, or when its own "next best action" changes. It can also take a while to learn, with lots of strategy changes, as it takes time for agents to discover and learn strategies in each state, and those strategies may change the optimal strategies in other states.

From the fact that the best response to always cooperating is to always defect, these results confirm our expectation that pure cooperation is not stable as a strategy, as each agent can learn that they can defect and attain a higher reward, until the strategy changes.

We can observe from Figure 8 how the Q values change rapidly at the beginning as the agents adopt different policies, causing short periods where the agents adopt different policies in the short term. The opponent then learns strategies that counter these short-term strategies. This is because both agents are learning best responses to each other simultaneously, so when each agent learns a new strategy to best respond, the best response to that new strategy now must be learned. We can see that after around 500,000 episodes, the Q values stabilise, with the Q value for cooperation slightly higher than that for

defection.

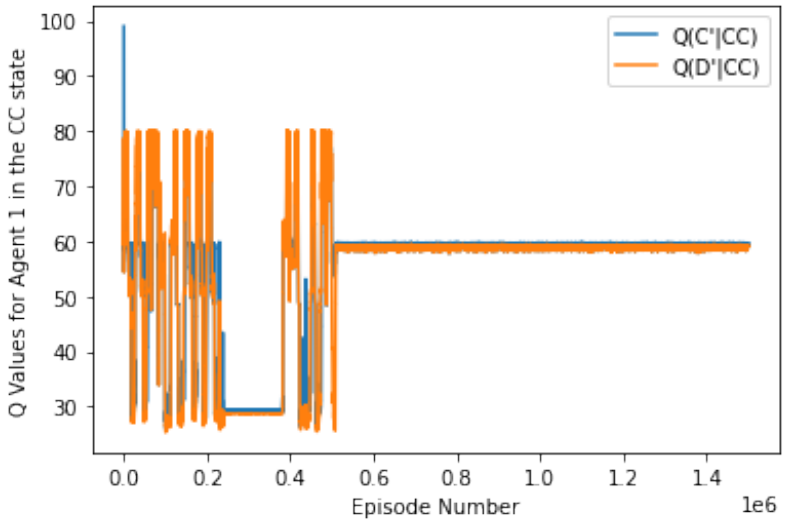


Figure 8: Evolution of Q values for the CC state in Agent 1

7.3.2 Stability of TFT

Unlike pure cooperation, when agents play TFT, they are incentivising other agents to cooperate and punishing defection, which is a useful strategy. However, we know for Equation 14 and Figure 5 that the optimal strategy against TFT is to always cooperate when γ is high, which as we know from Section 7.3.1 also is not stable.

7.3.3 Stability of Pavlov

During experimentation we observed that the long run equilibrium for both agents in many cases stabilised in the Pavlov strategy. This is initially strange, as we know that the best response to Pavlov is to always defect, so the agents should learn that response to each other. However, when we analyse the Q values in the CC state in Figure 9, we can see what is happening.

When $Q(CC, D)$ for an agent surpasses $Q(CC, C)$ as it learns the benefit of defection, the Q values for both agents start to drop, as their rewards go down from being usually 3, to usually 1. During this drop, both states are being explored, as when one Q value falls, the other will be higher. The fall stops when both agents cooperate with each other and get the reward of 3, which makes $Q(CC, C)$ higher than $Q(CC, D)$ for both agents.

$Q(CC, C)$ then rises rapidly back up to its steady state value for both agents, as $Q*$ is restored to its

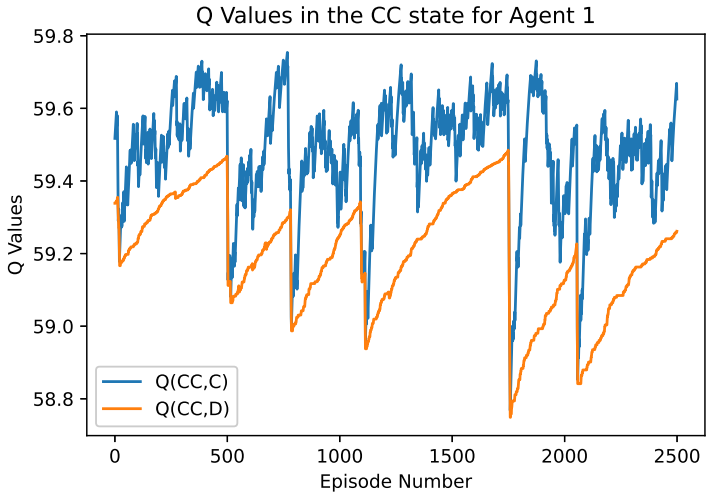


Figure 9: Q values in CC state in the Pavlov strategy equilibrium

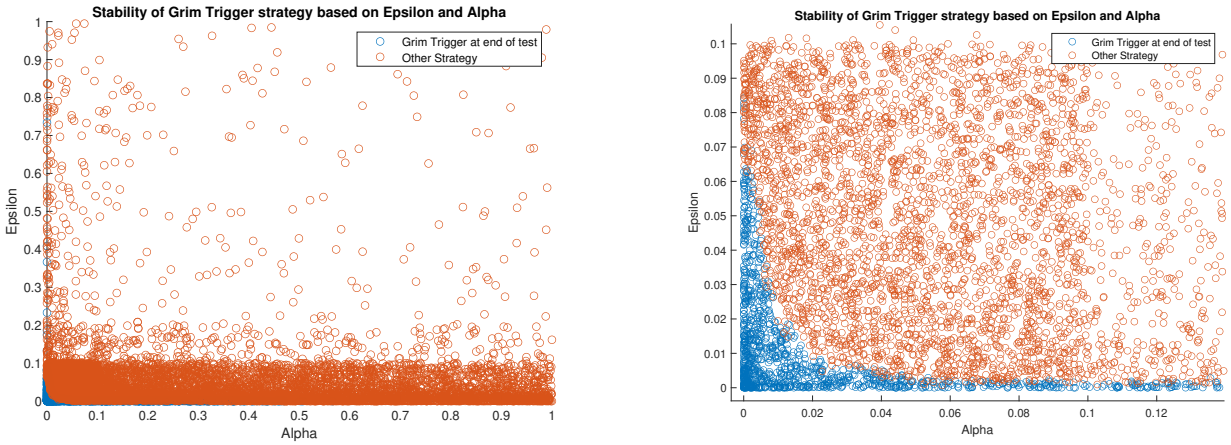
original value as both agents are cooperating. Therefore, this equilibrium does not occur because Pavlov is the best response to itself, but because both agents are learning online, which means that more complex interactions can occur and can be stable.

In this steady state, the same kind of alternating equilibrium is occurring in the DD and Initial states, where cooperation is usually the dominant strategy but periodically both Q values for these states drop when the Q value for defection catches up to the Q value for cooperation.

As a long run equilibrium, this strategy makes sense when two noisy learning agents are playing against each other. Whereas in Grim Trigger, when an agent randomly defects the rewards for both agents for the rest of the episode will be P, which is the punishment reward of 1. In Pavlov, the agents will punish a defection with a defection of their own, but when they both defect, both agents will switch to cooperating. This means that the agents can reenter the CC equilibrium of earning R each, the Reward of 3. Similarly to Tit-for-Tat, Pavlov is a strategy that discourages the opponent from defecting by defecting itself on the next turn.

7.3.4 Stability of Grim Trigger

As mentioned in Section 7.2.2, Grim Trigger is the optimal strategy against Grim Trigger when the chance that the opponent chooses randomly is less than 1/3. One might expect Grim Trigger to remain stable as a strategy when $\epsilon < 1/3$ for each agent. However, we can see in Figure 10 the stable values are actually in a much smaller range than that, and also depend on α.



(a) All values (b) Small values

Figure 10: Stability of Grim Trigger after 10M episodes of 10 steps each

We can see that the boundary between stability and instability is similar to a hyperbola, as can be seen in Figure 11.

