Folk Theorem and, in particular, at the thresholds where cooperation is more beneficial than defection in the game of a Prisoners' Dilemma.

Student: Sophie Shapcott

Supervisor: Dr Vincent Knight

Academic Year: 2019/20

School of Mathematics,
Cardiff University

A final year undergraduate project submitted in partial fulfiment of the requirements for MMORS (Masters in Mathematics, Operational Research and Statistics) taught programme.

SUMMARY

ACKNOWLEDGEMENTS

Contents

SUMMARY **i**

ACKNOWLEDGEMENTS **ii**

1 Introduction **1**

1.1 An Introduction to Games 1

1.2 Nash Equilibrium for Normal Form Games 4

1.2.1 Brouwer's Fixed Point Theorem 6

1.2.2 Proof of Nash's Theorem (Theorem 1.2) 6

1.3 Repeated Games 8

1.3.1 Finite Repeated Games 8

1.3.2 Infinite Repeated Games 10

1.4 Folk Theorem 12

1.5 Aims of the Project 14

2 Literature Review **16**

3 Methodology and Experiment Setup **17**

3.1 Data Collection Algorithm 17

3.2 Databases 22

3.2.1 Types of Database 23

3.2.2 Implementation of the Database 25

3.3 Software Implementation 26

3.4 Remote Computing 27

3.5 Calculating Nash Equilibria 30

3.5.1 Support Enumeration 31

3.5.2 Vertex Enumeration 34

3.5.3 Algorithm Execution Timings 36

3.6 Degeneracy 37

4 Analyses **39**

4.1 Initial Analysis 39
4.2 Analysis of the p-Threshold 43
4.2.1 Effects of the Number of Players 48
4.2.2 Effects of Noise 49
4.3 Conclusions and Further Work 50

References **52**

Appendices **58**

A APPENDIX A TITLE **58**

B APPENDIX B TITLE **59**

List of Figures

1.1 Graphs to show the row and column players' payoffs against a mixed strategy. 5
1.2 A plot to show the possible payoffs of the game between two players in which A Prisoner's Dilemma is repeated twice. 9
1.3 A plot highlighting the possible individually rational payoffs for the game of A Prisoner's Dilemma. 12

3.1 An example plot illustrating the p-threshold as indicated by the Folk Theorem 18
3.2 Representation of the algorithm used to collect the data. 18
3.3 DB Browser, a database graphical user interface. 26
3.4 A screenshot of the html report produced when utilising Coverage. 27
3.5 A screenshot of a GitHub pull request which allowed for collaboration between supervisors and author. 28
3.6 A screenshot of the authors connection to Siren via an SSH tunnel. 29
3.7 Representation of how the experiments were run remotely. Note, 'tmux sessions' correspond to emulators of terminals. 29
3.8 Command line code used to retrieve the database from the remote server. 30
3.9 Best response polytope, P, obtained from the payoff matrix given in Equation 3.5.2. 35
3.10 Violinplots of the log timings obtained for the experiment and calculation of Nash equilibria using the three algorithms as implemented in Nashpy. 37

4.1 Example graphs obtained from the experiment. 40
4.2 A plot to show the ratio of stochastic to deterministic strategies randomly chosen throughout the experiment. 42
4.3 A violinplot showing the distribution of the number of equilibria obtained for varying number of players. 44

4.4 Plots to show the p-thresholds for all 1001 sets of tournaments. . . 44
4.5 Plots to show the p-thresholds for all 1001 sets of tournaments. . . 45
4.6 A plot to show the minimum and maximum p-thresholds for those tournaments which had an mean threshold within the range $[0.5, 0.6]$. 46
4.7 Example plots of the tournaments where the mean p-threshold was within the range $[0.5, 0.6]$. 46
4.8 Plots to show the p-thresholds for all tournaments which led to non-degenerate games. 47
4.9 Plots to show the p-thresholds for all tournaments which led to non-degenerate games. 47
4.10 Violinplots of the thresholds for each number of opponents. 48
4.11 Violinplots of the thresholds for each level of noise. 49
4.12 Observation of one 6-player tournament set through the varying levels of additional noise. There was one stochastic player and 13 out of the 1100 tournaments played yielded potential degenerate games. The opponents here were: *Getzler*; *Punisher*; *Forgiver*; *GrudgerAlternator*; and *GraaskampKatzen*. 51

Chapter 1

Introduction

Game Theory is the study of interactive decision making and developing strategies through mathematics [14]. It analyses and gives methods for predicting the choices made by players (those making a decision), whilst also suggesting ways to improve their 'outcome' [30]. Here, the abstract notion of utility is what the players wish to maximise (see Chapter 2 in [30] for a detailed discussion on the topic of utility theory or Section 1.3 [51] for a more introductory explanation). One of the earliest pioneers of game theory is mathematician, John von Neumann who, along with economist Oskar Morgenstern, published *The Theory of Games and Economic Behaviour* in 1944 [30]. This book discusses the theory, developed in 1928 and 1940-41, by von Neumann, regarding "games of strategy" and its applications within the subject of economics [50]. Following this, several advancements have been made in the area, including, most notably, John Nash's papers on the consequently named Nash Equilibria in 1950/51 [32, 31]. Due to the "context-free mathematical toolbox" [30] nature of this subject, it has been applied to many areas, from networks [28, 44] to biology [11, 1]. In this project, the main focus is on a particular class of theorems, within game theory, known as "Folk Theorems" with application to the game of A Prisoner's Dilemma. These will be defined and discussed in the subsequent sections.

1.1 An Introduction to Games

Consider the following scenario:

Two convicts have been accused of an illegal act. Each of these prisoners, separately, have to decide whether to reveal information (defect) or stay silent (cooperate). If they both cooperate then the convicts are given a short sentence whereas if they both defect then a medium sentence awaits. However, in the situation of one cooperation and one defection, the prisoner who cooperated has the consequence of a long term sentence, whilst the other is given a deal [22].

This is one of the standard games in game theory known as A Prisoner's Dilemma. It has four distinct outcomes, for the given two player version, which can be represented as a table (see Table **??**). Each coordinate (a, b) in the table represents the utility values obtained for each player, where a is the utility value obtained by the row player and b is the utility gained by the column player. These utility values are as given in [3] and are used throughout this project. More formally, the game can be represented as the following matrix:

$$\begin{array}{c} \\ coop \\ defect \end{array} \begin{array}{c} \begin{array}{cc} coop & defect \end{array} \\ \begin{pmatrix} (3,3) & (0,5) \\ (5,0) & (1,1) \end{pmatrix} \end{array}$$

which is known as a *normal form* representation of the game. The following definition is adapted from [30].

In general a *normal form* or *strategic form* game is defined by an ordered triple $G = (N, (S_i)_{i \in N}, (u_i)_{i \in N})$, where:

- $N = \{1, 2, \ldots, n\}$ is a finite set of players;
- $S = S_1 \times S_2, \times \cdots \times S_n$ is the set of strategies for all players in which each vector $(S_i)_{i \in N}$ is the set of strategies for player i [1]; and
- $u_i : S \to \mathbb{R}$ is a payoff function which associates each strategy vector, $\mathbf{s} = (s_i)_{i \in N}$, with a utility [2] $u_i(i \in N)$.

Yet another way of representing this game is as a pair of matrices, A, B, defined as follows:

$$A = \begin{pmatrix} 3 & 0 \\ 5 & 1 \end{pmatrix} \text{ and } B = A^T = \begin{pmatrix} 3 & 5 \\ 0 & 1 \end{pmatrix}$$

This way of defining games allow for the use of calculating payoffs for each player (see Section 1.2).

Before continuing the discussion into the key notions of game theory, it needs to be highlighted that there is an important assumption, which is central to most

[1]Since the game of A Prisoner's Dilemma has a finite strategy set for each player $S_i = \{\text{cooperate}, \text{defect}\}(i \in N)$, in this project only finite strategy spaces are considered.

[2]'Utility' is referred to as a player's 'payoff' throughout the remainder of this report.

studies of game theory, entitled *Common Knowledge of Rationality.* This, more formally, is an infinite list of statements which claim:

- The players are rational;
- All players know that the other players are rational;
- All players know that the other players know that they are rational; etc.

Assuming Common Knowledge of Rationality allows for the prediction of rational behaviour through a process entitled *rationalisation* [23] (see section 4.5 in [30] for an alternative explanation of this assumption).

A strategy for player i, s_i, is *strictly dominated* if there exists another strategy for player i, say $\bar{s}_i$, such that for all strategy vectors $s_{-i} \in S_{-i}$ of the other players,

$$u_i(s_i, s_{-i}) < u_i(\bar{s}_i, s_i).$$

In this case we say that s_i is *strictly dominated* by $\bar{s}_i$. Here, $s_{-i} = \{s_1, s_2, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n\}$, i.e. the ith player's strategy has been omitted. The set, S_{-i}, is defined similarly. Looking at the row player's matrix of a Prisoner's Dilemma 1.1 (the first entries in the ordered tuples), it is clear that cooperation is a strictly dominated strategy. Due to the symmetricity of the game, this is also true for the column player. [30]

So far, only the pure strategies, $S_i = \{\text{coop}, \text{defect}\}$, have been discussed, thus the notion of a probability distribution over S_i is now introduced, giving the so-called *mixed strategies* as defined in [30]: Let $G = (N, (S_i)_{i\in N}, (u_i)_{i\in N})$ be a game (with each S_i finite), then a *mixed strategy* for player i is a probability distribution over their strategy set S_i. Define:

$$\Sigma_i := \{\sigma_i : S_i \to [0,1] : \sum_{s_i \in S_i} \sigma_i(s_i) = 1\}$$

to be the set of mixed strategies for player i. Hence, observe that the pure strategies are specific cases of mixed strategies, with $\sigma_i = (1,0)$ for cooperation and $\sigma_i = (0,1)$ for defection, in the example of a Prisoner's Dilemma.

This leads onto the following definition of a *mixed extension* of a game, taken from [30]: Let G be a finite normal form game as above, with $S = S_1 \times S_2 \times \cdots \times S_N$ defining the pure strategy vector set and each pure strategy set, S_i being non-empty and finite. Then the *mixed extension* of G is denoted by

$$\Gamma = (N, (\Sigma_i)_{i\in N}, (U_i)_{i\in N}),$$

and is the game in which, Σ_i is the ith player's strategy set and $U_i : \Sigma \to \mathbb{R}$ is the corresponding payoff function, where each $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_N) \in \Sigma = \Sigma_1 \times \Sigma_2 \times \cdots \times \Sigma_N$ is mapped to the payoff

$$U_i = \mathbb{E}_\sigma(u_i(\sigma)) = \sum_{(s_1,s_2,\dots,s_N)\in S} u_i(s_1, s_2, \dots, s_N)\sigma_1(s_1)\sigma_2(s_2)\cdots\sigma_N(s_n) \qquad (1.1)$$

for all players $i \in N$.

1.2 Nash Equilibrium for Normal Form Games

As mentioned above, mathematician, John Nash, introduced the concept of an equilibrium point and proved the existence of mixed strategy Nash Equilibria in all finite games. These notions are central to the study of game theory [30] and hence, in this section, Nash's concepts will be defined and proved in detail.

Firstly, before the definition of a Nash equilibrium, the idea, as given in [30] of a *best response* is introduced: For a game $G = (N, (S_i)_{i\in N}, (u_i)_{i\in N})$, the strategy, s_i, of the ith player is considered a *best response* to the strategy vector s_{-i} if $u_i(s_i, s_{-i}) = \max_{t_i\in S_i} u_i(t_i, s_{-i})$.

This leads onto the main definition of the section:

Definition 1.2.1. Given a game $G = (N, (S_i)_{i\in N}, (u_i)_{i\in N})$, the vector of strategies $s^* = (s_1^*, s_2^*, \dots, s_n^*)$ is a *Nash equilibrium* if, for all players $i \in N$, s_i^* is a best response to $s_i^* \in N$. [30]

In other words, s^* is a Nash equilibrium if and only if no player has any reason to deviate from their current strategy s_i^*.

Recall that, in Section 1.1, for any player in A Prisoner's Dilemma, defection dominated cooperation. This leads to the following observation:

The strategy pair (Defect, Defect), is the unique Nash equilibrium for A Prisoner's Dilemma, with a payoff value of 1 for each player. [30]

This can be visualised as followed: Assume the row player uses the following mixed strategy, $\sigma_r = (x, 1-x)$, i.e. the probability of cooperating is x and the probability of defecting is $1-x$. Similarly, assume the column player has the strategy, $\sigma_c = (y, 1-y)$. The payoff obtained for the row and column player, respectively, is then:

$$A\sigma_c^T = \begin{pmatrix} 3 & 0 \\ 5 & 1 \end{pmatrix} \begin{pmatrix} y \\ 1-y \end{pmatrix} = \begin{pmatrix} 3y \\ 4y+1 \end{pmatrix},$$

$$\sigma_r B = \begin{pmatrix} x & 1-x \end{pmatrix} \begin{pmatrix} 3 & 5 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 3x & 4x+1 \end{pmatrix}$$

Plotting these gives the following: From Figure 1.1 it is clear that, regardless of

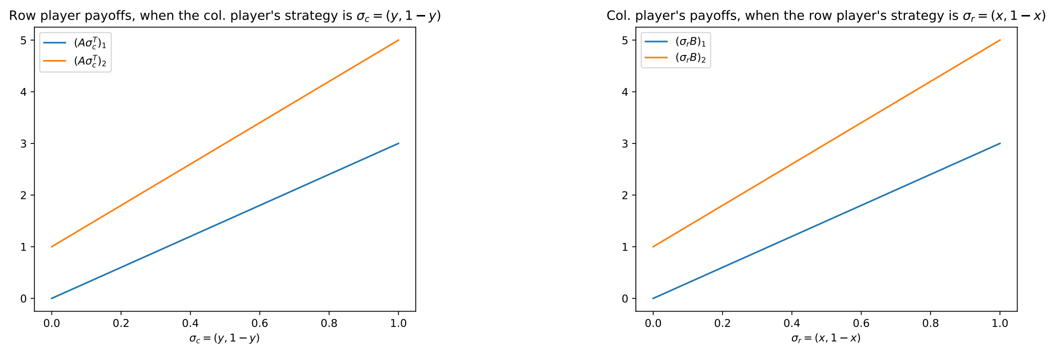


Figure 1.1: Graphs to show the row and column players' payoffs against a mixed strategy.

the strategy played by the opponent, defection is indeed the only rational move for one to play. Thus, both players have no incentive to deviate if and only if both play the strategy $\sigma = (0, 1)$, i.e. defection for every single game of A Prisoner's Dilemma.

On the other hand, consider, for example, a game with no dominated strategies [3]:

$$A = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}, \; B = \begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$$

Are Nash Equilibria guaranteed to exist? This result is given in the next theorem, taken from [31], Nash's second paper on equilibria in games.

Theorem 1.2.1. Every finite game has an equilibrium point.

The proof of Theorem includes the use of a *fixed point theorem* and thus, a short sub-section regarding one such result is given, for completeness, before providing a formal proof of 1.2.

[3]The game highlighted here is another standard used in game theory entitled *Matching Pennies*. Moreover, it is what is defined as a *zero-sum* game. Interested readers are encouraged to read Example 4.21 of [51] for an introduction to the game and Section 4.12 in [30] for an explanation of zero-sum games.

1.2.1 Brouwer's Fixed Point Theorem

Brouwer's Fixed Point Theorem is a result from the theory of topology. Named after the Dutch mathematician, L.E.J. Brouwer, it was proven in 1912 [10]. However, before stating this notion, a few conditions regarding the properties of sets are recalled.

The following three definitions appear as in [53, 5, 52] respectively:

Definition 1.2.2. A set $X \subseteq \mathbb{R}^d$ is called *convex* if it contains all line segments connecting any two points $\mathbf{x}_1, \mathbf{x}_2 \in X$.

Definition 1.2.3. An *open cover* of a set $S \subset X$, a topological space, is a collection of open sets $A_1, A_2, \ldots \subset X$ such that $A_1 \cup A_2 \cup \ldots \supset S$, that is, the union of the open sets contain S.

Definition 1.2.4. A subset $S \subseteq X$, a topological space, is called *compact* if, for each open cover of S, there is a finite sub-cover of S.

The presentation of Brouwer's Fixed Point Theorem is now given as in [30]

Theorem 1.2.2. Let $X \subseteq \mathbb{R}^n$ be a non-empty convex and compact set, then each continuous function $f : X \to X$ has a fixed point.

In other words if X and f satisfy the conditions given above then there exists a point $x \in X$ such that $f(x) = x$.

Since this project is regarding game theory, rather than topology, the proof to the above theorem is omitted. However, the interested reader is referred to [] for an in-depth consideration into the theory of topology.

1.2.2 Proof of Nash's Theorem (Theorem 1.2)

The proof provided here is adapted from the original, by Nash, as presented in [31] (with extra notes adapted from [30]). According to [30], the general idea here is to define a function, which satisfies the conditions required for Theorem 1.2.2, using the payoff functions on the set of mixed strategies. Then by identifying each equilibrium point with a fixed point of the function, the required result is obtained.