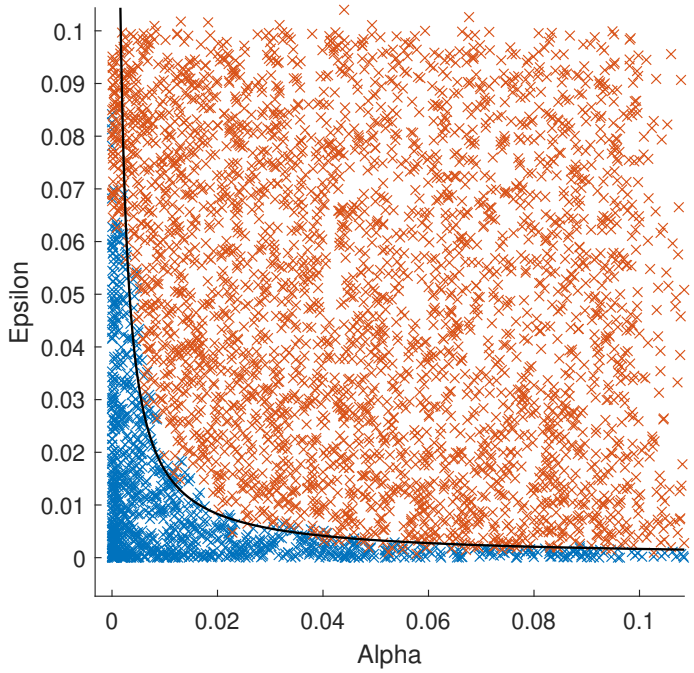
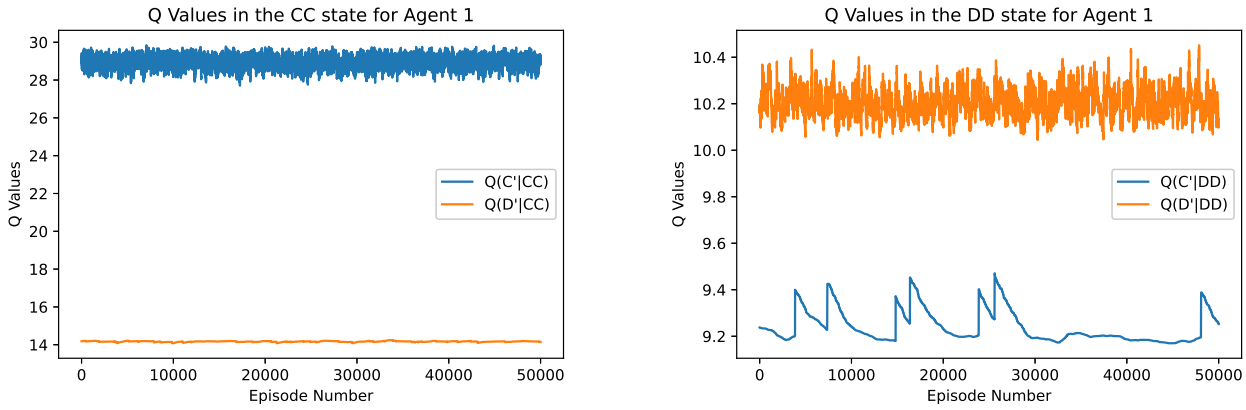


Figure 11: Stability of Grim Trigger after 10M episodes with Decision Boundary

To understand why Grim Trigger isn't stable at the same values of ϵ that we expect, we can analyse how the Q values evolve over time. In the CC state, the steady state Q values at $\gamma = 0.9$ are around 30 for cooperation and around 14 for defection. In the DD state the Q values for those actions are 9 and 10 respectively.



(a) CC Steady State

(b) DD Steady State

Figure 12: Grim Trigger strategy steady state Steady State. $\gamma = 0.9$

We know that when the agent is in the steady state at DD, if it cooperates it will either earn a reward of 0, and stay in the DD state with the max Q on the next turn being 10, or if the other agent also randomly cooperates it earns a reward of 3 with the next best Q value being 30. This is clearly a lot higher than the steady state value, so when this happens, it causes a large jump in $Q(DD, C)$, which is demonstrated in Figure 12b.

We can also observe that after each mutual cooperation, $Q(DD, C)$ will decay exponentially as described in Section 5.1.1.

We can calculate from Figure 12 that the discounted reward when both agents cooperate in the CC state is equal to $3 + 0.9 * 29 = 29.1$. The TD error between this and the steady state of Q(DD,C) of around 9.2 is 19.9. To change strategies to cooperation, Q(DD, C) must exceed Q(DD,D), which is around 10.2, an increase of 0.9. From this, we therefore know that if $\alpha > 4.5\%$, a single instance of mutual random cooperation in the CC state will cause both agents to switch strategies to Pavlov, and will then both continue to cooperate in the DD state, quickly reaching the steady state of 29.1 as shown in Figure 13.

When α is lower than 4.5%, it will take multiple mutual cooperations for Q(DD,C) to exceed Q(DD,D). The number of mutual cooperations is defined by $n = \frac{0.9}{19.9\alpha}$

We can see in Figure 10b that some stable values still exist above this value of α, however these are very small values of ϵ, meaning that in these trials there was never an instance where both agents randomly cooperated in the DD state. The probability of both agents cooperating in the DD state is $\frac{1}{4}\epsilon^2$, as when the random action is selected with probability ϵ for each agent, there is a 50% chance that the agent

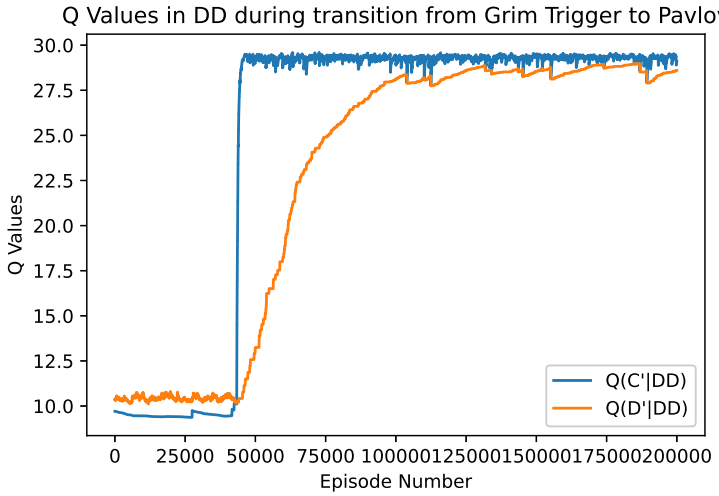


Figure 13: Q Values in the DD state as Grim Trigger transitions to Pavlov

will still randomly choose to defect. We therefore know that if the number of episodes is increased, the boundary between stability and instability moves closer to the origin, as there is more time for this improbable event to occur.

We can also see that the value of ϵ makes a difference in stability. When ϵ is lower, the random mutual cooperations are less frequent, so the probability of a run of mutual cooperations occurring such that Q(DD,C) reaches Q(DD,D) decreases, as it becomes more likely that the values will decay faster than mutual cooperations increase them when ϵ is lower.

The stability also depends on γ. This is because we know that the discounted return for mutual cooperation is equal to $\frac{3}{1-\gamma}$, and the reward for mutual defection is $\frac{1}{1-\gamma}$. The reward for cooperating when the opponent defects is defined as $\frac{\gamma}{1-\gamma}$, As defined above, for the strategy to change in one mutual cooperation, $\alpha\times$TD Error $> Q(DD, D) - Q(DD, C)$. Therefore:

$$\alpha > \frac{1-\gamma}{3-\gamma} \tag{20}$$

This means that as γ approaches 1, the α required for a singe mutual cooperation to change the strategy approaches 0. The other way to describe this is to say that given a constant α, as γ increases, the number of mutual cooperations in DD required to change the strategy decreases.

It is important to note in this analysis that the reason for the instability of Grim Trigger is that both agents learn the new strategy simultaneously. If we played the game when one agent learns in an episode

and the other agent is static, with both agents alternating between static and learning, then this would not happen and the Grim Trigger strategy would be stable for far greater values of ϵ and α. This is because the strategy changes only when Q(DD,C) exceeds Q(DD,D) for both agents. If not, Q(DD,C) will then decrease below Q(DD,D) for the agent that passed it, as it will quickly learn that it will be punished for cooperating by the opponent which will still choose defection. The long-term strategy change therefore only occurs when both agents learn to cooperate simultaneously

7.3.5 Stability of defection

Lastly, we can analyse the stability of pure defection. Like Grim Trigger, pure defection is a strategy that is the best response to itself. However, we can see that it is far more stable. The reason for this is that for the CC state to enter equilibrium with both agents cooperating, they must both cooperate in the CC state to increase Q(CC,C). When ϵ is the, the frequency of this occurring is extremely low, as it requires two subsequent random mutual cooperations. The probability of this occurring is $\frac{1}{16}\epsilon^4$, which means that this is a rare scenario.

Furthermore, when a mutual random cooperation does occur, the increase in Q(CC,C) is much smaller than the situation in the DD state in Grim Trigger, as the next best state is the same value for both cooperation and defection, so the TD error is only around 3, with the needed change in Q(CC,C) being around 1. This means that if $\alpha > \frac{1}{3}$, then if this randomly occurs once, then a new equilibrium of cooperation at CC can dominate, meaning that stability is not guaranteed.

In practice, much like in grim trigger, pure defection is stable across a smaller range of values of the hyperparameters. For example, using $\epsilon = \alpha = 5\%$, the agents break out of pure defection and assume a modified Pavlov strategy, in which the agent cooperates in CC and DD, and defects in the CD, DC and Initial states. This is shown in Figure 14. This strategy isn't fully stable, but is part of a semi-stable loop of strategies which it switches between. This is further explored in the next section.

7.3.6 Conclusion of stability of greedy agents

We have seen from these examples that the learning of agents is heavily dependent on the hyperparameters used. Even when strategies are theoretically stable in self-play, often they are not as stable when agents are actually learning when certain hyperparameters are used.

With this knowledge, we can now use this information to shape the learning of agents who are initialised

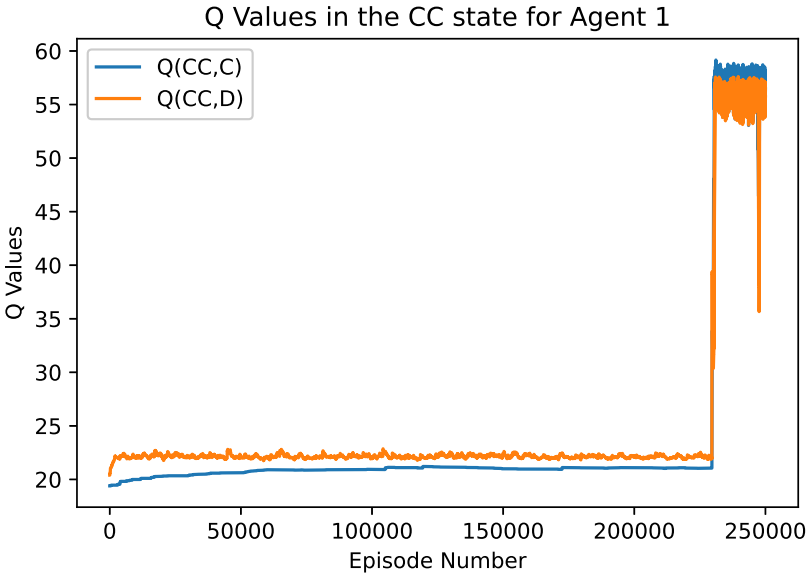


Figure 14: Q Values in the CC state for an agent initialised to always defect. $\alpha = \epsilon = 5\%$

with no existing strategies or knowledge of the game.

7.4 Average Rewards in Different Strategies

Another useful experiment is the analysis of the average reward of the Always Defect, Grim Trigger, Pavlov, and Tit-for-Tat strategies when two noisy agents play against each other. The average rewards per step for these strategies over different episode lengths and different values of ϵ can be found in Table 12. This selection was chosen as Always Defect, Grim Trigger and Pavlov are at least partially stable, and TFT is one of the most common strategies researched in the IPD.

	Always Defect	TFT	Grim Trigger	Pavlov
$\epsilon = 1\%$	1.01	2.97	2.91	2.98
$\epsilon = 2.5\%$	1.04	2.93	2.78	2.95
$\epsilon = 5\%$	1.08	2.86	2.60	2.89

(a) 10 step episode length

	Always Defect	TFT	Grim Trigger	Pavlov
$\epsilon = 1\%$	1.01	2.88	2.58	2.98
$\epsilon = 2.5\%$	1.04	2.72	2.16	2.94
$\epsilon = 5\%$	1.08	2.55	1.79	2.88

(b) 50 step episode length

Table 12: Average reward per step for greedy agents using different strategies

We can see that, as expected, the average reward per step for the “Always Defect” is the lowest, as both

agents attain the "Punishment" reward of P = 1 most turns. Occasionally one of them cooperates and gets 0 while the other gets 5 points. This is why we see that the average reward increases with ϵ.

The average reward in TFT is much closer to 3, as most of the time agents cooperate and both earn R = 3. When an agent randomly defects, both agents will then alternate between defection and cooperation, and will therefore alternate between T = 5 and S = 0 points, which is an average reward of 2.5. When episodes are longer, this has more of an effect, as agents are likely to spend a higher proportion of the episode in the DC and CD states, as the probability of always staying in CC is $(1-\epsilon)^{2n}$ when n is the number of steps.

This has an even greater effect in Grim Trigger. When one agent randomly defects, both agents will then subsequently defect for the rest of the episode, earning P = 1 each. The longer the episode, the higher the proportion of episode they are likely to defect for. For example, an episode with 1000 steps has an average reward of 1.2 per step when $\epsilon = 1\%$.

We can contrast these strategies with Pavlov. In Pavlov, the result of a random defection is initially that both agents defect on the turn after the random defection. After that turn, both agents will cooperate once more and will continue to do so until the next random defection. This means that they will only break from both earning R = 3 points for 2 turns, so the effect of the defections is comparatively small. Returning to the CC state also means that the average reward per step is more independent of episode length. If random defections occur at a rate of ϵ (assuming ϵ is small and that both agents defect at a rate of $\frac{\epsilon}{2}$), then Pavlov can expect to be in the DD state for around $\epsilon \times n_{steps}$ steps, and the DC and CD states for $\frac{\epsilon}{2} \times n_{steps}$ each. Using these estimates, we can calculate that the expected reward for $\epsilon = 1\%$ at 50 steps is:

$$\begin{aligned} R_{avg} &= 3(1-2\epsilon) + 1\epsilon + 5\epsilon/2 + 0\epsilon/2 \\ &= 2.98 \end{aligned}$$

This confirms that this approximation holds for low values of ϵ and longer episodes. The result is slightly inaccurate for shorter episodes, as when the CD or DC states occur on the last step, there is no DD state afterwards, so DC and CD combined occur more regularly than the DD state, but the longer the episode, the smaller this effect. This is why the reward is slightly lower for the 50 step episode than for the 10

step episode.

We can also see a trade-off in the choice of ϵ. When ϵ is higher, the always defect strategy is not stable, and leads to a form of Pavlov, which has a far higher reward. However, higher values of ϵ also leads to lower average rewards as shown in Table 12.

We have demonstrated that in this context, the Pavlov strategy leads to the best outcome for agents. While Pavlov works well in this multiagent setting, it is a strategy that is very easy to exploit. However, that doesn't affect its viability as a long run strategy in this situation, due to the stability it has demonstrated in Section 7.3.3. This is therefore a narrow solution to the problem as it results in cooperation between these specific learning agents, but if one agent weren't learning it would not be stable.

7.5 Transitions Between Strategies

In this section, we will investigate the transitions as learning agents both learn and transition between different strategies when agents are initialised with all Q values being 0. We know that as agents learn, their opponents must therefore learn different strategies as the best response. We can therefore plot how strategies evolve.

An important choice for this analysis is the choice of agent hyperparameters. I have chosen to use $\gamma = 0.95$ for both agents, as this is sufficiently high that agents will more heavily weight future rewards, encouraging cooperation. I also chose to test the values $\alpha = \epsilon = 1\%$ and compare them with the results for $\alpha = \epsilon = 5\%$. These two sets of hyperparameters will be henceforth referred to as low and high exploration.

These values were chosen because the experiments in Section 7 demonstrated that a lower random exploration rate leads to higher rewards in the Pavlov strategy, and further experiments showed that a 5% value for these parameters caused pure defection to be unstable. Both sets of values must therefore be tested and compared.