Proof. Firstly, a brief restatement of the notation needed is provided for clarity. Let $G = (N, (S_i)_{i\in N}, (u_i)_{i\in N})$ be a finite game with mixed extension $\Gamma = (N, (\Sigma_i)_{i\in N}, (U_i)_{i\in N})$. Here, N denotes the number of players; $S = S_1 \times S_2 \times \ldots \times S_N$ is the set of pure strategies for all players, with $(S_i)_{i\in N}$ the pure strategy set for player i; Σ is defined similarly but relating to mixed strategies; and

$U_i : \Sigma \to \mathbb{R}$ are the payoff functions as given in equation 1.1. Recall that σ_{-i} is used to denote the strategy choice of all players *except* the ith player.

Let $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_N)$ be a tuple of mixed strategies and $U_{i,t}(\sigma)$ be the ith player's payoff if they changed to their tth pure strategy and all other players continue to use their mixed strategy. Now, define function $f : \Sigma \to [0, \infty)$ such that

$$f_{i,t}(\sigma) = \max\left(0, U_{i,t}(\sigma) - U_i(\sigma)\right) \tag{1.2}$$

and also let

$$\sigma_i' = \frac{\sigma_i + \sum_t f_{i,t}(\sigma) s_i^t}{1 + \sum_t f_{i,t}(\sigma)} \tag{1.3}$$

be a modification of each $\sigma_i \in \sigma$, with $\sigma' = (\sigma_1', \sigma_2', \ldots, \sigma_N')$. In words, this modification increases the proportion of the pure strategy t used in σ_i if the payoff gained by the ith player is larger when they replace their mixed strategy by t. Else, it remains the same if doing this decreases their payoff as $f_{i,t}(\sigma) = 0$ in this case. Note, the denominator ensures that the ending vector is still a probability distribution by standardising.

The aim is to apply Theorem 1.2.2 to the mapping $T : \sigma \to \sigma'$ and show that it's fixed points correspond to Nash equilibria. Thus, firstly compactness and convexity of the set σ is shown along with continuity of the function f.

Observe that each σ_i can be represented by a point in a simplex in a real vector space with the vertices given by the pure strategies, s_i^t. Therefore, it follows that the set Σ_i is convex and compact. Using the result, *If $A \subseteq \mathbb{R}^n$ and $B \subseteq \mathbb{R}^m$ are convex compact sets then the set $A \times B$ is a convex compact subset of* $\mathbb{R}^{n+m}$, as highlighted in [30] gives the convexity and compactness of the set Σ, the cross product of all Σ_is.

The continuity of the function f depends upon the continuity of the payoff functions U_i. As given in [30], this is shown by first proving that the U_i are multilinear functions in the variables $(\sigma_i)_{i \in N}$ and then applying the fact that any multilinear function over Σ is a continuous function. [4] The result then follows.

Hence, by Theorem 1.2.2, the mapping T must have at least one fixed point. The proof is concluded by showing that any fixed points of T are Nash equilibria.

Suppose σ is such that $T(\sigma) = \sigma$. Then, the proportion of s_i^t used in the mixed strategy σ_i must not be altered by T. Therefore, in σ_i', the sum $\sum_t f_{i,t}(\sigma)$, in the denominator, must equal zero. Otherwise, the total sum of the denominator

[4] For a detailed consideration of the continuity of the payoff functions, please see [30], pages 148-149.

will be greater than one, decreasing the proportion of s_i^t. This implies that for all pure strategies q, $f_{i,q}(\sigma) = 0$. That is, player i can not improve their payoff by adopting any of the pure strategies. Note, this is true for all i and q by definition of $T(\sigma) = \sigma$ and thus no player is able to improve their payoff. By definition 1.2.1, this is exactly the conditions of a Nash equilibrium.

Now assume σ is a Nash equilibrium. Then, by definition, it must be that $f_{i,q}(\sigma) = 0$ for all pure strategies q for all players, $i \in N$. Note, if $f_{i,q}(\sigma) \neq 0$, then the ith player would benefit from changing their strategy to the pure strategy q, which violates the condition for a Nash equilibrium. From this it follows that $T(\sigma) = \sigma$, that is, σ is a fixed point of T. This concludes the proof.

$\square$

1.3 Repeated Games

The folk theorems studied in this project are a consequence of games which are repeated several times (not just once). Thus, before discussing the main ideas, the theory of both finitely- and infinitely- repeated games is presented.

Firstly, a couple of alterations to the terminology used in previous sections is redefined for conciseness and to be consistent with the literature []. What was known as a 'game' will become known as a *stage game* to highlight the fact that a one-off game is being considered. Also, what was defined previously as a 'strategy' will now be referred to as an *action* to differentiate it from a strategy of a repeated game (see Section 1.3.1).

1.3.1 Finite Repeated Games

According to [25], a *T-stage repeated game*, $T < \infty$ is when the stage game, G, is played T times, over discrete time intervals. Each ith player has a strategy based on previous 'rounds' of the game and the payoff of a repeated game is calculated as the total sum of the stage game payoffs.

Prior to giving the notion of a strategy in a repeated game, the idea of *history*, within the context of repeated games, is provided. The *history*, $H(t)$ of a repeated game is the knowledge of previous actions of all players up until the tth stage game, assumed to be known by player i for all $i \in 1, \dots, N$. Note that, when $t = 0$, $H(0) = \underbrace{(\emptyset, \emptyset, \dots, \emptyset)}_{N \text{times}}$, since no stage games have yet been played.

As given in [20], a *strategy* of a T-stage repeated game is defined to be a mapping

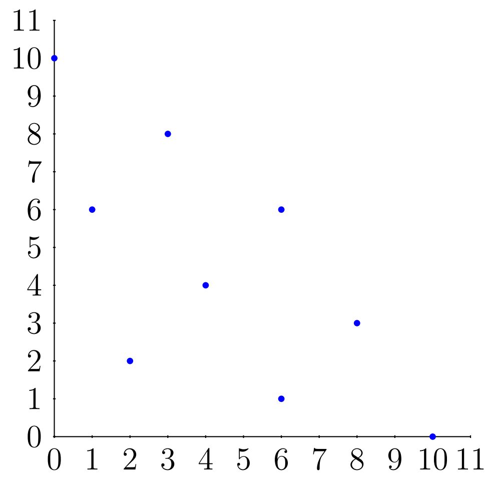


Figure 1.2: A plot to show the possible payoffs of the game between two players in which A Prisoner's Dilemma is repeated twice.

from the complete history so far to an action of the stage game, that is

$$\tau_i : \cup_{t=0}^{T-1} Ht \to a_i.$$

Here, $H(t)$ is the history of play as defined above and a_i is the ith player's action of the stage game.

Consider, for example, the environment in which the stage game of a Prisoner's Dilemma is repeated each time. This is known as the *Iterated Prisoner's Dilemma* (IPD) and has been a popular topic of research for many years see Chapter **??**. Note that the ojective here is to maximise your payoff. The player:

No matter what my opponents play, I will always defect,

commonly known as the 'Defector' has the following strategy mapping:

$$\tau_i : \cup_{t=0}^{T-1} Ht \to a_i,$$

where $a_i = D$ for all time periods $\tau \geq 0$. Other common IPD strategies include:

Cooperator - *No matter what my opponents play, I will always cooperate*;
Random - *I will either cooperate or defect with a probability of 50%*; and
Tit For Tat - *I will start by cooperating but then will duplicate the most recent decision of my opponent.*

Figure 1.2 shows the possible payoffs obtained in a 2-stage repeated IPD with two players:

Now, what about Theorem 1.2? Are there any Nash equilibria in repeated games?

In fact, it can be proven that there exist many equilibria in repeated games [15]. The next result, adapted from [20] guarantees at least one Nash equilibria.

Theorem 1.3.1. Consider a T-stage repeated game with $G = (N, (S_i)_{i\in N}, (u_i)_{i\in N})$ as the stage game, $0 < T < \infty$. Define by $\boldsymbol{\sigma}^* = (\sigma_1^*, \sigma_2^*, \ldots, \sigma_N^*)$, a stage Nash equilibrium of G. Then the sequence in which $\boldsymbol{\sigma}^*$ is continuously played is a Nash equilibrium of the T-stage repeated game.

Proof. Since $\boldsymbol{\sigma}^*$ is a stage Nash equilibrium, it is, in particular, a Nash equilibrium of the Tth stage game. Thus, no player has any reason to deviate here. But then $\boldsymbol{\sigma}^*$ was also played at the $(T-1)$th stage also, meaning there is still no reason to deviate. Therefore, continuing via backwards induction gives the required result. $\square$

Hence, for the T-stage IPD, every player executing the Defector strategy yields a Nash equilibrium. However, do there exist equilibria for which selecting the action 'cooperate' is more beneficial?

In preparation to answer this, the next Subsection 1.3.2 discusses the case when $T \to \infty$ and results linked to *infinitely repeated games.*

1.3.2 Infinite Repeated Games

The Folk Theorem discussed in Section 1.4 considers a stronger notion of equilibria known as *subgame perfect equilibria.* In order to fully understand this notion, a new representation of games is introduced.

Extensive Form Games

According to [30], an *extensive form game* is given by the ordered vector $\Gamma = (N, V, E, x_0, (V_i)_{i\in N}, O, u)$ where N is a finite set of players; (V, E, x_0) is a *game tree* [5]; $(V_i)_{i\in N}$ is a partition of the set $V \setminus L$, where L is the set of all leaves, or terminal points, of the game tree; O is the set of outcomes for the game; and u is a function which maps each leaf in L to an outcome in O.

This leads on to the following definition, adapted from [51]:
A player's *information set* is a subset of the nodes in a game tree where:

- Only the player concerned is deciding;

[5] The triple (V, E, x_0) is defined as a *tree* if the set of vertices, V, and the set of edges, E, create a *directed graph* (that is, each element in E is an ordered tuple) and x_0 represents the root, or starting node, of the graph.

- This player is not aware of which node has been reached except that it is definitely one of the elements found in this set.

In Figure **??**, the extensive form representation of the game A Prisoner's Dilemma is provided. Here, only two players are considered and any information sets are represented by a dashed line. Note, any normal form game can be represented as an extensive form game.

According to [51], a *subgame* is a sub-graph of the game tree such that:

- The sub-graph begins at a decision node, say x_i;
- This node, x_i, is the only element contained in its information set;
- The sub-graph contains all of the decision nodes which follow x_i.

This leads to the following definition of *subgame perfect equilibria*, adapted from [51].

Definition 1.3.1. A *subgame perfect equilibrium* is a Nash equilibrium which satisfies the condition that the strategies played define a Nash equilibrium in every subgame.

Hence the strategy defined in Theorem 1.3.1 is a subgame perfect equilibrium. A few final definitions are now highlighted before introducing the Folk Theorem.

Final Definitions Needed

Now, in order to be able to discuss the payoffs of strategies in infinite games; the notion of a *discounted payoff* is introduced. This is defined in [21] as:

$$V_i(\sigma) = \sum_{t=1}^{\infty} \delta^{t-1} U_i(\sigma),$$

where the discount factor, δ, can be thought of as the probability that the game continues. That is, the probability that another stage game will be played. Using this, the *average payoffs*, per stage game, can be defined by:

$$\frac{1}{\bar{T}} V_i(\sigma) = (1-\delta) V_i(\sigma),$$

where $\bar{T} = \frac{1}{1-\delta}$ is the average length of a game [21].

Finally, Figure 1.3 shows those payoffs which are individually rational for a two player version of A Prisoner's Dilemma. In general, an *individually rational payoff* is an average payoff which exceeds the payoffs obtained in the stage Nash equilibria for all players [21]. Note, often the Nash equilibrium payoffs are not the optimal payoff which players could achieve.

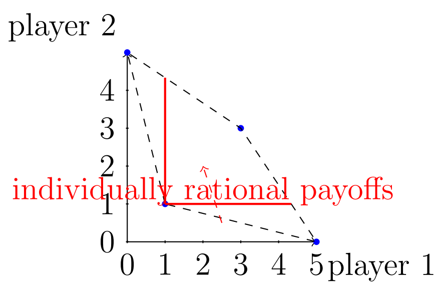


Figure 1.3: A plot highlighting the possible individually rational payoffs for the game of A Prisoner's Dilemma.

In the next section (Section 1.4), the main theorem of this project will be stated and proved.

1.4 Folk Theorem

According to [51], the class of theorems known as Folk Theorems are so-called because the result was well-known before a formal proof was provided. In general, these theorems state that players can achieve a better payoff than the Nash equilibrium (if the Nash equilibrium payoff is not optimal) when the stage game is repeated many times and the probability of the game continuing is large enough.

It is believed that Friedman, 1971 ([15]) was one of the first to publish a formal proof to the widely accepted Folk Theorem []. Thus, the presentation of the statement and proof given here is adapted from [15] as well as from [21].

Before stating the theorem, the list of assumptions which Friedman [15] requires the infinite repeated game to satisfy is given.

1. The mixed action sets, Σ_i are compact and convex for all $i \in N$.
2. The payoff functions, $U_i : \Sigma \to \mathbb{R}$, are continuous and bounded for all $i \in N$.
3. The $U_i(\sigma)$s are quasi-concave [6] functions of $sigma_i$ for all $i \in N$.
4. If $U_i' \leq U_i''$, for all $i \in N$ and $U_i', U_i'' \in H$, then, for all $U_i' \leq U \leq U_i''$, $U \in H$. Here, H is defined to be the set of feasible payoffs, $\{U(\sigma) : \sigma \in \Sigma\}$, where $U(\sigma) = (U_1(\sigma), U_2(\sigma), \ldots, U_N(\sigma))$.

[6]According to [46], a real-valued function f, defined on a convex subset $C \subset \mathbb{R}^n$, is *quasi-concave* if for all $\alpha \in \mathbb{R}$, the set $\{x \in C : f(x) \geq a\}$ is convex.

5. H^* is concave, where $H^* \subset H$ denotes the set of all Pareto optimal payoffs. [7].
6. All stage games are identical in the infinitely repeated game.
7. The discount parameter, δ, is equal in all time periods.
8. The stage game has a unique Nash equilibrium.
9. The Nash equilibrium is not Pareto optimal.

Note, Friedman [15] later goes on to prove that assumptions six to nine can be removed with only a small effect on the result. However, since the only game being studied in this project is the IPD (which satisfies all the above assumptions), this generalisation will be omitted. Thus, only the proof of the original theorem will be provided.

Theorem 1.4.1. If the above assumptions are all satisfied for the given infinite repeated game, then for any individually rational payoff V_i (greater than the payoff yielded by the stage Nash equilibrium σ^{ne}), there exists a discount parameter δ^* such that for all δ_i, $0 < \delta^* < \delta_i < 1$ there is a subgame perfect Nash equilibria with payoffs equal to V_i.

Proof. Consider the set of all actions which yield greater payoffs than the Nash equilibrium, denoted by:

$$B = \{\sigma : \sigma \in \Sigma, U_i(\sigma) > U_i(\sigma^{ne}), i = 1, 2, \dots, N\}$$

and define the following trigger strategy:

$$\sigma_{i1} = \sigma_i', \sigma_{it} = \begin{cases} \sigma_i', & \text{if } \sigma_{j\tau} = \sigma_j' \; j \neq i, \tau = 1, 2, \dots, t-1, t = 2, 3, \dots \\ \sigma_i^{ne}, & \text{otherwise,} \end{cases} \tag{1.4}$$

where $\sigma_i' \in B$. In words, the ith player will choose σ_i' unless any other player does not play σ_j', in which case they continue by playing their Nash equilibrium action, σ_i^{ne}. Now, by definition, the strategy in 1.4 is an equilibrium of the repeated game if

$$\sum_{\tau=0}^{\infty} \delta_i^{\tau} U_i(\sigma_i') > U_i(\sigma_{-i}', t_i) + \sum_{\tau=1}^{\infty} \delta_i^{\tau} U_i(\sigma^{ne}), \quad i = 1, 2, \dots, N,$$

which can be rearranged to

$$\frac{\delta_i}{1-\delta_i}[U_i(\sigma') - U_i(\sigma^{ne})] > U_i(\sigma_{-i}', t_i) - U_i(\sigma\prime), \quad i = 1, 2, \dots, N,$$

[7] Friedman defines a *Pareto optimal payoff* as a point in the payoff space $U_(\sigma^*)$ which satisfies the conditions: $\sigma^* \in \Sigma$ and $U_i(\sigma^*) > U_i(\sigma)$ for all $i \in N$

where $U_i(\sigma'_{-i}, t_i) = \max_{\sigma_i \in \Sigma_i} U_i(\sigma'_{-i}, \sigma_i)$, $t_i \in \Sigma_i$.

To check if this strategy is indeed a best response to all others players, who are executing the same strategy in 1.4, for the ith player consider their alternatives. They have two options: either the ith player executes the strategy in question, or they play the strategy in which $\sigma_{i1} = t_i$ and then $\sigma_{i\tau} = \sigma^{ne}$ will be the best response as every other player will convert to $\sigma_{j\tau} = \sigma^{ne}$, for all $\tau > 1$. Note that any other strategy is weakly dominated by one of these two, since playing t_i in any other stage $\tau \neq 1$ will yield less gains as these stages have increased discounting.