7.5.1 Transitions for a Single Initialisation with Low Exploration

Figure 15 shows the strategies explored in one trial that was initialised at 0. In this figure, each node represents the strategies of both agents. For example, the dark green node represents the state when both agents use the Pavlov strategy. For this section, nodes shall be labelled using 10 letters, 5 letters for each agent, with each letter representing their action in the states CC, CD, DC, DD, Initial. For example, the Grim Trigger strategy for both agents would be "CDDDC CDDDC", as both agents only cooperate in

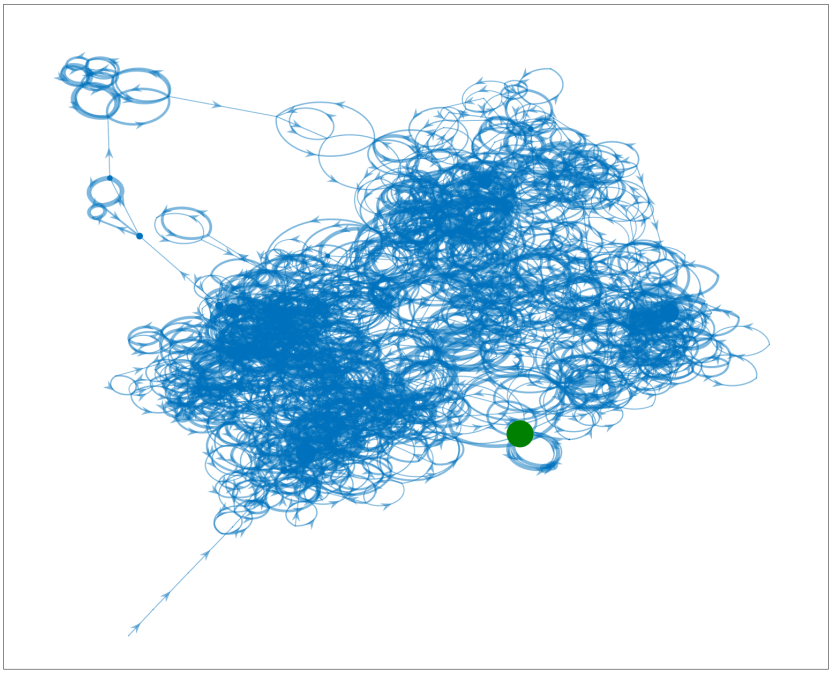


Figure 15: Diagram showing transitions between different strategies over one training cycle

CC and Initial.

The lines with arrows represent the number of steps in which the agents started a step with one strategy set and learnt new strategies after the end of the step. The thickness of the lines represents how often each transition occurred. We can see how the initialisation node is at the bottom left of the diagram, where both agents start to learn different strategies as their Q values change.

A closer inspection of the diagram shows us the most common loops that occur. Figure 16 shows us a loop where an agent in the Pavlov strategy learns to defect in the CC state. This is consistent with what we observed in Figure 9 in Section 7.3.3, where Q (CC, D) "catches up" with Q (CC, C) for an agent, and then both begin to decrease rapidly for both agents. We can then see strong lines connecting the "DDDCC" for both agents node with the "CDDCC for one agent and DDDCC for the other" nodes. This describes how both agents can learn these different strategies as both Q values decrease in Figure 9. We can also see that the most common way for both agents to return to Pavlov is from the "DDDCC for both agents" node, showing that the cycle returns to Pavlov when both agents defect such that Q(CC,C) is greater than Q(CC,D) for both agents.

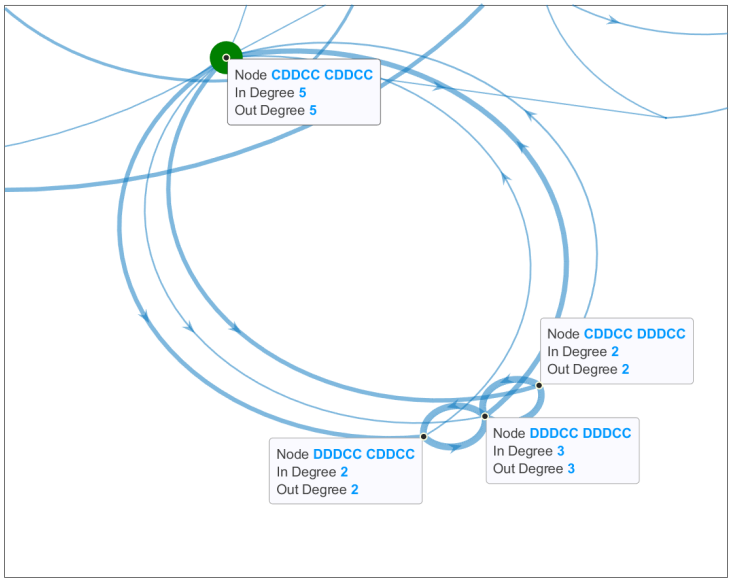


Figure 16: Diagram showing transitions between different strategies

We can also observe from the larger size of the green Pavlov node that this is the most common state in this system, with strategies lying at other nodes less frequently.

7.5.2 Analysis of Low Exploration Multiagent Strategy Exploration

In addition to plotting how agents transition between strategies over one initialisation, we can repeat the experiment and combine the plots to find the transitions and values that can be reached. Figure 17 shows the same diagram for 1,000 initialisations of 500,000 episodes each. Episodes of 50 steps were chosen so that the rewards were less biased toward strategies that cooperate on the first turn, and so random exploration would allow for the more thorough exploration of states.

It is clear that the system is very chaotic, as it can be seen that many strategy sets can evolve into many other strategy sets, rather than a set linear path of learning.

We can also observe how these different strategies lead to different rewards for each agent. Figure 18 shows the same diagram, but with the placement of the nodes based on the average reward per step that each agent experiences in a 50-step episode.

We can see that various distinct regions of points exist. To the top-left, strategies in which Agent 2 usually defects and Agent 1 usually cooperates are seen, with the opposite occurring to the bottom-right.

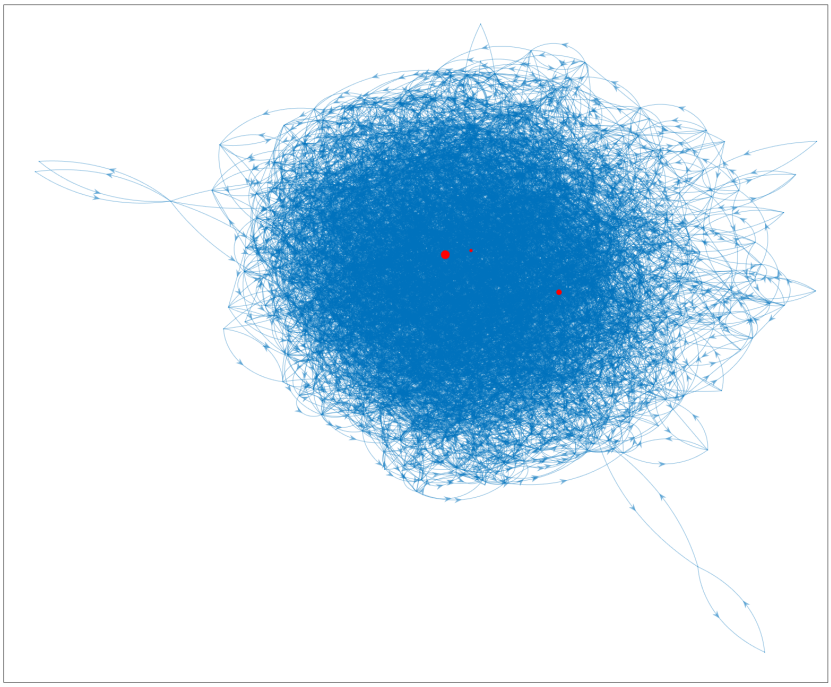


Figure 17: State Transitions over 1000 Initialisations. The red points are (from left to right) Pavlov, Grim Trigger, and Always Defect

We know this, because the average reward per step of 5 only occurs when the the agent defects in every turn. Around the (1,1) point, we have the region in which both agents usually defect, earning P=1 in most turns. To the top right we have the cooperative strategies, where agents can both earn around 3 per turn. As we expected from the calculations in Section 7.4 , the average reward for Pavlov agents is very close to (3,3), with both agents earning an average of 2.98 each per step.

We can also observe that other points exist between these. This occurs when loops between different states happen due to the strategy. For example, when both agents cooperate in DD and defect in CC, the average reward per step will be 2 for each agent, an average of P=1 and R=3. These points can be observed at the (2,2) point in Figure 18. These averages exist for all possible loops, so if the strategy leads to the agent alternating between CC and DC, then the average rewards will be 4 and 1.5 respectively. Longer loops also exist, so if the strategies resulted in the loop: CC-DD-DC-CC, then Agent 1 will earn an average of 3 and Agent 2 will earn an average of 1.33. This corresponds to the set of points existing around (3,1.33) Figure 18

The agent's response to exploration matters. This means that an agent such as Pavlov that responds to

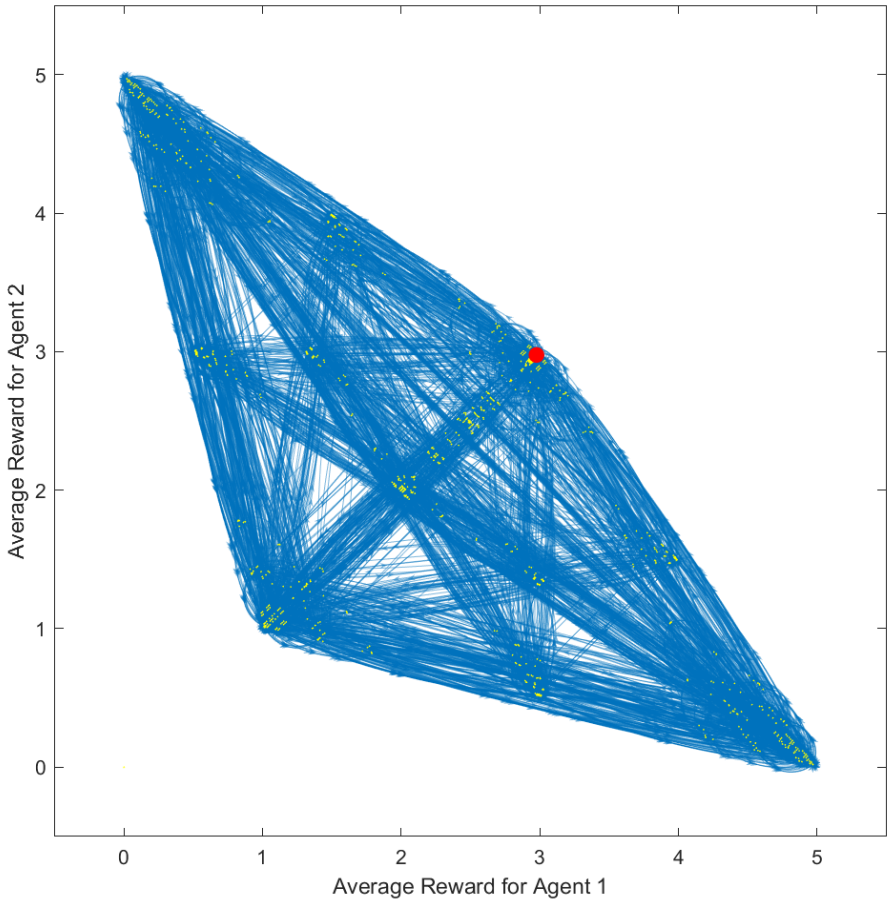
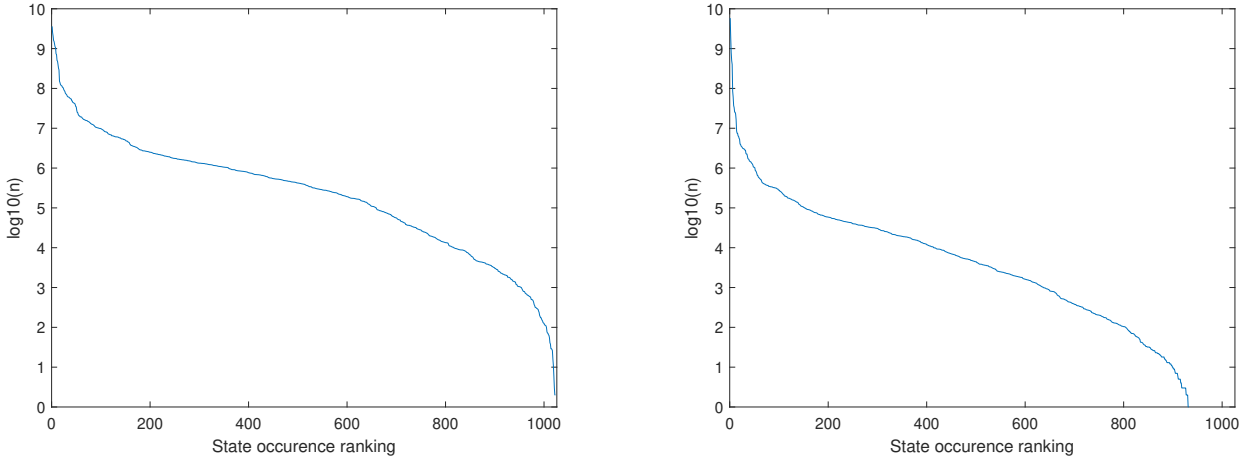


Figure 18: Different strategy pairs arranged by average agent rewards. Strategy pairs are in yellow, links between them in blue, and the red node is when both agents use Pavlov

exploration well has a higher average reward for both agents than Grim Trigger, which does not, due to its lack of forgiveness. We also have variation due to different initial states. The reward after the first 2 steps in Pavlov is independant of whether agents defect or cooperate on the first turn, as they will return to the CC equilibrium. Therefore if both agents defect initially then the reward for both will be slightly lower than for pure Pavlov, as this step slightly brings down the average.

The first point of note is that the histogram in Figure 19a shows us that almost all states are visited at least once. In fact, only two strategy sets did not occur. We can also see that the vast majority of steps occur in a few strategy sets, with most strategy sets existing between 10^4 and 10^7 steps, while the largest value existed for over 10^9 steps. This implies that almost every single strategy set is transitory and cannot exist in a stable way in the long run.



(a) All steps

(b) Transitions for steps past 1M episodes

Figure 19: Histogram showing for how many steps each set of 1024 strategies exist for. Note the logarithmic scale.

One problem with this method of visualisation is that temporal information is lost. We observe in Figure 19b that the results found when Q learning was run for 2M episodes, with only transitions that occurred after 1 million episodes being counted. This experiment was also run with more episodes per test, so the number of total steps analysed remained somewhat consistent. It is clear that even after 1M episodes, stability is not guaranteed with the chosen curriculum and hyperparameters, as most strategy sets are still visited. It is however clear that the histogram drops off faster after one million episodes, showing that the Q learners spent the majority of steps using only a handful of strategies.

Another useful method for visualisation is to combine all nodes together which exist for fewer than a certain number of steps. An example of this is shown on Figure 20, which is taken from the same data used to plot Figure 17.

Figure 20 shows how the larger nodes have most interactions with many smaller nodes, rather than with each other. We can also see that Pavlov (labelled with a green node) is the strategy pair that occurs in the most steps. This is encouraging as it shows that the goal of finding emergent reciprocity is very possible using these hyperparameters with our reward structure.

Analysing this diagram, we can see that there are 3 distinct regions. The first is at the top and top left of the diagram, where there are more stable long-term strategies. We can then also see a region in the bottom left where Agent 1 is likely to defect and Agent 2 is likely to cooperate, meaning that Agent 1 is exploiting Agent 2. To the right of the diagram, we can see the reverse of this occur, with agent 2 being

more dominant.