Now, if the discounted loss from playing the Nash equilibria,

$$\frac{\delta_i}{1-\delta_i}[U_i(\sigma') - U_i(\sigma^{ne})], \tag{1.5}$$

is greater than the gain achieved by playing t_i against σ'_{-i}, then the rational strategy choice for player i, assuming all other players are executing 1.4, is to play 1.4.

Observe, as the discount parameter, δ, tends to one from below, the discounted loss in 1.4 tends to infinity. However, the gain obtained from playing t_i, that is, $U_i(\sigma'_{-i}, t_i) - U_i(\sigma\prime)$ is finite. Thus, for all $\sigma'_i \in B$ there exists a $\delta^* \in (0, 1)$ such that for all $\delta_i > \delta^*$, the strategy 1.4 is optimal against the same strategy for all players $j \neq i$. Therefore, if the conditions are true for all players $i = 1, 2, \ldots, N$, the strategy $(\bar{\sigma}_1, \bar{\sigma}_2, \ldots, \bar{\sigma}_N)$, where $\bar{\sigma}_i$ denotes 1.4, yields a Nash equilibrium.

Finally, by construction, the strategy 1.4 is indeed a subgame perfect equilibrium.

□

1.5 Aims of the Project

This project stemmed from an initial idea used in Game Theory Coursework written by the author. In this coursework, regarding Nash equilibria and repeated games, the two graphs, as presented in Figure **??**, were obtained.

These two graphs were obtained using a similar method to this project and so please see Chapter **??** for the details. The plots show the least probability of defection obtained in the Nash equilibria of the game. The game matrices were defined by repeating an IPD tournament 100 times for the Defector strategy, and three other opponents, with 100 distinct probabilities of the game ending.

From Figure **??**, it can be seen that there appears to be a 'clear probability' of the game ending at which the least probability of defection jumps from zero to one.

(Throughout the rest of this project, this ‘clear probability’, p, will be referred to as the *p-threshold* of a game.) However, what was most intriguing was that the two different games obtained had a different p-threshold (approximately 0.25 for Figure **??** but for Figure **??** the threshold appears at around 0.5). This inspired the idea to investigate whether there are any specific characteristics of an IPD tournament that affect the value of this threshold. That is, does the number or the stochasticity of players, for example, cause the probability of a game ending for which the least probability of defection jumps to increase or decrease?

Therefore, the aims of this project are as follows:

1. To look thoroughly into the recent (and past) literature of research already produced in the areas of Folk Theorems and the IPD;
2. To execute a large experiment involving many tournaments of the IPD with different types / numbers of players to obtain graphs similar to those in Figure **??**;
3. To perform analyses on where the p-thresholds seem to lie and whether it is affected by the different environments of the tournaments (that is, differing number of players, stochasticity within the tournament itself and the players, etc.); and
4. To explore other ‘Folk-like’ Theorems and perform similar experiments and analyses.

Chapter 2

Literature Review

Chapter 3

Methodology and Experiment Setup

In this chapter the methods used to collected the data is provided along with justifications. The study required the execution of several IPD tournaments and thus appropriate software needed to be implemented and tested for accuracy.

3.1 Data Collection Algorithm

This section describes the overall algorithm used to collect the data and the attributes collected. Firstly, the aim of this exercise was to illustrate the Folk Theorem and analyse the p-values via a large empirical data collection. It was expected that plots similar to Figure 3.1 would appear. This clearly shows that there eventually does exist a probability of game ending where defection is not a rational decision.

Therefore, in order to observe whether any 'environmental' settings of the tournament does affect the p-threshold, a large amount of data is needed in order for any observations to be statistically significant. Figure 3.2 shows a pictorial representation of the collection method. Each step visible will be explained in detail throughout this chapter with references to the appropriate sections.

From Figure 3.2, it can be seen that the first step was to set up an empty database ready to input each tournament result into. The specific details of implementation into the algorithm is discuss in Section 3.2. However the choices made on the attributes to collect will be described here.

Table 3.1, shows each attribute chosen to be observed in this study. The attributes experiment_number and tournament_player_set were used to provide a

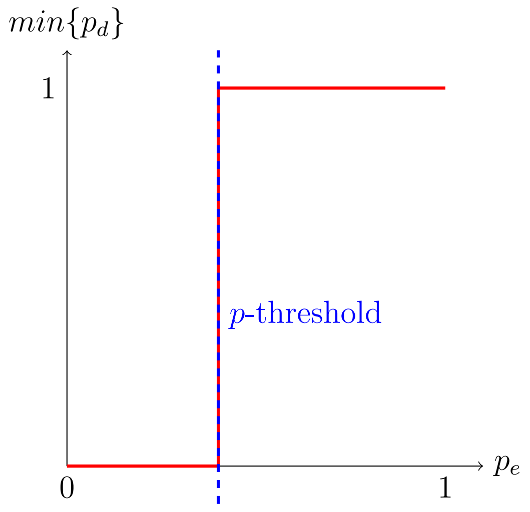


Figure 3.1: An example plot illustrating the p-threshold as indicated by the Folk Theorem

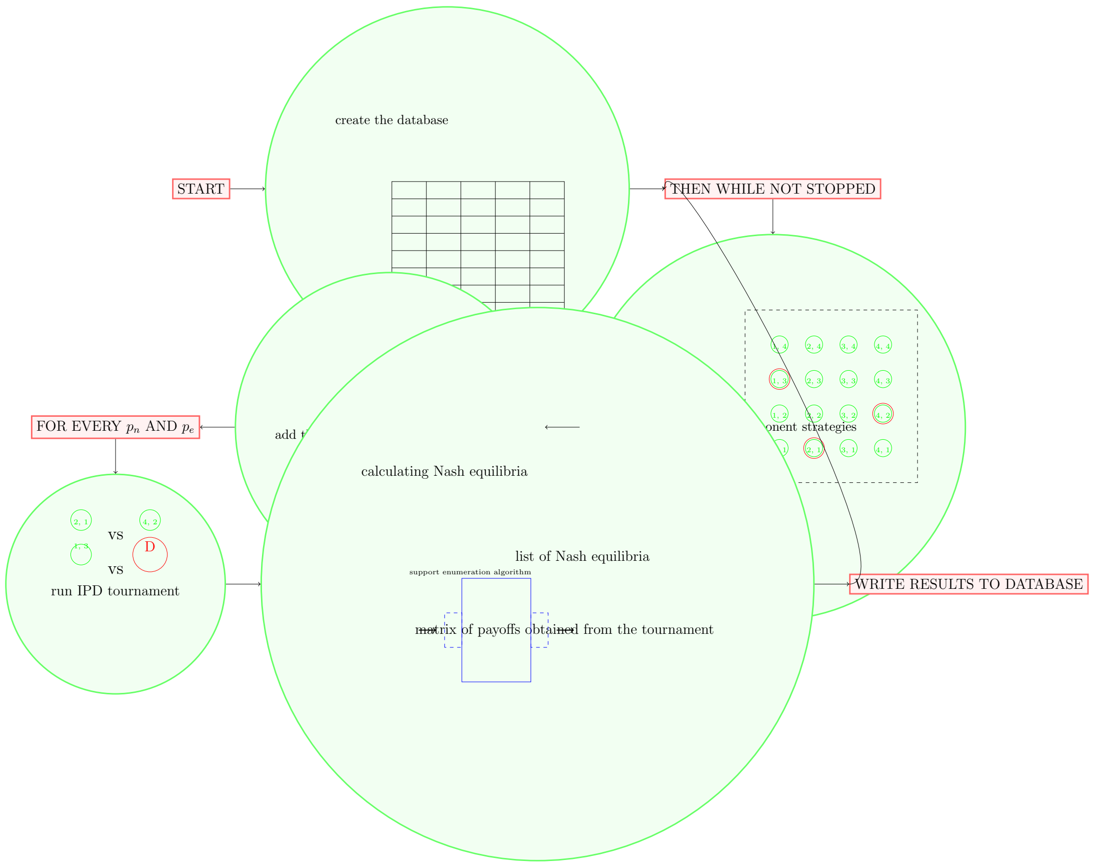


Figure 3.2: Representation of the algorithm used to collect the data.

Attribute	**Description**
experiment_number	A unique seed for each tournament run.
number_of_players	The number of strategies including the Defector.
tournament_player_set	A unique number for a particular set of strategies.
player_strategy_name	The strategy name as given in [13].
is_long_run_time	Characteristic from [13].
is_stochastic	Characteristic from [13].
memory_depth_of_strategy	Characteristic from [13].
prob_of_game_ending	The value of the game ending probability parameter implemented in the tournament.
payoff_matrix	A string of the matrix of mean payoff values from the tournament.
num_of_repetitions	A value indicating how many iterations of the tournament is required.
num_of_equilibria	The number of equilibria obtained as output from the algorithm chosen.
nash_equilibria	A string containing the list of yielded equilibria.
least_prob_of_defection	The lowest probability of playing the Defector obtained from the nash equilibria.
greatest_prob_of_defection	The highest probability of playing the Defector obtained from the nash equilibria.
noise	Level of additional noise added to the tournament.
warning_message	This column contained a string if the algorithm detected potential degeneracy.

Table 3.1: Table of attributes collected.

unique identification for each tournament and set of strategies for ease of separation during analysis. The three characteristics of the strategies were collected with the aim of analysing the different 'types' of strategies' effect on the threshold. The attributes least_prob_of_defection, warning_message, number_of_players and noise were the most important attributes to this study. These were the main effects with regards to analysis of the threshold and key descriptors of the game. The rest of the attributes were retained for self-evaluation of degeneracy and in order to replicate the tournaments for validity and further research.

Following this came the selection of opponents. The number of opponents selected ranged from one to eight and were randomly selected from the appropriate collection of strategies in Axelrod [13]. Out of the 235 strategies currently implemented in Axelrod, only 216 were valid for this experiment. Since this was considering the Folk Theorem for the IPD, research strategies which 'cheated' (that is, those which return False when entered into the obey_axelrod() function) were omitted. The Defector strategy was also removed as it would later be added to each set of players. Moreover, due to time constraints, the 18 strategies which are classified as having a long execution time were omitted.

The actual IPD tournament was run using the original Axelrod tournament setup [1] as implemented in the library Axelrod [13]. This is a round-robin type tournament where each strategy plays every other strategy once [3]. Each round robin was repeated 500 times (why?) to obtain 'smoother' estimates of the mean payoff values. Moreover, each strategy set was run in a tournament for 100 game ending probabilities within the range $[0.001, 0.999]$. Note, 0 was not included as this would imply the tournament would never end and 1 was omitted as the tournament would immediately end after the first turn. However, it is possible for the probabilities within the range $[0.999, 1]$ to be included in this range and this is recommended for further research. These tournaments were also repeated for additional levels of noise in the set, $\{0.1, 0.2, \ldots, 1\}$. The main output of interest here is the payoff matrix, which is then inputted as the game matrix in Nashpy. According to [13], each entry $a_{i,j}$ gives the mean payoff of player i against player j. Consider the following example:

$$\begin{pmatrix} 2.990 & 2.996 & 0.487 \\ 2.996 & 3.000 & 0.989 \\ 3.053 & 1.042 & 1.000 \end{pmatrix} \tag{3.1}$$

The matrix in 3.1 is yielded from a three player tournament with a game ending probability of 0.001 and no additional level of noise. The strategies in this case were Colbert; Tideman and Chieruzzi; and Defector. In this case, the entry $a_{1,2} = 2.996$ is interpreted as the mean payoff between Colbert; and Tideman and Chieruzzi.

For the calculation of the Nash equilibria, the *support enumeration algorithm* was used. This algorithm as well as the justification of its use is provided in Section 3.5.

Finally, the values obtained were written to the database file, one record is inserted for each strategy in order to retain all characteristics. Here, any entries that were not integers or floats had to be converted to strings in order to be stored. For example, the payoff matrix given in 3.1 is a list and hence could not be inserted into the database in its current format.

In summary, pseudo code for the overall algorithm is provided in Algorithm 1.

Algorithm 1: Folk Theorem Exploration

```
input : maximum number of opponents, number of strategy sets
        for each number of opponents, noise levels, game ending
        probabilities, number of repetitions of the tournament, the
        path to the database file, and whether or not support
        enumeration should be used to calculate the Nash
        equilibria.
output: a database containing the results as detailed above.
1  while True do
2      for each number of opponents do
3          for each repetition with the same number of strategies do
4              Randomly select a set of opponents and add in the
                 Defector
5              for each noise level do
6                  for each game ending probability do
7                      Run the IPD tournament
8                      Obtain the Nash equilibria and the corresponding
                         probabilities of defection using the algorithm
                         indicated
9                      for each player in the current set do
10                         Write the required information to a record in
                             the database file.
11                     end
12                 end
13             end
14         end
15         Repeat
16     end
17 end
```

3.2 Databases

There are many different options for types of file available for storage of data. For example, csv, json, tex, txt, db etc. These are generally split into two types: plain text and binary files. In this section, the justification for using an SQLite database is provided along with how this was implemented. But firstly, the advantages and drawbacks of plain text and binary files are discussed.

Plain text or flat file is a format which stores data entries in a single table with column separated by delimiters such as commas or tabs **??**. The contents are understandable by humans. Examples of these include csv and txt. The advantages of plain text formats include: a simple structure, less disk space used and portable **??**. However, there are also drawbacks. Flat files are not scalable and are protected by less security [49]. Only one user can edit the file at any one time and when wanting to search through the file, it has to be fully loaded into the system [9]. Moreover, the columns must all contain the same data type [47].

On the other hand, binary file is a format in which sequences of 0s and 1s are unconstrained as compared to plain text files (where the binary codes have to represent character sets) [43]. Examples of these include databases, executables and media files [43]. The benefits of using binary file formats include: uses less storage space used, is less effort computationally and more secure (it is not understood by humans) [4] Though this this also a disadvantage as it makes a file harder to edit and understand [4].

3.2.1 Types of Database

Using the reasons provided above, it was decided that a binary file, in particular, a database file would be the most appropriate format to use. Primarily, this was due to the fact that databases are generally more robust and support out of memory operations. Indeed, a csv file could have been used however every entry would have had to be a string and if this contained commas, it would break the column structure. Research into the ideal file format for database collection resulted in the identification of two main types of databases: relational databases and noSQL (Not Only SQL) databases.

Relational databases are a file format which stores data according to the relational model as described in [12]. Examples of relational databases management systems include: SQLite, MySQL, PostgreSQL, Oracle and Db2. Briefly, this model involves the structuring of the data in a table, where each row is a record of an instance with a unique ID, or key, and each column is an attribute of the instances. This provided an ideal way to identify relationships between the varying records. This model was developed in the 1970s and was motivated by the reason that originally structures of databases varied with the application used. There are many advantages to a relational database format, including: data consistency - the data is immediately available across several instances of the database, no ‘catch-up’ time needed; commitment - strict rules regarding permanent changes within the database; allows for stored procedures - blocks of code which can be repeatedly accessed; SQL (Structured Query Language) has been developed for ease of query performance using mathematics; and data locking / concurrency - allows for many users to query the database simultaneously without conflicts. A good paper on the model and benefits of relational databases is [35]. On the other hand, one major drawback to this format is its performance in handling extremely large data sets; which have become increasingly popular. Once the data goes beyond a certain size a relational database has to be distributed across many servers, also, this model does not support high scalability - that is, relational models are unable to support large volumes of workloads. Moreover, the strict structure required for this format means that if data cannot be easily transformed into this structure, the complexity of the model increases. The article [18], provides a more detailed description of these disadvantages.

Alternatively, noSQL databases were created with the motivation to be more efficient with large volumes of data. There are several different types of noSQL databases. Key-value store databases, such as RIAK, store the data as a simplistic key-value pair and are similar to hash tables. Column-oriented databases, for example Cassandra, are hybrid row-column databases and document-oriented databases, such as MongoDB, store 'records' in the form of documents with a unique key for representation. Finally, graph databases and object-oriented databases, such as Neo4j and Db4o, respectively, store the data as graphs (in the former case) and as objects, similar to those in Object Oriented Programming (in the latter case). Advantages of noSQL databases include: more flexibility with a wide range of models available; supports scalability; and are more efficient. However, these models are relatively new in comparison with relational models and there is no standard querying language. Also, some of these databases are not as effective as relational databases regarding consistency and commitment, and maintenance is challenging. For a more detailed approach to noSQL, see [33].

Using this information it was decided that a relational database would be the most appropriate. Although it is intended that a large amount of data will be collected, due to time limitations, it is thought that the database is unlikely to become too big for the system. Moreover, the structure and consistency of a relational model is ideal for comparison of the IPD experiments. The Database Management System decided upon was SQLite. This was due to the fact that there exist Python libraries, for example sqlite3 and sqlachemy, for accessing the database and its contents. Also, it is portable, with the entire database stored in a single file meaning it could be transferred easily from the varying computers being used. Other benefits include: its ease of use, with no configuration files

```
    database_management_sys = sa.create_engine(
    "sqlite:///" + database_filepath + "main.db"
)
connect_dbms_to_db = database_management_sys.connect()

read_into_sql = """
    INSERT into folk_theorem_experiment
        (experiment_number, number_of_players, tournament_player_set,
        player_strategy_name, is_long_run_time, is_stochastic,
        memory_depth_of_strategy, prob_of_game_ending, payoff_matrix,
        num_of_repetitions, num_of_equilibria, nash_equilibria,
        least_prob_of_defection, greatest_prob_of_defection, noise,
        warning_message)
    VALUES
        (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
"""

record = (
    experiment_number, number_of_players, tournament_player_set,
    str(player_strategy_name), is_long_run_time, is_stochastic,
    memory_depth_of_strategy, prob_of_game_ending, payoff_matrix_as_string,
    num_of_repetitions, num_of_equilibria, nash_equilibria_as_string,
    least_prob_of_defection, greatest_prob_of_defection, noise,
    warning_message,
)

connect_dbms_to_db.execute(read_into_sql, record)
```

Listing 1: Python code used to record the results from one strategy into the database.

and the fact it is self-contained [36].