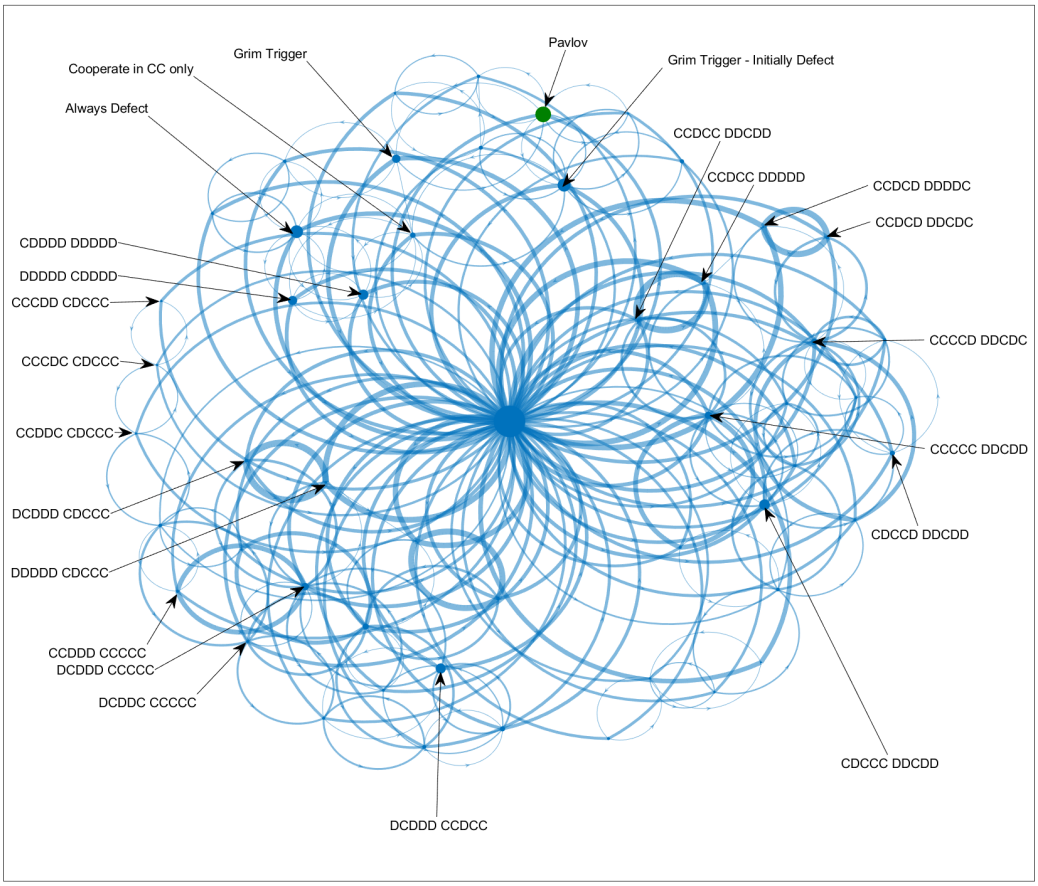


Figure 20: State Transitions with smaller nodes combined, and a selection of nodes labelled

Figure 21 shows that once the first million episodes have been removed from the data, the Pavlov node is now much larger compared to the combined central node of other strategies. This shows that most of the time agents learn this strategy. For example, we can see that the Pavlov strategy has relatively strong outwards connections to "CDDCC DDDCC" and "DDDCC CDDCC", which both have strong connections both ways with "DDDCC DDDCC". This then has a strong connection back to Pavlov. This is the exact same loop that was found in Section 7.5.1, showing that this is a frequently occurring end-state.

To see the rewards each agent gets from these common strategies, we can arrange the nodes as in Figure 18, so that their position depends on the average reward for each agent. This can be seen in Figure 22. We can see in this plot that the parameters we have chosen, as well as the algorithm used to explore, has

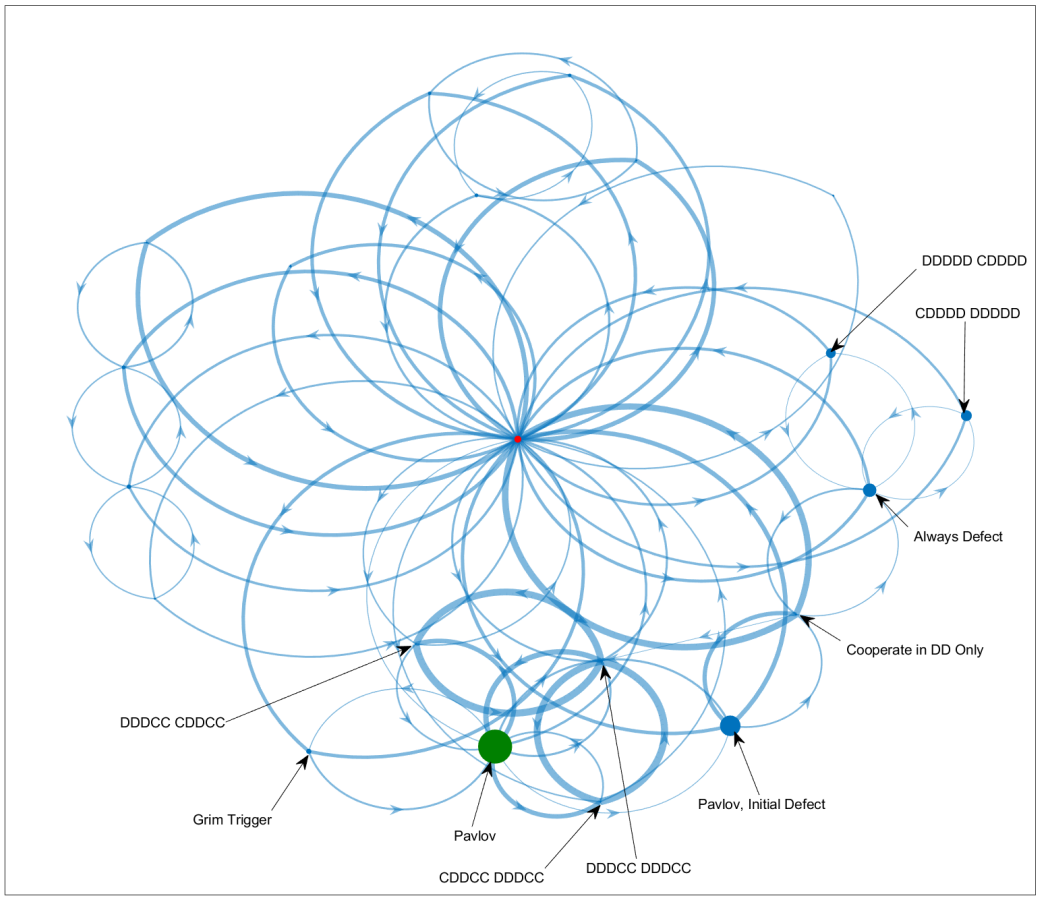


Figure 21: State Transitions after 1M episodes with smaller nodes combined, and a selection of nodes labelled

created an outcome which leads mostly to cooperative outcomes.

It is important to note that in Figure 22, the "Other Strategies" node is placed at (0,0) for data visualisation purposes. In reality, it is made up of a large number of strategies, all of which have different rewards for each agent.

This shows us that while nodes to exist in which one agent earns a high reward and the other earns a low reward, these are transitory, as the nodes are small. This is because any strategy that exists in a stable way has a larger node, as the agents stay in the strategy set for longer periods.

The visualisation shows us better that while in Pavlov, and the strategy sets that are learned from and return to Pavlov, rewards for both agents remain relatively high. Even when both agents learn to defect

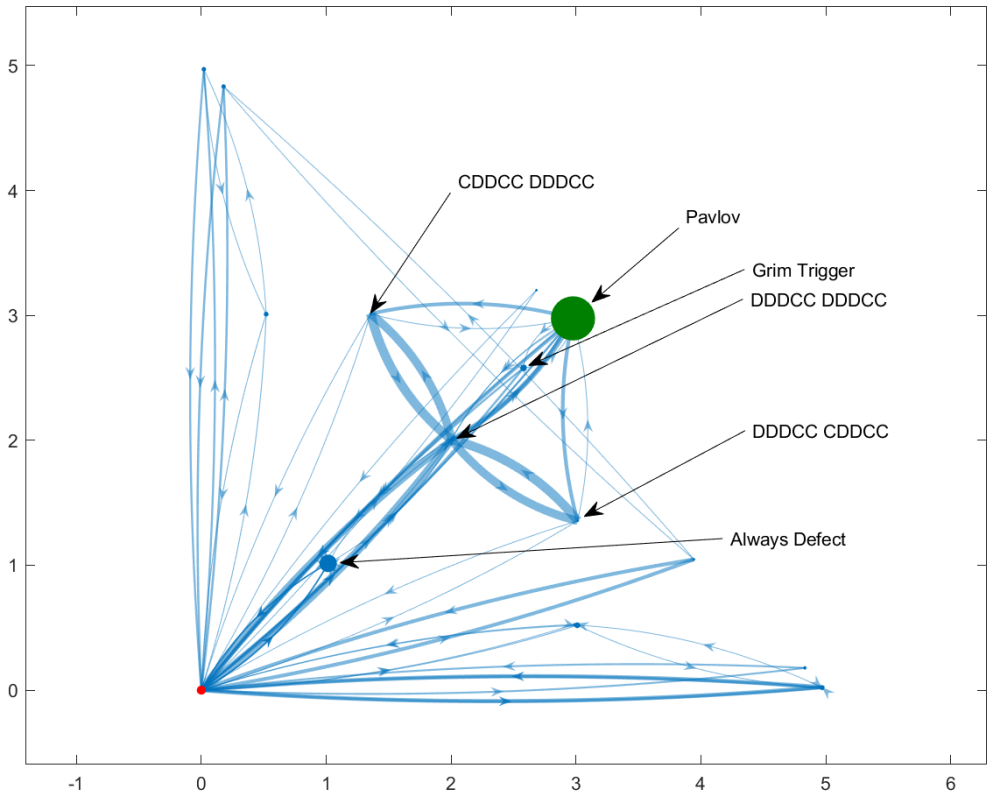


Figure 22: Caption

in CC, the average reward for both agents is 2, which is better than always defecting. These strategies are only transitory, and occur for brief periods compared to Pavlov, as can be deduced by the far smaller node sizes. This shows that the overall average reward will be much closer to 3 for the two agents. This also confirms the hypothesis developed over the smaller-scale test in Section 7.3.3, which states that the Pavlov strategy is stable in the long run, except for brief periods where $Q(D, CC)$ and $Q(C, CC)$ decrease rapidly, causing the greedy agents to switch between "CDDCC DDDCC", "DDDCC CDDCC" and "DDDCC DDDCC".

7.5.3 Analysis of High Exploration Multiagent Strategy Exploration

When the exploration level is higher, the dominant strategies change. We see in Figure 23 that the Pavlov strategy, highlighted in green, is significantly less dominant than in low exploration agents. It is mostly replaced by the node in which both agents engage in the "CDDCD" strategy, which is the same as Pavlov but with an initial defection from both agents. This has a slightly worse outcome, as both agents lose 2

points in the initial state when compared to Pavlov, but this is not significant in the long run.

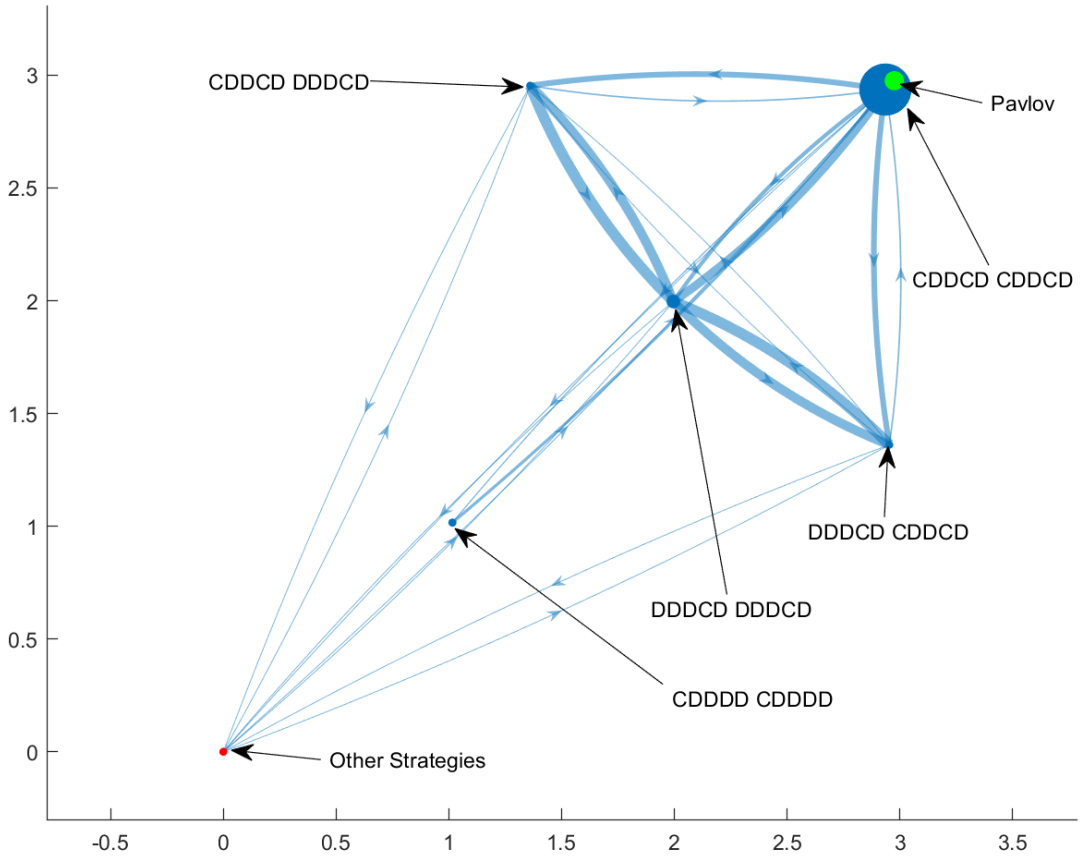
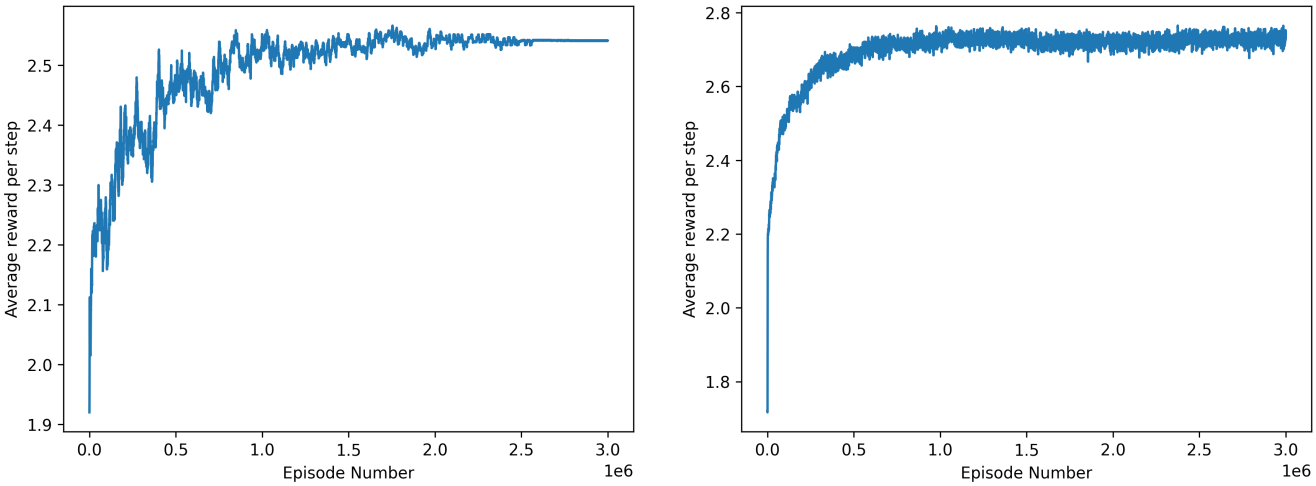


Figure 23: Strategy Transitions for high exploration agents

We can see that in high-exploration agents, the occurrence of the "always defect" strategy reduces so much that the node is not significant enough to be displayed. The only displayed node with a strategy in which defection dominates is the "CDDDD CDDDD" node; however, this is not stable for the same reason that Grim Trigger is not stable, so it will also decay as the number of episodes increases.

We do notice that the nodes in which the agents learn to defect in the CC state have a slightly higher weighting than in the low exploration agents. This makes sense, as the higher exploration and learning rates mean that Q(CC,D) will increase faster, causing more frequent changes in strategy from the modified Pavlov strategy that is usually present.



(a) Low Exploration Agents

(b) High Exploration Agents

Figure 24: Average reward per step for learning agents across 200 initialisations

7.5.4 Rewards earned by greedy Q learning agents in the IPD

Plotting average reward across two hundred initialisations in Figure 24, we can see that the learning agents quickly become more cooperative on average than when they are initialised with no knowledge of the system. We can see that at the start of learning, the average reward is around 2, showing that the balance between cooperation and defection is fairly even, as the average reward between mutual cooperation and mutual defection is 2. We can see that as the agents learn, the average rewards increase to over 2.5, showing more learning allows the agents to find more cooperative strategies which are stable.

We can see that the high exploration agent reaches a steady state faster than the low exploration agent. This makes sense, as they have a higher learning rate so explore strategies faster and therefore reach the steady state faster. We can also see that the high exploration agent reaches a higher steady-state reward per step. This confirms that the higher exploration means that agents do not enter and stay in adversarial strategies, and instead pursue cooperative strategies. It also shows that the decrease in the average reward per step in the Pavlov strategy when ϵ increases, as discussed in Section 7.4, is outweighed by the increased chance of entering a cooperative strategy.

This is a very significant result, as it shows that the dominant force in this system is emergent reciprocity, which is what this project was aimed at finding. We have also shown how the Pavlov strategy is stable in the multiagent context and how it dominates other strategies in terms of the frequency at which it occurs when agents are initialised with no knowledge of the system.

We have also proven that we can find cooperation more consistently when using agents with high exploration, which is an unexpected result given that increased exploration in many contexts leads to increased defection as mutual trust is inherently lower.