3.2.2 Implementation of the Database

The ability to import results straight from Python was through the library sqlalchemy [7]. This allowed for the creation of the database through to accessing the results via Python functions and expressions. For example, consider the code in Listing 1 which was used to insert a record of results for one strategy into the database:

Also, to ensure records were being inserted into the database correctly, the graphical user interface, DB Browser [38] was utilised. It is implemented with a “familiar spreadsheet-like interface” for ease of use and is compatible with SQLite Databases making it ideal for this use [38]. See Figure 3.3, for a screenshot of the interface.

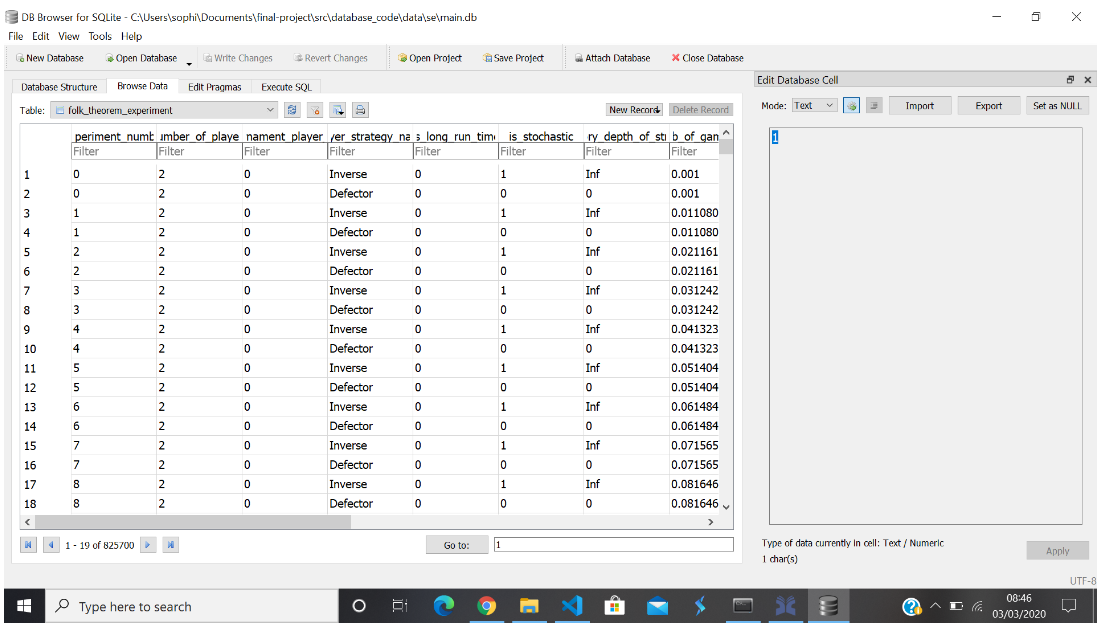

DB Browser for SQLite - C:\Users\sophi\Documents\final-project\src\database_code\data\se\main.db

Table: folk_theorem_experiment

	periment_numb	umber_of_playe	nament_player	yer_strategy_na	s_long_run_tim	is_stochastic	ry_depth_of_st	b_of_gan
1	0	2	0	Inverse	0	1	Inf	0.001
2	0	2	0	Defector	0	0	0	0.001
3	1	2	0	Inverse	0	1	Inf	0.011080
4	1	2	0	Defector	0	0	0	0.011080
5	2	2	0	Inverse	0	1	Inf	0.021161
6	2	2	0	Defector	0	0	0	0.021161
7	3	2	0	Inverse	0	1	Inf	0.031242
8	3	2	0	Defector	0	0	0	0.031242
9	4	2	0	Inverse	0	1	Inf	0.041323
10	4	2	0	Defector	0	0	0	0.041323
11	5	2	0	Inverse	0	1	Inf	0.051404
12	5	2	0	Defector	0	0	0	0.051404
13	6	2	0	Inverse	0	1	Inf	0.061484
14	6	2	0	Defector	0	0	0	0.061484
15	7	2	0	Inverse	0	1	Inf	0.071565
16	7	2	0	Defector	0	0	0	0.071565
17	8	2	0	Inverse	0	1	Inf	0.081646
18	8	2	0	Defector	0	0	0	0.081646

1 - 19 of 825700

Go to: 1

Edit Database Cell

Mode: Text

Type of data currently in cell: Text / Numeric

1 char(s)

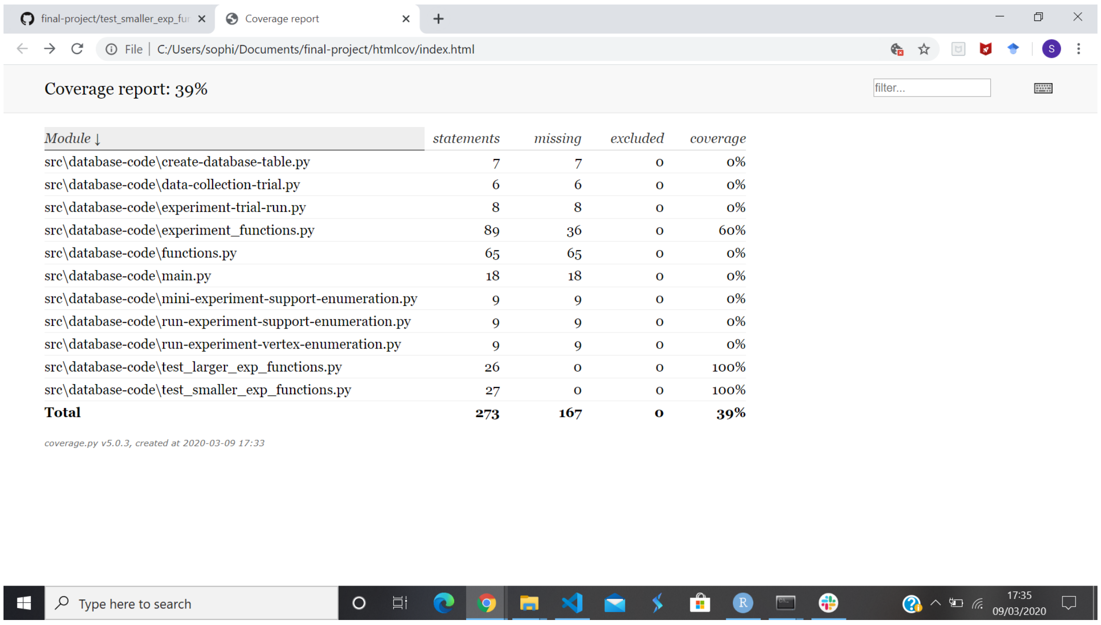
Figure 3.3: DB Browser, a database graphical user interface.

3.3 Software Implementation

There were many potential choices of language for the execution of this experiment however Python had the added advantage of pre-existing libraries, Axelrod [13] and Nashpy [48], which enabled running the IPD as well as calculation of the Nash equilibria. Thus, Python has been used for the collection and analysis of data.

Throughout the implementation of this experiment into Python, good software development principles have been followed [19, 41, 54]. Self-documenting code was ensured through the careful naming of variables as well as the use of docstrings to fully describe function parameters and usage. Python libraries Black [27] and Blackbook [20] assisted in improving the readability of the code, through formatting according to the guidelines of PEP-8 [40]. This gave consistency to all code files created during the study. Modularity through the creation of several smaller functions, focusing on one task, not only assisted with debugging but also enables future usability of the code in newer developments.

Testing is another key part of software development to ensure the durability of the code. Thus, unit tests have been implemented, using Pytest [26], to assist identification of bugs in the functions created for data collection. However, further work in this area is needed to provide a good coverage of the program. Indeed, from executing the Python library Coverage [6], a coverage of 60% was identified, yielding the report shown in Figure 3.4. This could be improved through the creation of integration tests between the database and the experiment results


final-project/test_smaller_exp_fu
Coverage report
File
C:/Users/sophi/Documents/final-project/htmlcov/index.html
filter...
Type here to search
17:35
09/03/2020

Coverage report: 39%

Module ↓	*statements*	*missing*	*excluded*	*coverage*
src\database-code\create-database-table.py	7	7	0	0%
src\database-code\data-collection-trial.py	6	6	0	0%
src\database-code\experiment-trial-run.py	8	8	0	0%
src\database-code\experiment_functions.py	89	36	0	60%
src\database-code\functions.py	65	65	0	0%
src\database-code\main.py	18	18	0	0%
src\database-code\mini-experiment-support-enumeration.py	9	9	0	0%
src\database-code\run-experiment-support-enumeration.py	9	9	0	0%
src\database-code\run-experiment-vertex-enumeration.py	9	9	0	0%
src\database-code\test_larger_exp_functions.py	26	0	0	100%
src\database-code\test_smaller_exp_functions.py	27	0	0	100%
Total	**273**	**167**	**0**	**39%**

coverage.py v5.0.3, created at 2020-03-09 17:33

Figure 3.4: A screenshot of the html report produced when utilising Coverage.

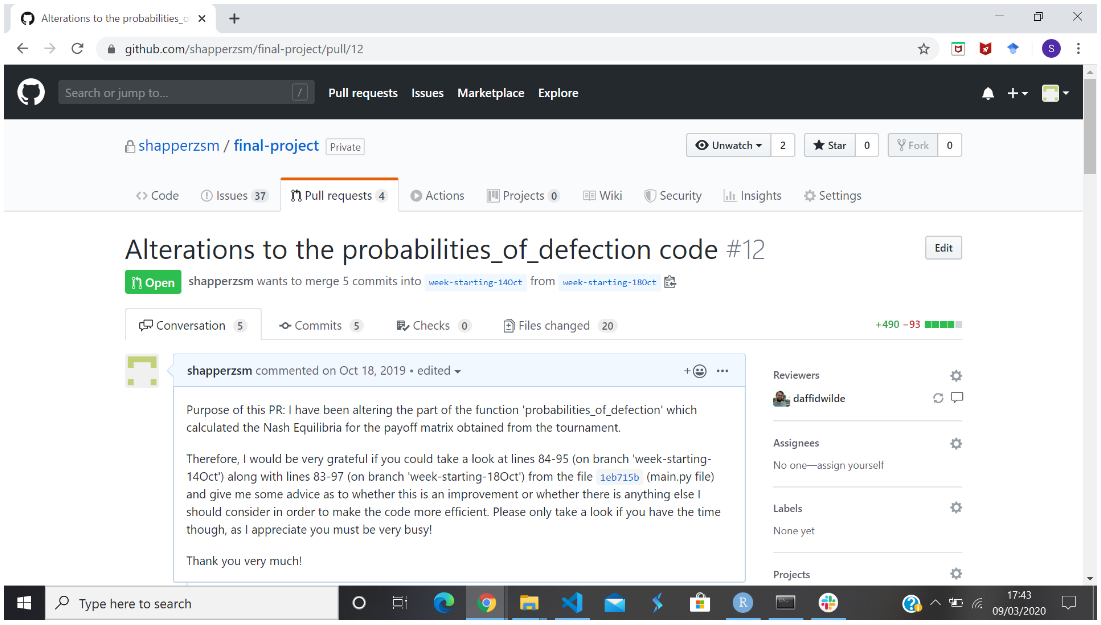
or the implementation of functional tests to confirm the end result of the full algorithm.

The use of version control is key for keeping track of past and present changes to the system. In this study the software Git (https://git-scm.com/) was in use. This allowed for the adaption of code from several different function attempts and, through GitHub (https://github.com/), enabled collaboration between the author and supervisors as seen in Figure 3.5. Moreover, the use of GitHub ensured that a back-up copy of all project files were available, should the current system being used happen to fail and acted as an intermediate step between the author's laptop and the remote server, used for running the experiment (see Section 3.4).

3.4 Remote Computing

This section describes the execution of the experiments via a remote computer and the reasons for doing so.

Firstly, due to the volume of data that was planned to be collected, it was decided that a running the data remotely would be ideal, in order to allow the code to run uninterrupted for several weeks. Thus, Cardiff University School of Mathematics' computer Siren, a headless server with a large storage, was used. However, when a trial was executed, it was decided that the run time was 'quick enough' and hence parallel processing would not be necessary. Yet, for further runs of the



Alterations to the probabilities_of_defection code #12

shapperzsm wants to merge 5 commits into week-starting-14Oct from week-starting-18Oct

shapperzsm commented on Oct 18, 2019 • edited

Purpose of this PR: I have been altering the part of the function 'probabilities_of_defection' which calculated the Nash Equilibria for the payoff matrix obtained from the tournament.

Therefore, I would be very grateful if you could take a look at lines 84-95 (on branch 'week-starting-14Oct') along with lines 83-97 (on branch 'week-starting-18Oct') from the file 1eb715b (main.py file) and give me some advice as to whether this is an improvement or whether there is anything else I should consider in order to make the code more efficient. Please only take a look if you have the time though, as I appreciate you must be very busy!

Thank you very much!

Figure 3.5: A screenshot of a GitHub pull request which allowed for collaboration between supervisors and author.

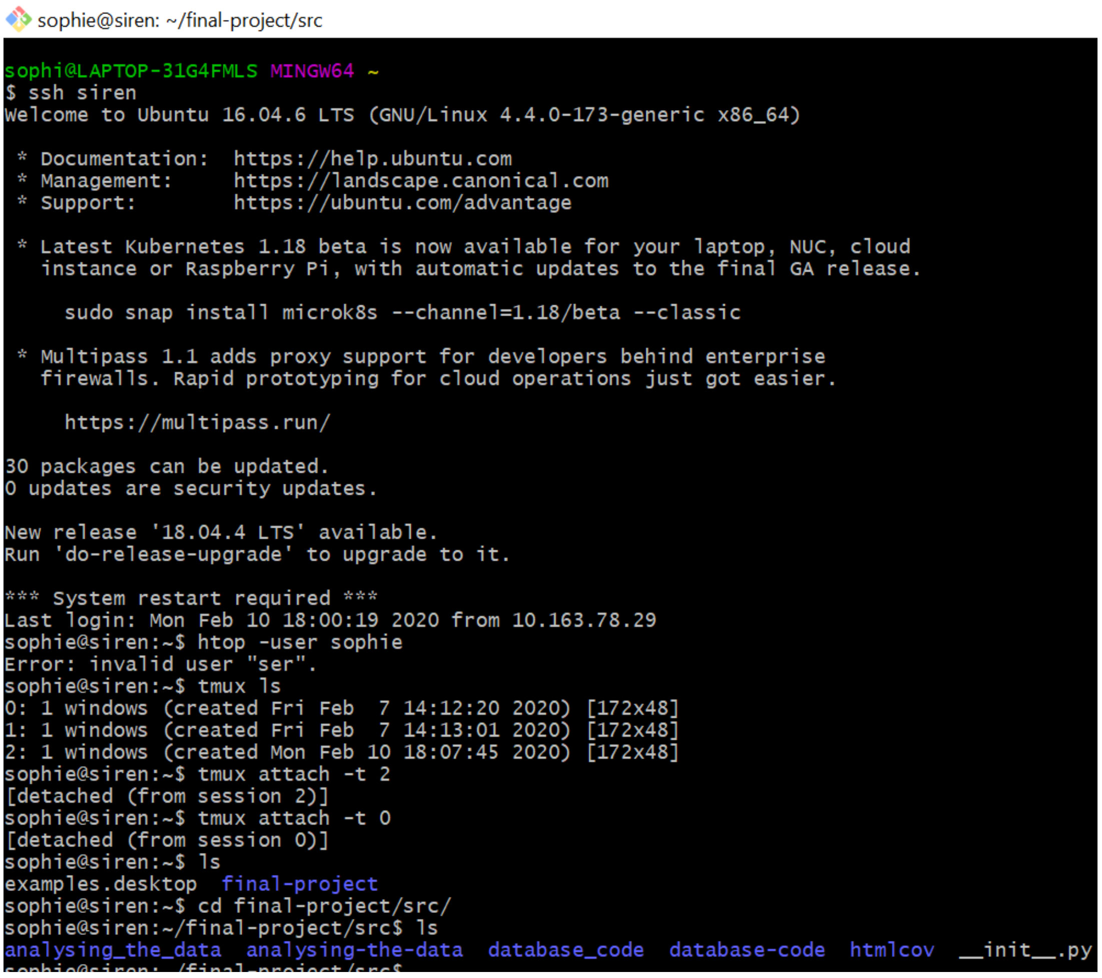
code, parallel processing is recommended, especially if the player set sizes being trialled are 'large'. See Section 3.5.3 for an explanation.

To connect to Siren, an SSH tunnel was used. An SSH (or Secure Shell) tunnel is used for sending / receiving network data over an encrypted connection. It adds a layer of network security to those applications that do not natively support encryption and lowers the risk of interception. For a more detailed description of SSH, see [45]. Figure 3.6 shows the author's connection to Siren via SSH.

In order for the experiment to keep running whilst disconnected from Siren, a terminal emulator was required, as Siren does not have a job scheduler. Specifically, the terminal multiplexer, TMUX [29] was used. This allowed for the creation and execution of several terminals from one screen. Moreover, a user could detach from a terminal and reattach later without execution being halted.

Figure 3.7, shows a diagram of how the remote server, ssh tunnel, and users laptop, were all in connection.

Once the experiment was running, the database had to be copied from Siren in order for the analysis to begin. This was also achieved via SSH. Note that, the code was written in a way which enabled the results to be written to the database concurrently, after every tournament. This was done to ensure that if the system were to break, data would still have been available and retained within the database. This meant the database file was almost always 'open' resulting



```
sophie@siren: ~/final-project/src
sophi@LAPTOP-31G4FMLS MINGW64 ~
$ ssh siren
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-173-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Latest Kubernetes 1.18 beta is now available for your laptop, NUC, cloud
   instance or Raspberry Pi, with automatic updates to the final GA release.

     sudo snap install microk8s --channel=1.18/beta --classic

 * Multipass 1.1 adds proxy support for developers behind enterprise
   firewalls. Rapid prototyping for cloud operations just got easier.

     https://multipass.run/

30 packages can be updated.
0 updates are security updates.

New release '18.04.4 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Mon Feb 10 18:00:19 2020 from 10.163.78.29
sophie@siren:~$ htop -user sophie
Error: invalid user "ser".
sophie@siren:~$ tmux ls
0: 1 windows (created Fri Feb  7 14:12:20 2020) [172x48]
1: 1 windows (created Fri Feb  7 14:13:01 2020) [172x48]
2: 1 windows (created Mon Feb 10 18:07:45 2020) [172x48]
sophie@siren:~$ tmux attach -t 2
[detached (from session 2)]
sophie@siren:~$ tmux attach -t 0
[detached (from session 0)]
sophie@siren:~$ ls
examples.desktop  final-project
sophie@siren:~$ cd final-project/src/
sophie@siren:~/final-project/src$ ls
analysing_the_data  analysing-the-data  database_code  database-code  htmlcov  __init__.py
```

Figure 3.6: A screenshot of the authors connection to Siren via an SSH tunnel.

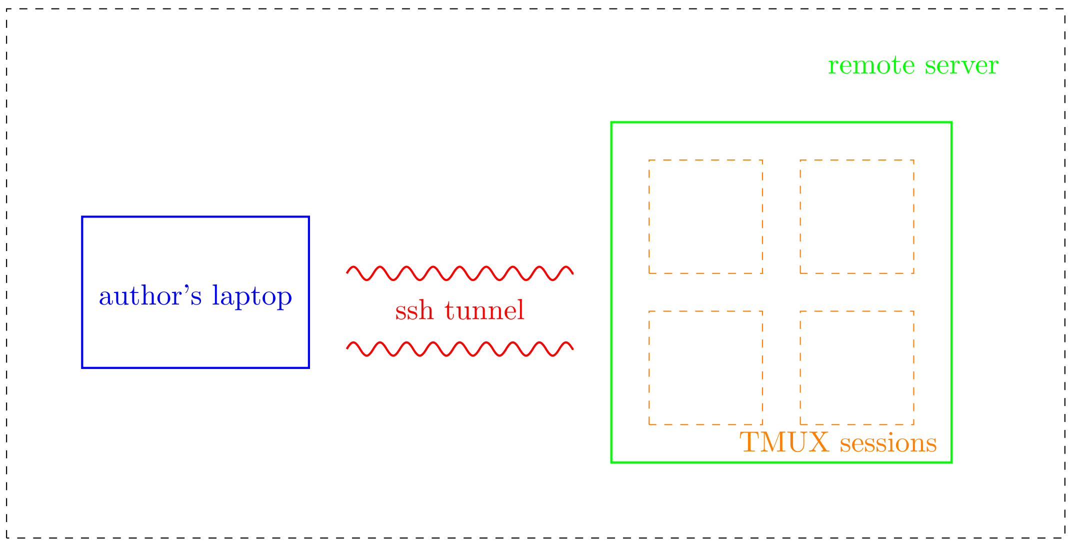
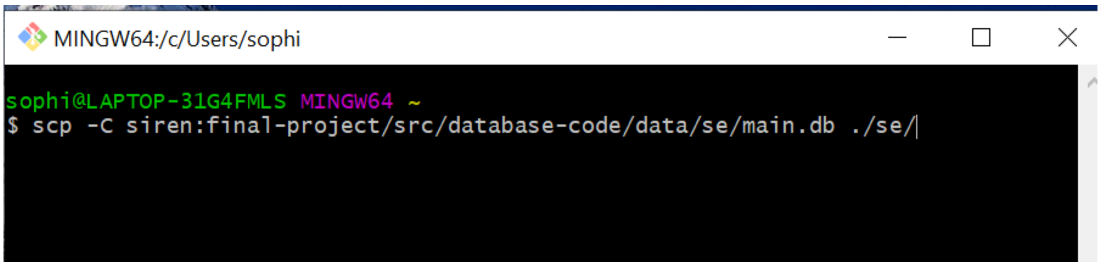


Figure 3.7: Representation of how the experiments were run remotely. Note, 'tmux sessions' correspond to emulators of terminals.



```
sophi@LAPTOP-31G4FMLS MINGW64 ~
$ scp -C siren:final-project/src/database-code/data/se/main.db ./se/
```

Figure 3.8: Command line code used to retrieve the database from the remote server.

in the integrity check failing once transferred over SSH and an OperationalError in sqlalchemy. Thus, in order to to load the database with no failures into a Jupyter Notebook, it needed to be compressed before transferring. This was achieved using the command line code as seen in Figure 3.8.

3.5 Calculating Nash Equilibria

Recall, Nash's Theorem, Theorem **??** explains that there exists at least one equilibrium in every finite game. However, it does not indicate how to obtain them. The proof of the theorem relies on finding the fixed point of the defined mapping however, the proof of Brouwer's Fixed Point Theorem is an existence, and not a constructive, proof. That is, it does not give a method for obtaining the fixed point. Indeed, although not NP-complete [1], finding Brouwer fixed points has been shown to be a hard problem [37, 17]. Thus, defining algorithms which obtain Nash equilibria to some degree of efficiency has been a large research topic for many years (include some evidence, i.e. paper references here?). Within the Python Library Nashpy [13], three such algorithms have been implemented: Support Enumeration, Vertex Enumeration and Lemke-Howson. However, the Lemke-Howson Algorithm will only find *one* equilibria and hence is not suitable for this study. Therefore, the definitions of the first two algorithms, in the case of a two player game, are provided. Unless specified otherwise, this section (Section 3.5) is adapted from [34].

[1]Both finding Brouwer fixed points and Nash equilibria cannot be NP-complete since existence of a solution is guaranteed [34]. Most problems in the set NP-complete are situations in which a solution might not exist [34]. However, [37] has shown that these two problems belong to a alternative complexity class, PPAD or Polynomial Parity Argument (Directed case). For a discussion into this, readers are referred to [37].

3.5.1 Support Enumeration

Before stating the method for the support enumeration algorithm, a few extra theoretical ideas are needed.

Recall, a *mixed strategy*, σ, is a probability distribution over the pure strategies. Then:

Definition 3.5.1. The *support* of σ is the set of all pure strategies, $s_i \in \sigma$, such that $s_i > 0$. That is, all pure strategies which have a positive probability within the mixed strategy.

Definition 3.5.2. A game $G = (N, (S_i)_{i=1,\dots,N}, (u_i)_{i=1,\dots,N})$ is called *non-degenerate* if no mixed strategy of support size $1 \leq k \leq |S_i|$ has more than k pure best responses.

For example, consider the following 2-player normal form game:

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 2 & 0 & 3 \end{pmatrix} \tag{3.2}$$

Here, if the column player was playing the strategy $\sigma_2 = (0.2, 0.8, 0)$ then the support is {strategy1, strategy2}. Also, observe that, if the column player picked strategy1 then the row player could choose either their first or second strategy and hence this game is *degenerate*.

The Algorithm

```
Algorithm 2: Support Enumeration
    input : A nondegenerate two-player normal form game, where $A, B$
            are the row and column player's payoff matrices and $\sigma_1, \sigma_2$
            are their strategy vectors, respectively.
    output: Every Nash equilibrium of the input game.
1   for all $k = 1, \dots, \min m, n$ do
2       for all support pairs $I, J$, with $I \in M, J \in N$ and $|I| = |J| = k$ do
3           Solve
                $\sum_{i \in I} \sigma_{1i} B_{i,j} = v,$ such that $\sum_{i \in I} \sigma_{1i} = 1, \sigma_{1i} \geq \mathbf{0}$ for all $j \in J$   (3.3)
            and
                $\sum_{j \in J} A_{i,j} \sigma_{2j} = u,$ such that $\sum_{j\ inJ} \sigma_{1i} = 1, \sigma_{1i} \geq \mathbf{0}$ for all $i \in I$   (3.4)
4           Check the best response conditions
                $\sigma_{1i} > 0 \implies (A\sigma_2)_i = \max\{(A\sigma_2)_k | k \in M\}$   (3.5)
            and
                $\sigma_{2j} > 0 \implies (\sigma_1 B)_j = \max\{(\sigma_1 B)_l | l \in N\}$   (3.6)
5       end
6   end
```

Note, solutions are not guaranteed to exist for Equations 1.3 and 1.4. In this case, the support does not yield a Nash Equilibrium.

A example

Here we consider the computation of Nash equilibria via support enumeration for a payoff matrix obtained from a two player IPD tournament executed during this experiment [2].

[2]Note, the Defector's opponent in this tournament is stochastic player, Inverse. Here, $p_e = 0.0514, \quad p_n = 0$. This game was not identified as degenerate.

Consider the following matrix:

$$A = \begin{pmatrix} 3.000 & 0.829 \\ 1.686 & 1.000 \end{pmatrix} B = A^T \tag{3.7}$$

Firstly, take $k = 1$, that is, looking for any pure best responses:

$$A = \begin{pmatrix} \underline{3.000} & 0.829 \\ 1.686 & \underline{1.000} \end{pmatrix} \quad B = \begin{pmatrix} \underline{3.000} & 1.686 \\ 0.829 & \underline{1.000} \end{pmatrix}$$

Thus, two pairs of pure best responses are visible, giving the following two Nash equilibria:

$$\sigma = \{(1, 0), (1, 0)\} \text{ and } \sigma = \{(0, 1), (0, 1)\}.$$

Since this is a two-player game, the only other support that needs checking is $I = J = \{1, 2\}$.

Here, Equations 1.3 and 1.4 become,

$$3\sigma_{r1} + 0.829\sigma_{r2} = 1.686\sigma r1 + \sigma_{r2} \text{ and } 3\sigma_{c1} + 0.829\sigma_{c2} = 1.686\sigma c1 + \sigma_{c2}.$$

Then, rearranging and checking the constraint $\sum \sigma_r = 1, \quad \sum \sigma_c = 1$, yields

$$\sigma_{r1} = \sigma_{c1} = \frac{19}{165} \text{ and } \sigma_{r2} = \sigma_{c2} = \frac{146}{165}$$

Note, checking the best response condition for two players with the same number of pure strategies is trivial [24]. However, it is included here for completeness.

$$A\sigma_c^T = \begin{pmatrix} 3 & 0.829 \\ 1.686 & 1 \end{pmatrix} \begin{pmatrix} \frac{19}{165} \\ \frac{146}{165} \end{pmatrix} = \begin{pmatrix} 1.079 \\ 1.079 \end{pmatrix} \text{ and } \sigma_r B = \begin{pmatrix} \frac{19}{165} & \frac{146}{165} \end{pmatrix} \begin{pmatrix} 3 & 1.686 \\ 0.829 & 1 \end{pmatrix} = \Big(1.07$$

Therefore, the best response condition holds for both players. Hence, the third and final Nash equilibrium is given by:

$$\sigma = \{(\frac{19}{165}, \frac{146}{165}), (\frac{19}{165}, \frac{146}{165})\}$$

Advantages and Drawbacks

Support enumeration is known to be a robust method for obtaining Nash That is, given a non-degenerate game, it is guaranteed to return all equilibria and, even in the case of degeneracy, it will find some equilibria (see Section 3.6). However, this method essentially compares all pairs of supports and thus has an exponential complexity [39]. This implies that support enumeration is computationally expensive and the larger the game becomes the slower it will become.

3.5.2 Vertex Enumeration

Vertex enumeration is based on a geometric representation of games and hence, here, a brief introduction to this is provided. The reader is referred to [34] for a more detailed approach to this topic.

Definition 3.5.3. Let A, B be *positive* payoff matrices for the row and column player; that is, each element $a_{ij}, b_{ij} > 0$,for all $i = 1, \ldots, M, j = 1, \ldots, N$. Then the row, column *best response polytopes* [3], denoted P, Q are given respectively by

$$P = \{x \in \mathbb{R}^M | x \geq \mathbf{0},\ xB \leq \mathbf{1}\} Q = \{y \in \mathbb{R}^N | Ay^T \leq \mathbf{1},\ y \geq \mathbf{0}\}. \tag{3.8}$$

Note, here it is assumed that the utility values which appear in 3 and 3 have been normalised to 1. This means that the vertices are no longer probabilities and hence scaling will be required to find the Nash equilibria. Also, the strictly positive payoffs is not a constraint since a constant can be added to each with no effect.

For example, consider the payoff matrix

$$A = \begin{pmatrix} 1 & 5 \\ 4 & 1 \end{pmatrix} \tag{3.9}$$

Then the best response polytope, P here is given by the inequalities: $x_1 \geq 0$,
$x_2 \geq 0$,
$x_1 + 5x_2 \leq 1$,
$4x_1 + x_2 \leq 1$
which yields the following as given in Figure 3.9.

[3]Note, in general, a *polytope* is defined as a bounded set $\{z \in \mathbb{R}^d | Cz^T \leq q\}$ where C is a $k \times d$ matrix and z is a $1 \times d$ vector [34].

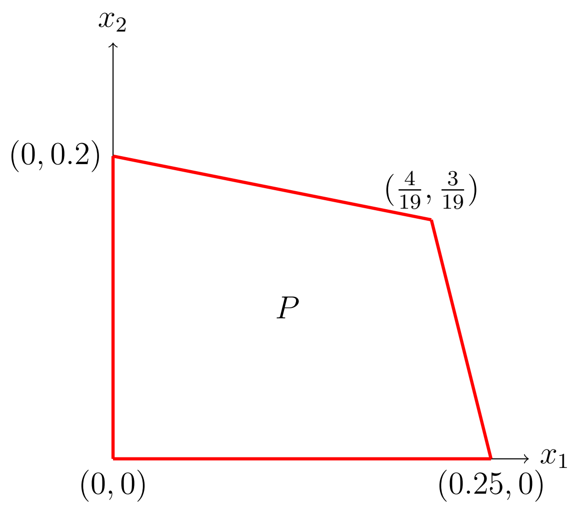


Figure 3.9: Best response polytope, P, obtained from the payoff matrix given in Equation 3.5.2.

The Algorithm

Two algorithms are stated here, the first 'prepares' the polytope for the second one, which returns the Nash equilibria.

Algorithm 3: Vertex Labelling

```
input  : A polytope, $P \in \mathbb{R}^n$.
output: A labelled-vertex polytope.
1 enumerate each of the defining inequalities of $P$, $c_1, \dots, c_k$
2 for each vertex $v_i \in P$ do
3     find the inequalities of $P$ which are binding at $v_i$, that is the
        defining equations are equalities.
4     the label of $v_i$ is given by $\{c_{i1}, \dots, c_{il}\}$, where $c_{ij}$ is in the label if
        and only if equation $c_j$ is binding for $v_i$
5 end
```

A pair of labels $v_i \in P$, $u_j \in Q$ are called *fully labelled* if every inequality 'number' of $P \cup Q$ appears in either the label of v_i or the label of u_j. Then there is a notion which states that each fully labelled pair, when normalised, corresponds to a Nash equilibrium. Note, this does not include the vertex pair $(\mathbf{0}, \mathbf{0})$ since, although this is a fully labelled pair, it corresponds to neither opponent playing any strategies.

Algorithm 4: Vertex Enumeration

```
input  : Best response polytopes, $P, Q$, as defined in 3.5.2, for a
         nondegenerate game.
output: All Nash equilibria of the corresponding game.
1  for each polytope, $P, Q$ do
2      Execute Algorithm 3
3      for each pair of vertices $\{u_i, v_j\}$ in $P, Q$ respectively, except
        $(\mathbf{0}, \mathbf{0})$ do
4          check if they are fully labelled. if they are fully labelled then
5              add to the list of equilibria
6          end
7          else
8              continue
9          end
10     end
11 end
```

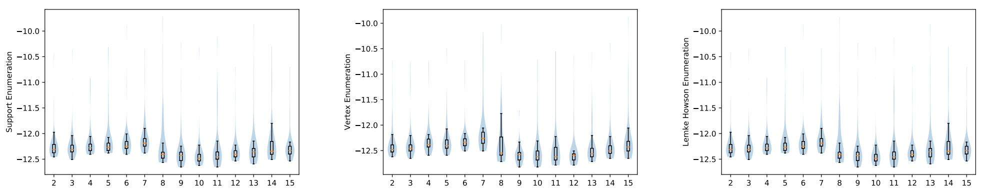
Advantages and Drawbacks

Vertex enumeration is more efficient than support enumeration, according to [34], since there are more supports in a game than there are vertices. Indeed, consider the example of $M = N$ where M, N are the number of binding inequalities in the best response polytopes, P, Q, respectively. Using the support enumeration algorithm, approximately 4^n support pairs will need to be looked at but, according to the 'Upper Bound Theorem' for polytopes [42, 2, 8], P and Q have less than 2.6^n vertices. Thus, given there exists an efficient method for enumerating vertices, then this implies less further computational complexity. However, if the game is degenerate, this algorithm will not return any Nash equilibria.

3.5.3 Algorithm Execution Timings

Due to the robustness of the support enumeration, it was decided that this would be the main method of calculating Nash equilibria. On the other hand, vertex enumeration was also included in a different run of the experiment, in order to compare the accuracy of the methods - though this algorithm does break if the algorithm is degenerate.

Having said this, timings of each of the three algorithms were obtained, to see if the computational complexity is significantly different for any of them. For the



(a) Log timings of the calculation of Nash equilibria using the support enumeration algorithm.

(b) Log timings of the calculation of Nash equilibria using the vertex enumeration algorithm.

(c) Log timings of the calculation of Nash equilibria using the Lemke-Howson enumeration algorithm.

Figure 3.10: Violinplots of the log timings obtained for the experiment and calculation of Nash equilibria using the three algorithms as implemented in Nashpy.

purposes of this trial run, the following parameters were used: 100 tournament repetitions, $p_e = 0.2$ and $p_n = 0$. The results can be seen in Figure 3.10.

From Figure 3.10, it can be seen that there is not a significant difference in the execution times of the algorithms. Hence, although support enumeration has a greater computational complexity than vertex enumeration - it is not going to have any recognisable impact on the number of experiments executed within the time frame.

3.6 Degeneracy

Recall, the definition of nondegeneracy, given in Definition 3.5.1. This implies a *degenerate* game is one in which there exists a support of size k where the number of pure best responses is greater than k. For example, consider the following payoff matrix:

$$A = \begin{pmatrix} 1 & 4 & 3 \\ 0 & 4 & 2 \end{pmatrix} \tag{3.10}$$

Then, if the column players picks their second strategy (a support of size 1), the row player can pick either of their strategies (two pure best responses). Hence this game is degenerate.

Let $G = (N, S_{ii\in N}, u_{ii\in N})$ be a degenerate game. Then, recall that support enumeration may not return all Nash equilibria. This is due to the fact that if solutions to the equations given in 1.3 and 1.4 of Algorithm 2 exist, they may not be unique. Indeed, the number of Nash equilibria in a degenerate game may be infinite [34]. Considering degeneracy in terms of vertex enumeration, means

```python
with warnings.catch_warnings(record=True) as w:
    warnings.simplefilter("always")

    if support_enumeration is True:
        nash_equilibria = list(game.support_enumeration())

    else:
        highlight_numpy_warning = np.seterr(all="warn")
        nash_equilibria = list(game.vertex_enumeration())

if len(w) == 0:
    warning_message = None

else:
    warning_message = str([w[i].message for i in range(len(w))])
```

Listing 2: Python code used to 'catch' potential degeneracy.

that a vertex of the best response polytope $P = \{x \in \mathbb{R}^M | x \geq \mathbf{0},\ xB \leq \mathbf{1}\}$ may have more than M labels leading to a 'badly defined' polytope. This implies the algorithm will break (need to double check).

Within the library Nashpy, the algorithms have been implemented such that if potential degeneracy is identified, for example by the reasons given above, then a warning is issued. Thus, in order to retain whether a game is possibly degenerate, the algorithm was required to 'catch' the warnings when given. This was achieved using the code seen in Listing 2, with the Python warnings module:

Note, warnings for potential degeneracy are only highlighted if the Nash equilibria obtained do not make sense, that is, are not a probability distribution. However, it is possible for the algorithm to obtain some correct Nash equilibria, but the game could still be degenerate. This means that, when stating statistics regarding the degeneracy of the games obtained during the experiment (see Chapter **??**), these are only the ones caught by the algorithms in Axelrod and are not necessarily all of them.

In this chapter, the creation and execution of the empirical experiment was detailed in its entirety. Justifications were given as to why certain methods and software were used and / or not used. Also, how the Nash equilibria, explored in the next chapter, was explained with an example given. (Bullet point list of main conclusions?) In Chapter **??**, an introductory analysis of the data obtained is given.

Chapter 4

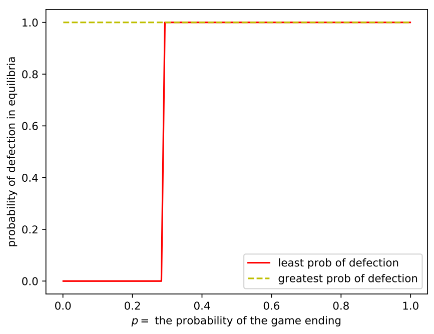
Analyses

In this chapter, an analysis of the data collected via the methods described in the previous chapter (Chapter **??**) is given. Firstly, a brief initial overview is provided, where descriptive statistics of the equilibria obtained and the overall characteristics of the strategies used is discussed. Following this, a critical analysis of the *p*-thresholds obtained is carried out. Here, the environmental effects, on the outcome of the game, discussed include: number of opponents and level of added noise. Note, as of writing, the database currently has 825700 entries (rows) and a total number of 159 tournament sets.

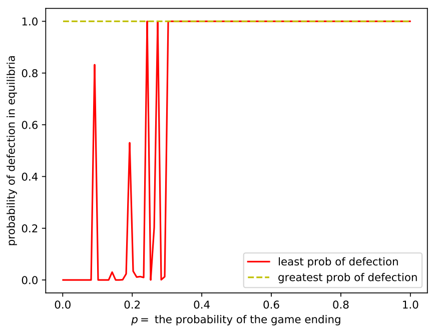
4.1 Initial Analysis

In this section, all the data (including those games which could be degenerate) are considered. Taking a brief look at the graphs produced for each set of tournaments, it can be seen that the main 'shapes' obtained are as seen in Figure 4.1.

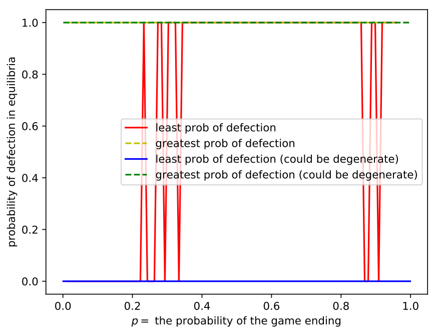
In Figure 4.1a, a clear p-threshold of approximately 0.28 is apparent, clearly visualising the Folk Theorem. In this tournament set the opponent was *Inverse*, a stochastic player, indicating that perhaps the stochasticity of a player does not affect the accuracy of the threshold as first thought. In Figure 4.1b, the precise value of the p-threshold is less clear, lying approximately in the range [0.1, 0.3]. This could be due to the potential degeneracy identified or just the element of randomness that appears within each tournament, possibly magnified by the number of stochastic players. The opponents within this set were: *Feld: 1.0, 0.5, 200*; *Cooperator*; *EvolvedLookerUp2_2_2*; *Tullock: 11*; and *ZD-GEN-2: 0.125, 0.5, 3*. Figure 4.1c shows a potential problem with the visualisation of the Folk Theorem when degenerate games are involved. It becomes unclear as



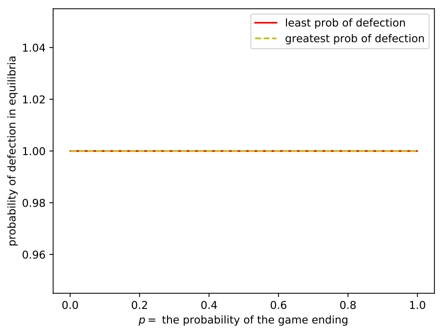
(a) 2-player tournament set with no added noise, one stochastic player and no degeneracy identified.



(b) 6-player tournament set with no added noise, three stochastic players and potential degeneracy was yielded from 10 of the tournaments.



(c) 5-player tournament set with no added noise, two stochastic players and five tournaments played yielded potentially degenerate games.



(d) 4-player game with no added noise, one stochastic player and no degeneracy identified.

Figure 4.1: Example graphs obtained from the experiment.

to what is happening in the graph, especially where the p-threshold lies. The opponents, in this case, were: *Random: 0.5*; *Grumpy: Nice, 10, -10*; *Fortress3*; and *Negation*. Finally, Figure 4.1d gives an example of a tournament set for which there was always a non-zero probability of defection, regardless of the game ending probability. In this case, the precision of game-ending probabilities chosen was not accurate enough to identify the p-threshold. It is implied that the tournament has to have an ending probability of almost 0 (within the interval (0, 0.001) in order for a zero probability of defection to be rational. Here, the opponents of the *Defector* were: *AntiCycler*; *e*; and *Stalker: (D,)*. Similarly, it is observed that some graphs obtained were constant at 0, again indicating that the precision of game-ending probabilities were not fine enough to highlight the p-threshold. This indicates that the ending probability of the tournament has to lie within the interval (0.999, 1). That is, almost immediately, the decision to defect is no longer rational.

The summary statistics gained from running the *describe* method of a pandas database is given in Table **??**. From this, it can be seen that the number of opponents the *Defector* played against ranged from one to seven, with an average of four opponents. Also, as expected, the mean probability of the game ending encountered was 0.5. Observe that, overall, there were $175,399$ distinct tournaments played with a total of 159 distinct sets of strategies. Looking now at the statistics for Nash equilibria, it can be seen that a total of $823,823$ equilibrium points were calculated in this experiment, with an average of $1.914 \approx 2$ equilibria per game. However, observe, at least one game obtained 39 equilibria which will be explored into later on in this section. Considering the probabilities of defection within these equilibria, notice that both the greatest and the least probabilities of defection ranged from zero to one inclusive with a 50th percentile of zero. But, looking at the average values, the least probability has a mean of 0.342 and only just above this, the greatest probability has a mean of 0.460.

Next, further descriptive statistics are calculated for the strategies. This is to obtain a more in-depth view on the types of strategies randomly chosen to play and their characteristics. Executing *value_counts* method on the column of strategy names, it is observed that the player which appeared the most times (9 times) is *ZD-GEN-2: 0.125, 0.5, 3*; followed closely by *Tideman and Chieruzzi* with 7 sets of tournaments. On the other hand 38 out of the 200 strategies playing in this experiment appeared only once. Running the *value_counts* method again, but this time on the memory depths of the strategies found the majority of strate-

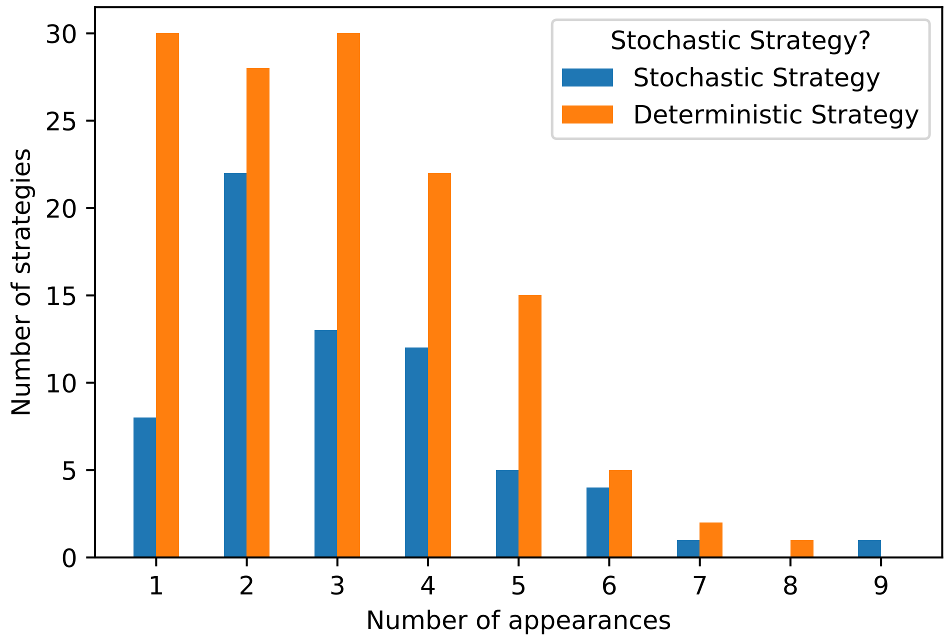


Figure 4.2: A plot to show the ratio of stochastic to deterministic strategies randomly chosen throughout the experiment.

gies to have an infinite memory depth. On the other hand, strategies having no memory or a depth equal to one were also significant. Considering the stochasticity of players alongside how many appearances each strategy made yielded the following chart in Figure 4.2.

It is clear that there is a clear bias towards deterministic strategies in this experiment. However, this is to be expected as running the following code:

```
len(axl.filtered_strategies(filterset={"stochastic": True})),
len(axl.filtered_strategies(filterset={"stochastic": False}))

(86, 156)
```

it can be seen that over half the strategies coded into the Axelrod library are classed as deterministic. Looking at Figure 4.2 again, observe, the majority of deterministic strategies were executed either once or three times. On the other hand, a large proportion of the stochastic strategies were played twice.

Further, the number of Nash equilibria obtained for each game was analysed and their distributions with respect to the number of opponents of the *Defector* plotted. Executing the *value_counts* method on the 'num_of_equilibria' column gave the conclusion that the majority of games (131773) yielded one Nash equilibria with 28793 games obtaining 3 equilibria. Also, the maximum number of equilibria yielded, 39, was for one game, which, doing a search on the database, was found to be a six player game with noise=0.1. The opponents of the *Defector* were: *Inverse Punisher*; *Prober*; *PSO Gambler 2_2_2 Noise 05*; *Handshake*; and *More Tideman and Chieruzzi*. This game was expected to be degenerate however, when taking a closer look at this entry in the database, no degeneracy was identified. This could be worth looking at in further investigation of the work.

Figure 4.3 shows the distributions of the number of Nash equilibria as per the number of players. This visualisation turns out to not be extremely revealing — possibly as an effect of degeneracy — however some insights can be found here. Observe, all of the distributions observed on this plot have a clear modal value of one, that is, irrespective of the number of players, the majority of games yielded only one equilibria. Moreover, there also seems to be an increase in density around 3 equilibria which becomes more prominent as the number of players increases. As can be seen from the plot, the variance in the number of equilibria increases with the number of players, apart from when there were 6 players (5 opponents of the defector), where the spread is maximum. This could be due to the 39 equilibria gained for one game as discussed in the paragraph above. Looking now at the mean of the distributions, observe that these are also slightly increasing as the number of players increase.

4.2 Analysis of the p-Threshold

Firstly, for clarity, here is a restatement of the definition of a *p-threshold*: The probability of the tournament ending for which the least probability of defection in Nash equilibria is non-zero.

In order to analyse the p-thresholds of the tournaments, a csv file was created [1] containing the minimum, mean, median and maximum probabilities for each set of tournaments. This was in order to gain as much information as possible from tournaments which gave graphs such as Figure 4.1b as described above. Within this file, other than the varying thresholds, the information about the number of players, tournament strategy and level of noise was retained. Moreover, it contained a column which identified whether any of the strategy sets lead to possible degenerate games.

Now, an exploration into the overall p-thresholds is given.

[1]Please see Appendix **??** for the code used to obtain these files.

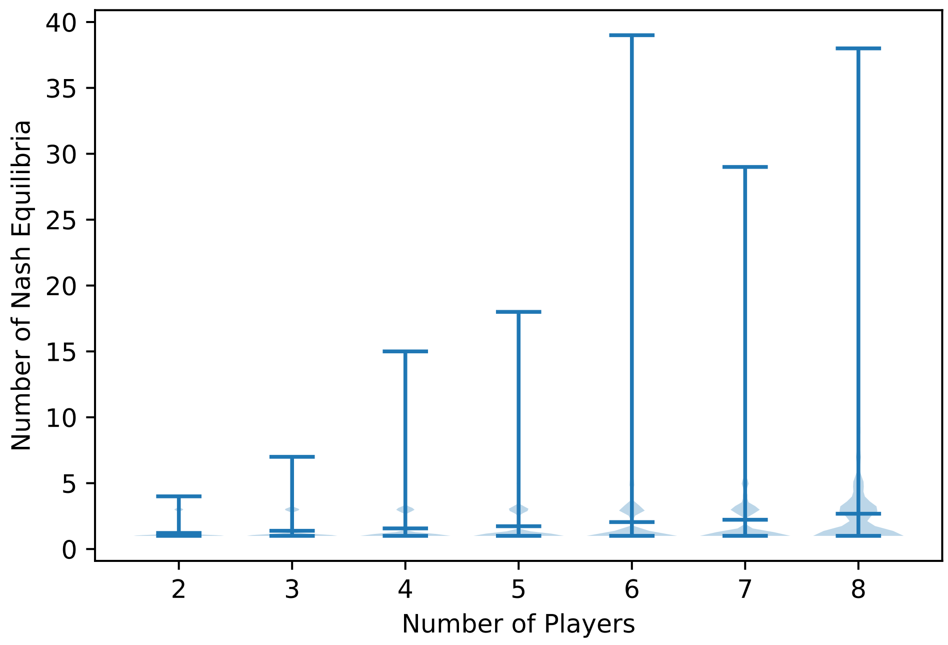
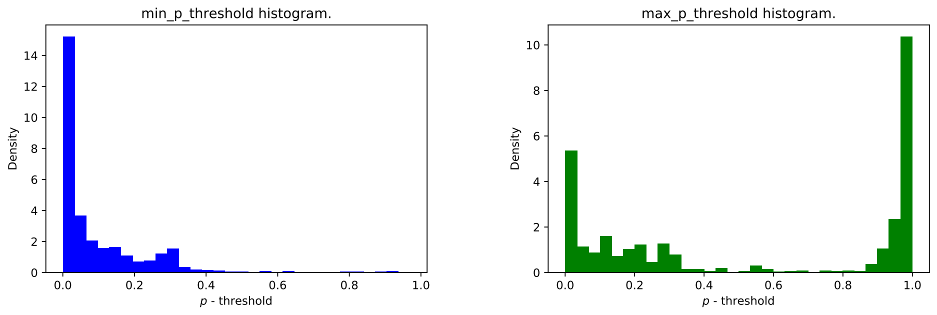


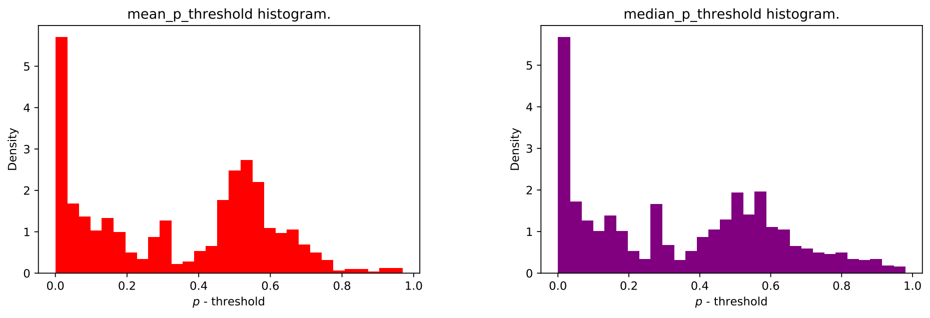
Figure 4.3: A violinplot showing the distribution of the number of equilibria obtained for varying number of players.



(a) A plot to show the minimum p-thresholds.

(b) A plot to show the maximum p-thresholds.

Figure 4.4: Plots to show the p-thresholds for all 1001 sets of tournaments.



(a) A plot to show the mean p-thresholds. (b) A plot to show the median p-thresholds.

Figure 4.5: Plots to show the p-thresholds for all 1001 sets of tournaments.

Observe, in Figure 4.4a, the majority of minimum thresholds were less than 0.4, with a clear modal value of 0. That is, in a significant proportion of the tournaments, there was no probability of the game ending for which the probability of defection was zero. Now considering Figure 4.4b, it can be seen that the modal value for the maximum threshold is 1. Also, in comparison with the minimum threshold, there is a larger spread in the density. Yet, there is still a peak around zero as with the minimum threshold data.

Looking at the mean and median threshold data, Figure 4.5, observe that the distributions obtained are similar with, again, a modal value of zero, indicating there is always a chance of defection irrespective of the game ending probability. However, in these histograms there is also a clear peak around a threshold of 0.5. This suggests that some of the tournaments may have several p-thresholds, perhaps due to stochasticity, degeneracy or just inevitable randomness that appears in the tournaments. Indeed, obtaining the minimum and maximum p-thresholds for those tournaments where the mean threshold was within the range $[0.5, 0.6]$, it can be seen that, from Figure 4.6 for a significant proportion, their minimum threshold was around zero and their maximum around one. Moreover, looking closer at three of the tournaments which satisfied the above, there is a clear randomness within the corresponding graphs, Figure **??**.

The plots contained in Figure 4.7 were sampled randomly using the random library in Python. What is interesting here is that they all contain varying amounts of additional noise and further analysis into this would be beneficial in seeing if this is one of the main causes for the inaccuracy of the thresholds. For example, looking at how many of the tournaments with this property had an added noise level.

Before continuing onto the general overview of the thresholds for those tourna-

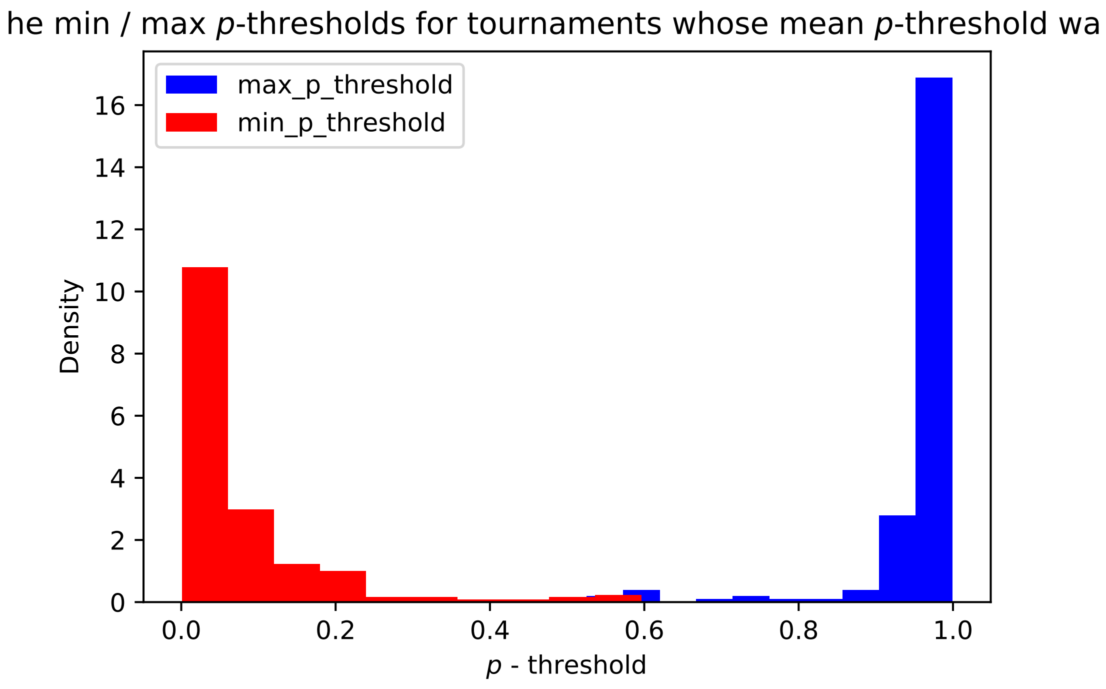
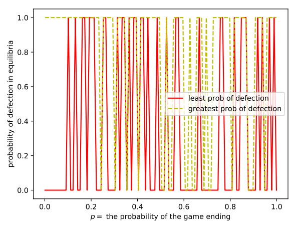
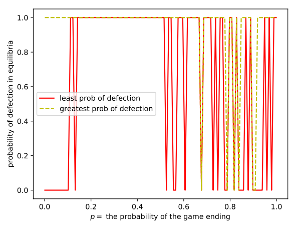


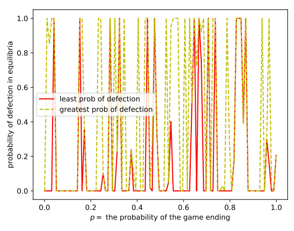
Figure 4.6: A plot to show the minimum and maximum p-thresholds for those tournaments which had an mean threshold within the range $[0.5, 0.6]$.



(a) 6-player tournament set with an additional noise level of 0.2, two stochastic players and no degeneracy was identified.

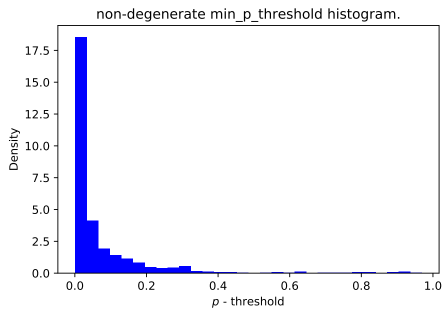


(b) 3-player tournament set with an additional 0.1 of noise, one stochastic player and no degeneracy was identified.

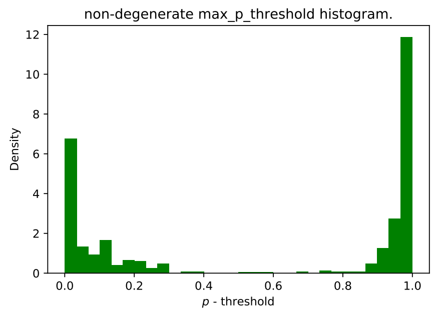


(c) 3-player tournament set with an additional noise level of 0.5, one stochastic player and only one tournament yielded potential degeneracy, with a game-ending prob of 0.9788383838383838384.

Figure 4.7: Example plots of the tournaments where the mean p-threshold was within the range $[0.5, 0.6]$.

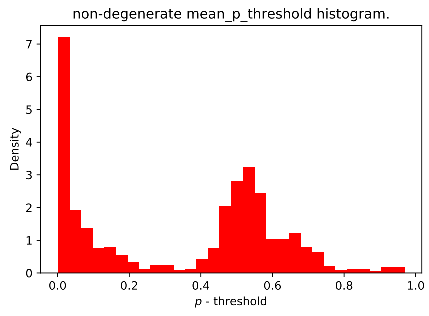


(a) A plot to show the minimum p-thresholds.

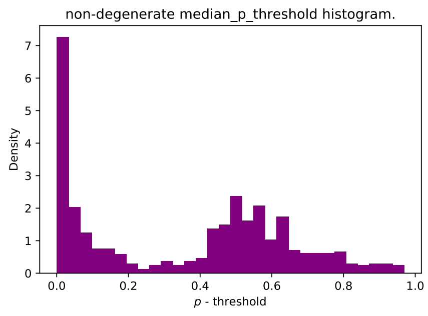


(b) A plot to show the maximum p-thresholds.

Figure 4.8: Plots to show the p-thresholds for all tournaments which led to non-degenerate games.



(a) A plot to show the mean p-thresholds.



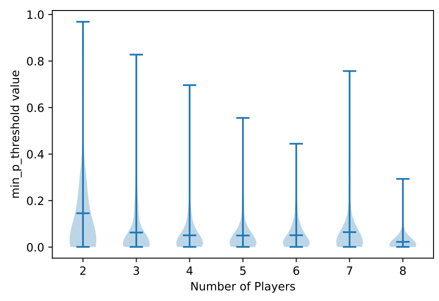
(b) A plot to show the median p-thresholds.

Figure 4.9: Plots to show the p-thresholds for all tournaments which led to non-degenerate games.

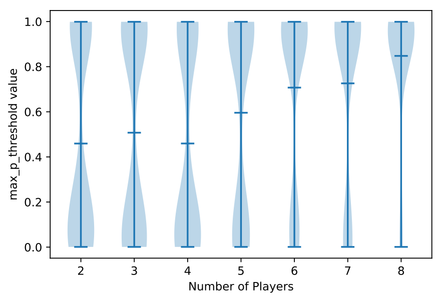
ments which led to definite non-degenerate games, a brief point is made regarding those tournaments which, for all probabilities of the game ending, had no positive probability of defection. Indeed, out of the 1754 total tournaments 753 of them had the above property of no apparent threshold (that is, lie within the unobserved interval of (0.999, 1)). That is, the plots obtained for these tournaments were (approximately). Further investigation into these tournaments, as well as those in which the probability of defection was always positive, is highly recommended.

Regarding degeneracy, out of all 1754 tournaments, 372 were highlighted as potentially leading to degenerate games. These are omitted from the following plots in order to focus solely on non-degenerate games.

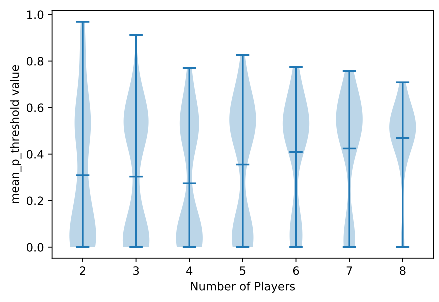
Comparing Figures 4.8a, 4.8b, 4.9a and 4.9b with Figures 4.4a, 4.4b, 4.5a and 4.5b, respectively, it can be seen that, in general, there is no significant change in the



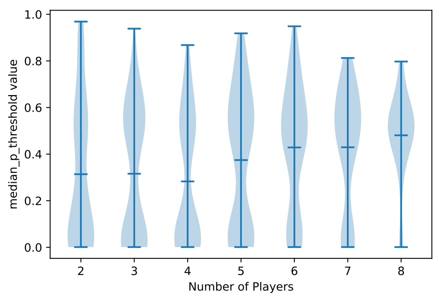
(a) Minimum p-threshold violinplot.



(b) Maximum p-threshold violinplot.



(c) Mean p-threshold violinplot.



(d) Median p-threshold violinplot.

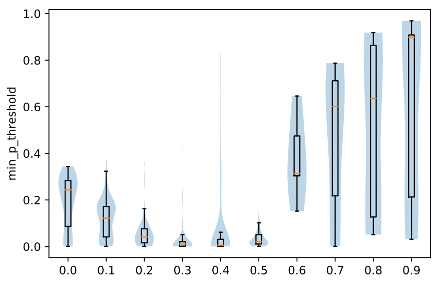
Figure 4.10: Violinplots of the thresholds for each number of opponents.

distributions of the thresholds. But, there is a more prominent peak in Figure 4.5a around 0.3 than in the corresponding non-degenerate plot of Figure 4.9a. Further work regarding the effects of degeneracy is advised as the above discussion seems to indicate that degeneracy does not seem to have as much of an affect on the thresholds as initially expected.

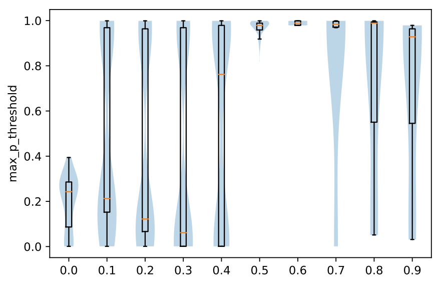
4.2.1 Effects of the Number of Players

In this section, the p-thresholds will be analysed with respect to the number of opponents the *Defector* played against. Note, in this section, only non-degenerate tournaments will be considered.

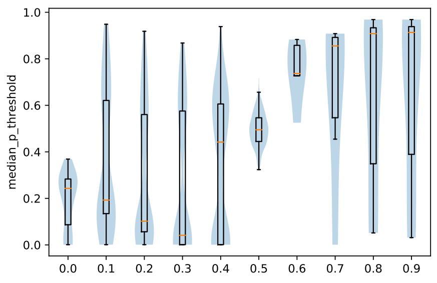
From Figure 4.10c, it can be seen that the distributions of the minimum p-thresholds with respect to the number of players all have a modal value around 0. However, apart from the 7-player tournaments, the spread of the distributions decrease, along with the mean values. Considering the maximum thresholds, Figure **??**, the distributions become bimodal with mode values of around 0 and 1. But, as the number of players increases the modal value at zero becomes less prominent with the 8-player tournament distribution not having a mode around 0. The variance of the distributions are similar and, apart from 4-player tournaments, the means increase with the number of players. Looking now to the



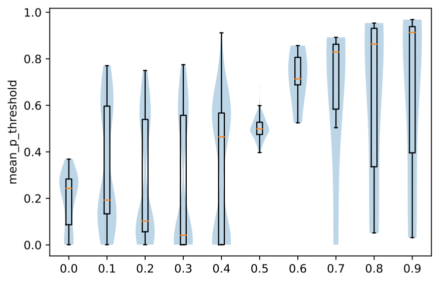
(a) Minimum p-threshold violinplot.



(b) Maximum p-threshold violinplot.



(c) Median p-threshold violinplot.



(d) Mean p-threshold violinplot.

Figure 4.11: Violinplots of the thresholds for each level of noise.

mean and median violinplots, Figures 4.10c and 4.10d respectively, there is no significant difference between the two plots other than the spread is a little larger in the median thresholds. Within the mean plot, Figure 4.10c, it can be seen that the distributions also start off bimodal, at 0 and approximately 0.5. But again, the modal value at zero becomes less distinct with the 8-player tournament distribution being unimodal. The modal value at around 0.5 is a consequence of the reason stated in the previous section. Moreover, observe that, apart from 4-player tournaments, the variance of the distributions seem to decrease with the size of the player set whilst the means increase from a value of approximately 0.3 for 2-player tournaments to around 0.5 for 8-players. Looking at Figure 4.10 as a whole, it is implied that the number of players has no significant effect on the value of the p-threshold.

4.2.2 Effects of Noise

Here, an analysis on the effects of noise on the p-threshold is provided. The addition of noise to a tournament indicates that, with a certain probability, the action of a particular strategy is altered [16]. That is, an action of *coop* changes to *defect* and vice versa.

Figure 4.11a, shows the distribution of the minimum p-thresholds for each level of additional noise added to the tournaments. Here, it can be noted that the distributions of noise levels at least 0.6 have a large variance, indicating that adding a too large noise probability is highly random and thus no conclusions can be drawn. On the other hand, considering the noise levels less than 0.6, observe that the mean p-thresholds decrease from around 0.25 to approximately 0 as the noise increases. These distributions are also clearly unimodal. Looking now at Figure 4.11b, it can be seen that the majority of distributions have a much larger spread here varying over the full spectrum of game-ending probabilities and similar observations can be made from Figures 4.11c and 4.11d.

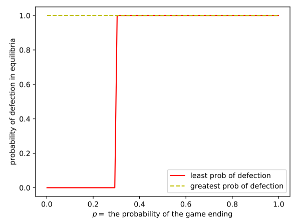
Figure 4.12 shows an example of the same tournament set through a variety of differing noise levels. Indeed, here, it can clearly be seen that the amount of noise does effect the p-threshold. However, this is to be expected since, by definition, as the noise level increases, the *Defector* will be observed as similar to the *Cooperator* when there is no additional noise.

Therefore, on the whole, there are not many significant conclusions that can be made here. This implies that the addition of noise to an already random tournament obscures any possible visions of the threshold, especially as the magnitude increases.

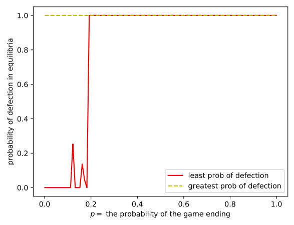
4.3 Conclusions and Further Work

In this section, the beginnings of an analysis into the p-thresholds was discussed. The effects of the number of players and level of additional noise on these thresholds were the main focus with a brief discussion regarding the degeneracy of games also given. This turned out to be a non-trivial task due to the proportion of tournament sets which yielded potential degeneracy at certain game ending probabilities and the inevitability of randomness within the tournaments. However a few points of interest were highlighted and these are summarised here.

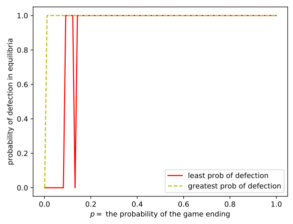
Firstly, the potential degeneracy of games yielded by the tournaments, which was highlighted as a potential factor, at first glance appears to not create as much of an issue as originally thought. The histograms of the p-thresholds, when the tournaments including potential degeneracy were omitted, did not have any significant changes when compared to the original histograms of all tournaments. However, further exploration is advised here. Regarding the number of players in a tournament, it was initially hypothesised that this would be a key factor in the variability of the p-threshold. However, on obtaining the distributions of the thresholds for tournament sets of size one to eight, it was implied that



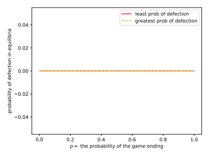
(a) 6-player game with no additional noise.



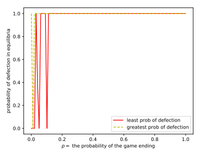
(b) 6-player game with an additional noise level of 0.1.



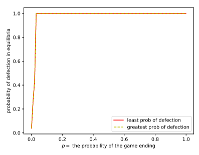
(c) 6-player game with noise level 0.2.



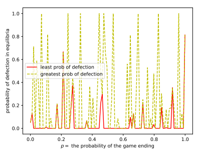
(g) 6-player game with an additional noise level of 0.6. Note, the remaining noise levels of 0.7, 0.8, 0.9 and 1.0 all yielded the same constant 0 graph as represented here.



(d) 6-player game with noise level 0.3.



(e) 6-player game with an additional noise level of 0.4.



(f) 6-player game with noise level 0.5.

Figure 4.12: Observation of one 6-player tournament set through the varying levels of additional noise. There was one stochastic player and 13 out of the 1100 tournaments played yielded potential degenerate games. The opponents here were: *Getzler*; *Punisher*; *Forgiver*; *GrudgerAlternator*; and *GraaskampKatzen*.

the number of players does not have a significant impact on the value of the threshold. Finally, the effect of additional noise on the p-threshold was analysed. Here, it was observed that, as expected, the level of additional noise did affect the p-threshold however there was no significant trends appearing out of the randomness.

As stated above, this is only the very start of an analysis into the p-thresholds described by the ‘original’ Folk Theorem and the effects of the varying environmental factors which appear in tournaments of the IPD. Thus many questions regarding this are still to be researched and a few recommendations regarding further work are now given. Firstly, it was observed that there were a significant proportion of tournaments for which the graph remained constant at zero or one. That is, the p-threshold was not identified using the precision of game-ending probabilities chosen and therefore must lie within the intervals (0, 0.001) or (0.999, 1), respectively. Hence, these tournament sets could be rerun with a much finer precision within the appropriate intervals to highlight exactly what is happening here. Also, an analysis into the characteristics of the strategies involved and the additional noise levels included in the tournaments could provide a clearer insight into potential reasons for this. Moreover, with regards to analysing the characteristics of players, it is suggested that those tournament sets in which stochastic players were included could be removed and the tournaments rerun. This could help in revealing whether the stochasticity of the player has any effect on the threshold, as the author has hypothesised.

To check the reliability of the data collected, a second experiment is recommended using a different algorithm for calculating the Nash equilibria, for example vertex enumeration. This could be used in comparison with the data already collected to identify whether the algorithms are producing the same Nash equilibria or, more importantly, whether they identified the same games as being degenerate. Furthermore, it is suggested that this experiment be executed with a larger number of tournaments repeats (greater than 500) to observe whether this ‘smooths’ the payoff matrices with greater success to enable for a clearer visualisation of the p-thresholds. Finally, some multivariate data analysis of the results, for example regression, could provide some more insights into this topic.

Overall, this chapter has been successful in visualising the Folk Theorem using the data collection setup as explained in Chapter **??** and using the plots obtained as in Figure 4.1a.

References

[1] IB Adeoye et al. “Application of game theory to horticultural crops in south-west Nigeria.” In: *Journal of Agricultural and Biological Science* 7.5 (2012), pp. 372–375.

[2] N. Alon and G. Kalai. “A Simple Proof of the Upper Bound Theorem”. In: *European Journal of Combinatorics* 6.3 (Sept. 1985), pp. 211–214. DOI: 10.1016/s0195-6698(85)80029-9.

[3] Robert Axelrod. “Effective choice in the prisoner’s dilemma”. In: *Journal of conflict resolution* 24.1 (1980), pp. 3–25.

[4] Kalid Azad. *A little diddy about binary file formats.* [Accessed on:04/03/2020]. Better Explained. Apr. 16, 2005. URL: https://betterexplained.com/articles/a-little-diddy-about-binary-file-formats/.

[5] Margherita Barile. *Open Cover.* [Accessed on; 06/01/2020]. Wolfram MathWorld. URL: http://mathworld.wolfram.com/OpenCover.html.

[6] Ned Batchelder. *Coverage.py 5.0.3.* Version 5.0.3. Jan. 12, 2020. URL: https://github.com/nedbat/coveragepy.

[7] Michael Bayer. “SQLAlchemy”. In: *The Architecture of Open Source Applications Volume II: Structure, Scale, and a Few More Fearless Hacks.* Ed. by Amy Brown and Greg Wilson. aosabook.org, 2012. URL: http://aosabook.org/en/sqlalchemy.html.

[8] A. Brondsted. *An Introduction to Convex Polytopes.* Graduate Texts in Mathematics. Springer New York, 2012. ISBN: 9781461211488. URL: https://books.google.co.uk/books?id=7PXxBwAAQBAJ.

[9] Alex Burke. *The Advantages of a Relational Database Over a Flat File.* [Accessed on: 04/03/2020]. bizfluent. Sept. 26, 2017. URL: https://bizfluent.com/list-7269497-advantages-database-over-flat-file.html.

[10] Stephan C. Carlson. *Brouwer’s fixed point theorem.* [Accessed on: 06/01/2020]. Encyclopaedia Britannica. Sept. 13, 2016. URL: https://www.britannica.com/science/Brouwers-fixed-point-theorem.

[11] Bor-Sen Chen, Chia-Hung Chang, and Hsiao-Ching Lee. "Robust synthetic biology design: stochastic game theory approach". In: *Bioinformatics* 25.14 (2009), pp. 1822–1830.

[12] E. F. Codd. "A Relational Model of Data for Large Shared Data Banks". In: *Software Pioneers*. Springer Berlin Heidelberg, 2002, pp. 263–294. DOI: `10.1007/978-3-642-59412-0_16`.

[13] The Axelrod project developers. *Axelrod: v4.7.0*. Apr. 2016. DOI: `10.5281/zenodo.3517155`. URL: `http://dx.doi.org/10.5281/zenodo.3517155`.

[14] Oxford English Dictionary. *game theory, n.* Online. Accessed: 08.10.2019. Mar. 2013. URL: `https://www.oed.com/view/Entry/319028?redirectedFrom=game+theory&`.

[15] James W Friedman. "A non-cooperative equilibrium for supergames". In: *The Review of Economic Studies* 38.1 (1971), pp. 1–12.

[16] Nikoleta E. Glynatsi and Vincent A. Knight. *A meta analysis of tournaments and an evaluation of performance in the Iterated Prisoner's Dilemma*. 2020. arXiv: `2001.05911 [cs.GT]`.

[17] Michael D Hirsch, Christos H Papadimitriou, and Stephen A Vavasis. "Exponential lower bounds for finding Brouwer fix points". In: *Journal of Complexity* 5.4 (Dec. 1989), pp. 379–416. DOI: `10.1016/0885-064x(89)90017-4`.

[18] Nishtha Jatana et al. "A Survey and Comparison of Relational and Non-Relational Database". In: *International Journal of Engineering Research and Technology* 01 (06 Aug. 2020). ISSN: 2278-0181.

[19] Rafael C. Jiménez et al. "Four simple recommendations to encourage best practices in research software". In: *F1000Research* 6 (June 2017), p. 876. DOI: `10.12688/f1000research.11407.1`.

[20] Vince Knight, Henry Wilde, and Nikoleta Glynatsi. *Nikoleta-v3/blackbook: v0.0.2*. 2019. DOI: `10.5281/ZENODO.2553363`.

[21] Vincent Knight. *Chapter 10 - Infinitely Repeated Games*. [Accessed on: 15/01/2020]. 2017. URL: `https://vknight.org/Year_3_game_theory_course/Content/Chapter_10_Infinetely_Repeated_Games/`.

[22] Vincent Knight. *Chapter 2 - Normal Form Games*. Online. [Accessed on: 13/10/2019]. 2017. URL: `https://vknight.org/Year_3_game_theory_course/Content/Chapter_02-Normal_Form_Games/`.

[23] Vincent Knight. *Chapter 3 - Rationalisation*. Online. Accessed on: 23/10/2019. 2019. URL: `https://vknight.org/gt/chapters/03/`.

[24] Vincent Knight. *Chapter 5 - Support Enumeration.* Online. [Accessed on: 16/03/2020]. 2019. URL: `https://vknight.org/gt/chapters/05/`.

[25] Vincent Knight. *Chapter 9 - Finitely Repeated Games.* Online. Accessed on 06/01/2020. 2017. URL: `https://vknight.org/Year_3_game_theory_course/Content/Chapter_09_Finitely_Repeated_Games/`.

[26] Holger Krekel et al. *pytest x.y.* 2004. URL: `https://github.com/pytest-dev/pytest`.

[27] Lukasz Langa. *Black: v19.10b0.* 2019.

[28] Xiannuan Liang and Yang Xiao. "Game theory for network security". In: *IEEE Communications Surveys & Tutorials* 15.1 (2012), pp. 472–486.

[29] Nicholas Marriott. *tmux.* URL: `https://github.com/tmux/tmux`.

[30] Michael Maschler, Eilon Solan, and Shmuel Zamir. *Game Theory.* Cambridge University Press, 2013. DOI: `10.1017/CBO9780511794216`.

[31] John Nash. "Non-cooperative games". In: *Annals of mathematics* (1951), pp. 286–295.

[32] John F Nash et al. "Equilibrium points in n-person games". In: *Proceedings of the national academy of sciences* 36.1 (1950), pp. 48–49.

[33] Ameya Nayak, Anil Poriya, and Dikshay Poojary. "Type of NOSQL Databases and its Comparison with Relational Databases". In: *International Journal of Applied Information Systems* 5.4 (Mar. 2013). ISSN: 2249-0868.

[34] Noam Nisan et al. *Algorithmic Game Theory.* With a forew. by Christos H. Papadimitriou. Cambridge University Press, 2007. ISBN: 978-0-521-87282-9.

[35] Oracle. *What is a Relational Database.* [Accessed on 02/03/2020]. Oracle. 2020. URL: `https://www.oracle.com/uk/database/what-is-a-relational-database/`.

[36] ostezer and Mark Drake. *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems.* [Accessed on: 02/03/2020]. Digital Ocean. Mar. 2019. URL: `https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems`.

[37] Christos H Papadimitriou. "On the complexity of the parity argument and other inefficient proofs of existence". In: *Journal of Computer and system Sciences* 48.3 (1994), pp. 498–532.

[38] Mauricio Piacentini et al. *DB browser for SQLite.* 2015.

[39] Safraz Rampersaud, Lena Mashayekhy, and Daniel Grosu. "Computing Nash Equilibria in Bimatrix Games: GPU-Based Parallel Support Enumeration". In: *IEEE Transactions on Parallel and Distributed Systems* 25.12 (Dec. 2014), pp. 3111–3123. DOI: `10.1109/tpds.2014.2307887`.

[40] Guido van Rossum, Barry Warsaw, and Nick Coghlan. *PEP 8 – Style Guide for Python Code*. Python. July 5, 2001. URL: `https://www.python.org/dev/peps/pep-0008/`.

[41] Geir Kjetil Sandve et al. "Ten Simple Rules for Reproducible Computational Research". In: *PLoS Computational Biology* 9.10 (Oct. 2013). Ed. by Philip E. Bourne, e1003285. DOI: `10.1371/journal.pcbi.1003285`.

[42] Raimund Seidel. "The upper bound theorem for polytopes: an easy proof of its asymptotic version". In: *Computational Geometry* 5.2 (Sept. 1995), pp. 115–116. DOI: `10.1016/0925-7721(95)00013-y`.

[43] John Spacey. *9 Types of Binary File*. [Accessed on: 04/03/2020]. Simplicable. May 14, 2017. URL: `https://simplicable.com/new/binary-file`.

[44] V. Srivastava et al. "Using game theory to analyze wireless ad hoc networks". In: *IEEE Communications Surveys Tutorials* 7.4 (Apr. 2005), pp. 46–56. DOI: `10.1109/COMST.2005.1593279`.

[45] SSH.COM. *SSH tunnel*. [Accessed on: 05/03/2020]. SSH Communications Security, Inc. Oct. 11, 2016. URL: `https://www.ssh.com/ssh/tunneling`.

[46] Christopher Stover. *Quasi-Concave Function*. [Accessed on: 15/01/2020]. Wolfram Mathworld. URL: `http://mathworld.wolfram.com/Quasi-ConcaveFunction.html`.

[47] Techopedia. *Flat File*. [Accessed 04/03/2020]. Techopedia. Oct. 31, 2011. URL: `https://www.techopedia.com/definition/25956/flat-file`.

[48] The Nashpy project developers. *Nashpy: v0.0.19*. June 2019. DOI: `10.5281/zenodo.3256475`. URL: `http://dx.doi.org/10.5281/zenodo.3256475`.

[49] Steve Thomas. *What are the advantages and disadvantages of using the flat file data model?* [Accessed on: 04/03/2020]. Quora. Dec. 2018. URL: `https://www.quora.com/What-are-the-advantages-and-disadvantages-of-using-the-flat-file-data-model`.

[50] John Von Neumann, Oskar Morgenstern, and Harold William Kuhn. *Theory of games and economic behavior (commemorative edition)*. Princeton university press, 2007.

[51] James N. Webb. *Game Theory: Decisions, Interaction and Evolution*. Springer Undergraduate Mathematics Series. Springer, 2007. ISBN: 978-1-84628-423-6.

[52] Eric W. Weisstein. *Compact Space*. [Accessed on: 06/01/2020]. Wolfram Mathworld. URL: `http://mathworld.wolfram.com/CompactSpace.html`.

[53] Eric W. Weisstein. *Convex*. [Accessed on: 06/01/2020]. Wolfram MathWorld. URL: `http://mathworld.wolfram.com/Convex.html`.

[54] Greg Wilson et al. "Best Practices for Scientific Computing". In: *PLoS Biology* 12.1 (Jan. 2014). Ed. by Jonathan A. Eisen, e1001745. DOI: `10.1371/journal.pbio.1001745`.

Appendix A

APPENDIX A TITLE

Appendix B

APPENDIX B TITLE