8 Conclusion

Within this report, I have demonstrated how Q learners can be used to find optimal strategies against other agents in the Iterated Prisoner's Dilemma. We have proved that these agents can adapt to randomness in other agents, and how this randomness affects the discounted values of each state-action pair.

We have also described how the system changes when two learning agents are used instead of one. We have proved that most sets of strategies are not stable when used by greedy learning agents, even in some cases when they are the best response to themselves in self-play. We have investigated the effect of hyperparameters on this kind of stability and selected values for them to maximise the Chance of cooperation. We showed that increasing the exploration and learning rates leads to less stability in many strategies, even when those strategies are the best response to themselves, an interesting quirk of using greedy agents.

Finally, we have shown how the strategies of the greedy learning agents evolve over time, and which strategies occur the most. Through this, we have shown that the dominant strategies that evolve are mostly cooperative, as shown through the steady-state average reward of over 2.7 when high exploration agents are used. This is an important result, as many papers in this field deal with the response of stochastic rather than greedy agents, so this approach is more novel.

Therefore, we have shown that greedy agents using the correct hyperparameters are capable of emergent reciprocity. Furthermore, the instances of emergent antagonism in which agents learn to defect and antagonise were scarce to the point that no trials of the high exploration agent encountered it. This was due to the fact that strategies that value defection are not stable when exploration and learning rates are increased.

The most surprising aspect of our results is that the dominant emergent strategy is Pavlov, which is a strategy that is both unstable in self-play and easily exploitable by an opponent always defecting. The result of having a stable loop from Pavlov to agents learning to defect in the CC state, to the agents then learning more and returning to Pavlov is particularly interesting. Furthermore, we have proved that

Pavlov has higher average rewards for both agents than other strategies that encourage cooperation, such as Grim Trigger and Tit-for-Tat. The agents have therefore found a more optimal solution than originally hypothesised, as encouraging Tit-for-Tat was the original goal of this project.

8.1 Potential Future Work

Although this report focused on two sets of values for learning and exploration rates, more research into how the long-term reward and strategies vary across more values of ϵ and α would prove useful, as in this report only two different sets of hyperparameters were investigated for their effect on strategy transitions.

Although this report showed how the use of certain hyperparameters in agents leads to a cooperative outcome, we also know that the high exploration makes the average rewards of these strategies lower. Creating a schedule that reduces the exploration and learning rates of these agents could help the agents be able to always end in the cooperative state like in high exploration agents, as well as to increase the rewards in that state like the low exploration agents. This could also help agents to converge to this steady state strategy faster.

In this report, we have only researched behaviour in Q learners using 1-step history. An obvious extension to this project would be to analyse how agents would respond to using a longer history, such as 2 steps. Research into interactions where one agent has a longer memory than the other could also lead to interesting results, as this longer history could allow one agent to more effectively shape the strategy of the other agent.

Another avenue for exploration would be to vary the payout matrices during training to encourage cooperation. Whilst we proved that emergent cooperative behaviour is dominant, research into shaping the agents into a desired cooperative strategy would certainly prove useful. This could be done through the use of deep learning, in which the deep learner is rewarded when agents adopt a cooperative strategy.

References

[1] Hanchao Wang, Hongyao Tang, Jianye Hao, Xiaotian Hao, Yue Fu, and Yi Ma. Large scale deep reinforcement learning in war-games. In *2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1693–1699, 2020.

[2] Sarah Legge. Cooperative lions escape the prisoner's dilemma. *Trends in Ecology & Evolution*, 11(1):2–3, 1996.

[3] John A. C. Conybeare. Public goods, prisoners' dilemmas and the international political economy. *International Studies Quarterly*, 28(1):5–22, 1984.

[4] J. Y. Wakano and N. Yamamura. A simple learning strategy that realizes robust cooperation better than pavlov in iterated prisoners' dilemma. *Journal of Ethology*, 19(1):1–8, 2001.

[5] Grofman and Pool. How to make cooperation the optimizing strategy in a two-person game. *The Journal of Mathematical Sociology*, 5(2):173–186, 1977.

[6] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. *Machine Learning Proceedings 1993*, page 330–337, 1993.

[7] Weijun Zeng, Minqiang Li, Fuzan Chen, and Guofang Nan. Risk consideration and cooperation in the iterated prisoner's dilemma. *Soft Computing*, 20(2):567–587, 2014.

[8] Jakob N. Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness, 2017.

[9] Alessandro Bravetti and Pablo Padilla. An optimal strategy to solve the prisoner's dilemma. *Scientific Reports*, 8(1), 2018.

[10] William Poundstone. *Prisoner's dilemma*. Oxford University Press, 1993.

[11] A. W. Tucker and Philip D. Straffin Jr. The mathematics of tucker: A sampler. *The Two-Year College Mathematics Journal*, 14(3):228–232, 1983.

[12] Robert Axelrod. *The Evolution of Cooperation*. Basic Books, 2006.

[13] Kun Shao, Zhentao Tang, Yuanheng Zhu, Nannan Li, and Dongbin Zhao. A survey of deep reinforcement learning in video games, 2019.

[14] Tuomas W. Sandholm and Robert H. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37(1):147–166, 1996.

[15] M. Broom, C. Cannings, and G. T. Vickers. Multi-player matrix games. *Bulletin of Mathematical Biology*, 59(5):931–952, 1997.

[16] Steven Kuhn. Prisoner's dilemma. *Stanford Encyclopedia of Philosophy*, Apr 2019.

[17] Wolfgang Slany and Wolfgang Kienreich. On some winning strategies for the iterated prisoner's dilemma or mr. nice guy and the cosa nostra. *The Iterated Prisoners' Dilemma*, page 171–204, 2007.

[18] M.L. Littman. Markov decision processes. In Neil J. Smelser and Paul B. Baltes, editors, *International Encyclopedia of the Social & Behavioral Sciences*, pages 9240–9242. Pergamon, Oxford, 2001.

[19] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '96, page 195–210, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[20] Francisco S. Melo. Convergence of q-learning: a simple proof. *Institute for Systems and Robotics,*.

[21] Christopher J. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[22] Michael Wunder, Michael L. Littman, and Monica Babes. Classes of multiagent q-learning dynamics with epsilon-greedy exploration. In *ICML*, pages 1167–1174, 2010.

[23] Shai Ben-David, Eyal Kushilevitz, and Yishay Mansour. Online learning versus offline learning. *Machine Learning*, 29(1):45–63, 1997.

[24] Shashi Mittal and Kalyanmoy Deb. Optimal strategies of the iterated prisoner's dilemma problem for multiple conflicting objectives. *IEEE Transactions on Evolutionary Computation*, 13(3):554–565, 2009.

[25] Fiona McGillivray and Alastair Smith. Trust and cooperation through agent-specific punishments. *International Organization*, 54(4):809–824, 2000.

Department of Engineering Science

Supplementary Questions for 4th Year Project Students

UNIVERSITY OF OXFORD

Factor	Answer	Things to Consider	Record details here
Has the checklist covered all the problems that may arise from working with the VDU?	☑ Yes ☐ No		
Are you free from experiencing any fatigue, stress, discomfort or other symptoms which you attribute to working with the VDU or work environment?	☑ Yes ☐ No	Any aches, pains or sensory loss (tingling or pins and needles) in your neck, back shoulders or upper limbs. Do you experience restricted joint movement, impaired finger movements, grip or other disability, temporary or permanently	
Do you take adequate breaks when working at the VDU?	☑ Yes ☐ No	Periods of two minutes looking away from the screen taken every 20 minutes and longer periods every 2 hours Natural breaks for taking a drink and moving around the office answering the phone etc.	
How many hours per day do you spend working with this computer?	☐ 1-2 ☐ 3-4 ☑ 5-7 ☐ 8 or more		
How many days per week do you spend working with this computer?	☐ 1-2 ☐ 3-5 ☑ 6-7		
Please describe your typical computer usage pattern	9-12, 1:30-3, 4-6		

Student Declaration and Academic Approval

Student Declaration:

I have completed the DSE Workstation Checklist and the Supplementary Questions for my computer-related risk assessment for 4YP Project Number indicated below:

4YP Project Number: 12571

4YP Student's Name *(please print)*: THOMAS FOSTER-BROWN

4YP Student's Signature: T Foster-Brown

Academic Approval

I confirm my approval of this 4YP DSE Risk Assessment.

Academic Supervisor's Name: *(please print)*

Jakob Foerster

Academic Supervisor's Signature: