Understanding responses to environments for the Prisoner's Dilemma: A meta analysis, multidimensional optimisation and machine learning approach

Nikoleta E. Glynatsi

Submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy.

June 2020

Abstract

This thesis investigates the optimal behaviour that Iterated Prisoner's Dilemma strategies should adopt as a response to different environments. The Iterated Prisoner's Dilemma (IPD) is a particular topic of game theory that has attracted academic attention due to its applications in the understanding of the balance between cooperation and competition in social and biological settings.

This thesis uses a variety of mathematical and computational fields such as linear algebra, research software engineering, data mining, network theory, natural language processing, data analysis, mathematical optimisation, resultant theory, markov modelling, agent based simulation, heuristics and machine learning.

The literature around the IPD has been exploring the performance of strategies in the game for years. The results of this thesis contribute to the discussion of successful performances using various novel approaches.

Initially, this thesis evaluates the performance of 195 strategies in 45,600 computer tournaments. A large portion of the 195 strategies are drawn from the known and named strategies in the IPD literature, including many previous tournament winners. The 45,600 computer tournaments include tournament variations such as tournaments with noise, probabilistic match length, and both noise and probabilistic match length. This diversity of strategies and tournament types has resulted in the largest and most diverse collection of computer tournaments in the field. The impact of features on the performance of the 195 strategies is evaluated using modern machine learning and statistical techniques. The results reinforce the idea that there are properties associated with success, these are: be nice, be provocable and generous, be a little envious, be clever, and adapt to the environment.

Secondly, this thesis explores optimal behaviours focused on a specific set of IPD strategies called memory-one, and specifically a subset of them that are considered extortion-

ate. These strategies have gained much attention in the research field and have been acclaimed for their performance against single opponents. This thesis uses mathematical modelling to explore the best responses to a collection of memory-one strategies as a multidimensional non-linear optimisation problem, and the benefits of extortionate/manipulative behaviour. The results contribute to the discussion that behaving in an extortionate way is not the optimal play in the IPD, and provide evidence that memory-one strategies suffer from their limited memory in multi agent interactions and can be out performed by longer memory strategies.

Following this, the thesis investigates best response strategies in the form of static sequences of moves. It introduces an evolutionary algorithm which can successfully identify best response sequences, and uses a list of 192 opponents to generate a large data set of best response sequences. This data set is then used to train a type of recurrent neural network called the long short-term memory network, which have not gained much attention in the literature. A number of long short-term memory networks are trained to predict the actions of the best response sequences. The trained networks are used to introduce a total of 24 new IPD strategies which were shown to successfully win standard tournaments.

From this research the following conclusions are made: there is not a single best strategy in the IPD for varying environments, however, there are properties associated with the strategies' success distinct to different environments. These properties reinforce and contradict well established results. They include being nice, opening with cooperation, being a little envious, being complex, adapting to the environment and using longer memory when possible.

Acknowledgements

First and foremost, I would like to express my greatest and utmost gratitude to my tolerant and supportive supervisor, Dr Vincent Knight, whose guidance, encouragement and jokes have been invaluable throughout my PhD. I am extremely grateful for our long research meetings, our friendly chats that concluded with me spoiling a movie/show for you, and our academic trips around the world.

I would also like to thank Dr Jonathan Gillard for his advice through parts of this work and Dr Marc Harper for his input and commentary in the narrative on the meta tournaments research.

On a more personal note, I would like to thank my family who have been a source of great inspiration and motivation throughout life. They never questioned my decisions and have supported me to the fullest. Except that time when my mother disapproved of me travelling to New York alone. I would like to express my sincere gratitude to my friends Nicki Verdeli, Kostas Soulanis and Chris Athanasiou, for their continuous support and friendship throughout my years in Cardiff. Their homemade meals, inspirational talks about not getting anxious and competitive games of UNO have been vital in the completion of this PhD.

I would also like to thank my colleagues Waleed Ali, Asyl Hawa, Geraint Palmer, Chris Seaman, Lorenzo De Biase, Henry Wilde and Emily O'Riordan for their company and endless coffee breaks. Finally, I would like to thank Mrs Joanna Emery for making my academic journey in Cardiff University possible, and the professional services staff of the School for Mathematics for their continuous help the past four years with filling forms, booking rooms, resolving IT issues and preparing coffee and biscuits.

Dissemination of Work

Publications (4 Published & 5 In preparation)

1. 2018: **Reinforcement learning produces dominant strategies for the Iterated Prisoner's Dilemma.** Marc Harper, Vincent Knight, Martin Jones, Georgios Koutsovoulos, Nikoleta E. Glynatsi, Owen Campbell - PLOS One - Preprint arXiv:1707.06307

2. 2018: **An evolutionary game theoretic model of rhino horn devaluation.** Nikoleta E. Glynatsi, Vincent Knight, Tamsin Lee. Ecological Modelling - Preprint arXiv:1712.07640

3. 2017: **Evolution reinforces cooperation with the emergence of self-recognition mechanisms: an empirical study of the Moran process for the Iterated Prisoner's dilemma.** Vincent Knight, Marc Harper, Nikoleta E. Glynatsi, Owen Campbell - PLOS ONE - Preprint arXiv:1707.06920

4. 2016: **An open framework for the reproducible study of the Iterated prisoner's dilemma.** Vincent Knight, Owen Campbell, Marc Harper et al - Journal of Open Research Software

In preparation

1. 2019: **Properties of Winning Iterated Prisoner's Dilemma Strategies.** Nikoleta E. Glynatsi, Vincent A. Knight and Marc Harper - In preparation to be submitted - Preprint arXiv:2001.05911

2. 2019: **A bibliometric study of research topics, collaboration and influence in the field of the Iterated Prisoner's Dilemma.** Nikoleta E. Glynatsi and Vincent A. Knight - Submitted to Palgrave Communications - Preprint arXiv:1911.06128

3. 2019: **Game Theory and Python: An educational tutorial to game theory and repeated games using Python.** Nikoleta E. Glynatsi and Vincent A. Knight - Submitted to the Journal of Open Source Education - Available on GitHub Nikoleta-v3/Game-Theory-and-Python

4. 2019: **A theory of mind: Best responses to memory-one strategies. The limitations of extortion and restricted memory.** Nikoleta E. Glynatsi and Vincent A. Knight - Under review at Scientific Reports Nature - Preprint arXiv:1911.12112

5. 2019: **Recognising and evaluating the effectiveness of extortion in the Iterated Prisoner's Dilemma.** Vincent Knight, Marc Harper, Nikoleta E. Glynatsi, Jonathan Gillard - In preparation to be submitted - Preprint arXiv:1904.00973

Talks & Posters

Invited Talks (Keynotes)

- How does a smile make a difference?, PyCon UK, Cardiff, 2018.
- The Fallacy of Meritocracy, PyCon Balkan, Belgrade, 2019.

Others

- Accessing open research literature with Python - PyCon Namibia, Namibia 2017.
- Writing tests for research software - PyCon Namibia, Namibia 2017.
- Optimisation of short memory strategies in the Iterated Prisoners Dilemma - Wales Mathematics Colloquium, Gregynog Hall 2017.
- PIP INSTALL AXELROD (**poster**) - Euroscipy, Erlangen Germany 2017.
- Arcas: Using Python to access open research literature - Euroscipy, Erlangen Germany 2017.
- A trip to earth science with python as a companion - PyConUK, Cardiff 2017.
- The power of memory (**poster**) - SIAM UKIE Annual Meeting, Southampton 2018.
- Rhinos with a bit of Python - PyConNA, Namibia 2018.

- Memory size in the Prisoners Dilemma - Wales Mathematics Colloquium, Gregynog Hall 2018.

- Memory size in the Prisoners Dilemma - SIAM UKIE National Student Chapter, Bath University 2018.

- Stability of defection, optimisation of strategies and testing for extortion in the Prisoner's Dilemma (**poster**) - STEM for Britain, London 2019.

- Stability of defection, optimisation of strategies and testing for extortion in the Prisoner's Dilemma - 18th International Conference on Social Behaviour, Sedona, Arizona 2019.

- An introduction to Time Series - Joint workshop between CUBRIC & Mathematics Departments, Cardiff 2019.

Software development

- Arcas, an open source package designed to help users collect academic articles' metadata from various prominent journals and pre print serves. **Contribution:** Main developer.

- Axelrod-Python library, an open source framework decided to the study of the Iterated Prisoner's Dilemma. **Contributions:** Implementation of spatial tournaments functionality, implementation/addition of strategies (from the literature) to the library, reviewing of code contributed by other contributors.

- SymPy, a Python library for symbolic mathematics. **Contributions:** Implementation of Dixon's and Macaulay's resultants which were developed for my manuscript "A theory of mind: Best responses to memory-one strategies. The limitations of extortion and restricted memory".

- Pandas, an open source library providing high-performance, easy-to-use data structures and data analysis tools. **Contribution:** Fix bug which converted missing values to strings.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Game theory is a field that makes use of mathematical tools and logic to model and analyse situations of conflict, cooperation, and competition. One of the most well-known examples of a strategic game is the Prisoner's Dilemma, consisting of two players which can either cooperative or defect. A more realistic version of the game is that of the Iterated Prisoner's Dilemma where the two players play more than once in succession. The players remember the previous actions taken and change their strategy accordingly.

The world is surrounded by situations of conflict, and understanding the emergent outcome of interactions between two players can have a significant impact in economical and political sciences.

In 1984 the "The evolution of Cooperation" was published by Robert Axelrod introducing the usage of the Iterated Prisoner's Dilemma and computer modelling in studying situations of conflict. Axelrod explored the optimal behaviour of players in round robin tournaments using computer strategies. Many tournaments have followed Axelrod's, and today the literature and various codebases contain hundreds of strategies. The aim of all these strategies has been to capture the best behaviour when playing the game.

This thesis aims to reinforce the understanding of optimal behaviour in the Prisoner's Dilemma for a variety of environments. It summarises, evaluates and builds upon previous literature. It does not only contribute to the discussion of dominant behaviour but also provides new mathematical results for the continued understanding of the questions raised throughout.

This introductory Chapter is set as follows:

- section 1.1 introduces the Prisoner's Dilemma.
- section 1.2 covers a brief literature review.
- section 1.3 formalises the research questions and sets out the structure of the thesis.
- section 1.4 presents the software development techniques used throughout the thesis.

1.1 Prisoner's Dilemma

Game theory was formalised in 1944 [215] and is the study of interactions as *games*. A game is a model of interacting decision makers referred to as *players*. Each player has a set of possible *actions*, and the game captures the interactions of the players' actions by allowing each player's *payoffs* to be dependant on the actions of all players. More precisely, as given in [232], the formal definition of a game is as follows:

Definition 1.1.1. A **game** consists of

- a set of players,
- for each player, a set of actions and
- for each player, payoff functions mapping the set of all actions to a numerical value.

One of the most well known games is the Prisoner's Dilemma (PD) originally described in [82]. The PD is a two player non-cooperative game which illustrates aspects of political philosophy and morality; how selfishness will lead to an 'inefficiency' of the outcome even though selflessness can be evolutionarily advantageous.

More specifically, in the PD each player has two actions, to either be selfless and cooperate, denoted as (C), or to be selfish and defect, denoted as (D). Each decision is made simultaneously and independently. The players' payoffs are generally represented by Equation (1.1). Both players receive a reward for mutual cooperation, R, and a payoff P for mutual defection. A player that defects while the other cooperates

receives a payoff of T, whereas the cooperator receives S.

$$S_p = \begin{pmatrix} R & S \\ T & P \end{pmatrix} \quad S_q = \begin{pmatrix} R & T \\ S & P \end{pmatrix} \tag{1.1}$$

It is assumed that two cooperating players do better than two defecting ones, and thus, the payoff of two cooperating players is larger than the payoff of two defecting players; $R > P$. A player, however, has the temptation to deviate, as that player will receive a higher payoff T than that of mutual cooperation R whilst the cooperator's payoff S is smaller than P. In consequence, the payoffs are constrained by Equation (1.2).

$$T > R > P > S \tag{1.2}$$

A second constraint which ensures that a social dilemma arises, is that the sum of the utilities to both players is best when they both cooperate, Equation (1.3).

$$2R > T + S \tag{1.3}$$

An equivalent representation of the PD is the *donation game*. In the donation game each player can cooperate by providing a benefit b to the other player at a cost c with $0 < c < d$. Thus, the players' payoffs for the donation game are as given by Equation (1.4).

$$S_p = \begin{pmatrix} b-c & c \\ b & 0 \end{pmatrix} \quad S_q = \begin{pmatrix} b-c & b \\ c & 0 \end{pmatrix} \tag{1.4}$$

This thesis studies the PD as given by Equation (1.1). There are numerical experiments presented in the following Chapters. These have been carried out using the payoff values of $R = 3, P = 1, T = 5$ and $S = 0$, which are the values most commonly used in the literature [13, 38, 44, 46, 74, 84, 121, 122, 159, 163, 177, 250, 274].

In non-cooperative games the players interact in order to achieve their best possible outcome. A *best response strategy* is a strategy that maximises the utility of a player

given a known strategy of the other player. A solution concept commonly used in game theory is the Nash equilibrium [214] which is a pair of best response strategies at which neither of the players has a reason to deviate.

In the PD due to constraint (1.2) it never benefits a player to cooperate. A player that cooperates receives either a payoff of R or S depending the action of the other player, whereas if a player defects they receive either T or P, and $T > R$ and $P > S$. Once both players defect neither have a reason to change their decision. Thus, in the PD mutual defection is a Nash equilibrium and defection is the best response strategy.

The game can be studied in a manner where prior outcomes matter. The repeated form of the game is called the Iterated Prisoner's Dilemma (IPD) and it differs from the original concept of a PD because participants can learn about the behavioural tendencies of their opponent. In the IPD defecting is no longer necessarily the dominant action, and identifying a best response is not always trivial.

1.2 Brief history of the IPD

In the 1980's Robert Axelrod studied the best way of behaving in the IPD by running a series of computer tournaments with two collections of strategies [38]. These strategies were written/submitted by researchers. Axelrod performed an evolutionary tournament [34] and two round robin tournaments [32, 33]. The strategy that took over the population and won both tournaments was the strategy Tit For Tat. Axelrod's results demonstrated the robustness of the strategy in those environments and subsequently the robustness of reciprocal behaviour. These results, however, did not consider the success of the strategy in other environments. This became more evident as further competitions and mathematical formulations introduced new dominant strategies. A brief summary of selected works and their dominant strategies are given by Table 1.1.

More details on these works will be presented in Chapter 2, and following Chapters 2 and 3 it will become evident that the literature on the IPD is rich, and new strategies and competitions are being published every year. The question, however, still remains the same: what is the best way to play the game?

Year	Reference	Environment	Dominating Strategies
1980	[32]	Round robin tournament with 13 participants	Tit For Tat
1980	[33]	Round robin tournament with a probabilistic ending and 13 participants	Tit For Tat
1984	[34]	Ecological tournament with 64 participants	Tit For Tat
1987	[43]	Round robin & ecological tournament with 12 participants	Gradual
1991	[46]	Round robin tournament with noise and 13 participants	Nice and Forgiving
2005	[159]	Varied with 223 participants	Varying
2012	[274]	Round robin tournament with 13	Generous zero-determinants
2016	[164]	Round robin tournament with 130 participants	Heuristically trained strategies
2017	[122]	Round robin tournament with 200 participants	Heuristically trained strategies

Table 1.1: An overview of published works that introduced dominating IPD strategies in their respective environments. These strategies were either explicitly calculated, intelligently designed, or were developed through training methods.

1.3 Research questions & Thesis structure

This thesis contains eight Chapters, which together attempt to answer the research question:

> What is the optimal behaviour an Iterated Prisoner's Dilemma strategy should adopt as a response to different environments?

Initially, Chapter 2 provides a condensed literature review which summarises the already established results of the literature. Chapter 2 separates the reviewed manuscripts under different research topics identified manually. To complement the manual separation of articles under research topics, Chapter 3 automatically partitions 2,422 IPD articles using data mining, machine learning and natural language processing. The data set of 2,422 articles' metadata has been collected using a bespoke research software tool, which was written for this work but has since been used by others. The data set is further analysed using network theoretic approaches to explore the behaviour of authors.

There are four Chapters to the thesis which explore optimal behaviour using original approaches. Namely, Chapter 4 analyses a set of 45,600 computer tournaments of distinct types and evaluates 195 strategies' performance. Chapter 5 explores best response strategies to environments of memory-one opponents and Chapter 6 explores best response strategies in the form of static sequences of moves to a collection of opponents. Finally Chapter 7, uses the data set of best response sequences generated in Chapter 6 to train an IPD strategy using a recurrent neural network.

The six Chapters of the thesis and their role is illustrated in Figure 1.1. An arrow

between one Chapter and another implies that the work described in one serves as motivation for the other.

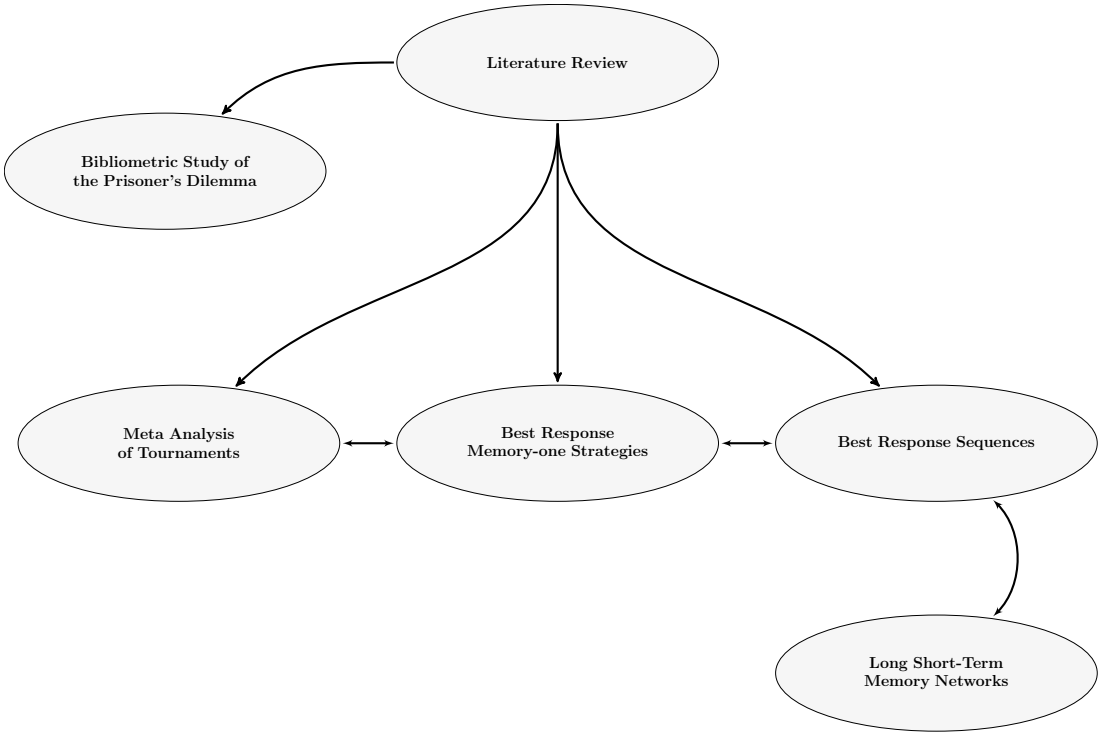


Figure 1.1: Structure of this thesis.

A summary of each Chapter is given below:

- Chapter 1 has contextualised the three main research questions of this thesis. Background of game theory and the PD has been given, and the structure of the remainder of the thesis has been outlined.

- Chapter 2 provides a literature review for the PD and a manual classification of the reviewed papers under research topics. The manually identified research topics include evolutionary dynamics, intelligently designed strategies and structured strategies which have undergone training.

- Chapter 3 presents a bibliometric analysis of 2,422 IPD articles. It uses natural language processing to identify five research topics, and a graph theoretic approach to quantify the collaborativeness of the field. The five identified topics are human subject research, biological studies, strategies, evolutionary dynamics on networks and modelling problems as a PD.

- Chapter 4 generates and analyses a set of 45,600 computer tournaments. It evaluates 195 strategies, many of which are well known strategies from the literature.

It presents the top performing strategies and analyses their salient features. The results show that there is not yet a single strategy that performs well in diverse IPD scenarios, nevertheless there are several properties that heavily influence the best performing strategies. These are: be nice, be provocable and generous, be a little envious, be clever, and adapt to the environment.

- Chapter 5 explores best responses to a collection of memory-one strategies as a multidimensional non-linear optimisation problem. It presents a closed form algebraic expression for the utility of a memory-one strategy against a given set of opponents, a compact method of identifying its best response to that given set of opponents whilst having a theory of mind, and it introduces a well designed framework that allows the comparison of an optimal memory-one strategy and a more complex strategy which has a larger memory. The results add to the literature that has shown that extortionate play is not always optimal by showing that optimal play is often not extortionate.

- Chapter 6 explores the problem of IPD best responses in the form of sequences. It heuristically identifies the best response sequence against 192 strategies, and generates a data set of 750 best response sequences of 205 turns. The Chapter mainly serves as a foundation for Chapter 7 but does present a novel heuristic and a study of its performance.

- Chapter 7 uses the data set of best response strategies obtained from Chapter 6 to train a type of recurrent neural network to predict best response sequences. The recurrent neural network used is the long short-term memory network (LSTM) which has gained a lot of attention in the machine learning literature but not in the IPD literature. A total of 8 were trained which were then used to introduce 24 distinct IPD strategies. It is demonstrated that a set of these strategies can win standard tournaments and the best LSTM performers on average rank at the top 25% of any standard tournament.

- Chapter 8 summarises the work of the previous chapters, and indicates possible directions of future work, identifying further research questions that have arisen.

Chapters 4-7 explore optimal behaviour in the IPD. The disparity between the approaches is their depth, as illustrated in Figure 1.2. Chapter 4 explores optimal behaviour by analysing a data set of tournaments and evaluating the performance of pre-designed strategies. The exact opposite is done in Chapter 5 whereas for a given

set of two memory-one opponents a best response strategy is calculated explicitly. Similarly, a best responses sequence against a given opponent is calculated in Chapter 6, but this is done using a heuristic method. Finally, Chapter 7 uses a machine learning algorithm to generate a optimal behaviour based on recurrent networks without any manual input.

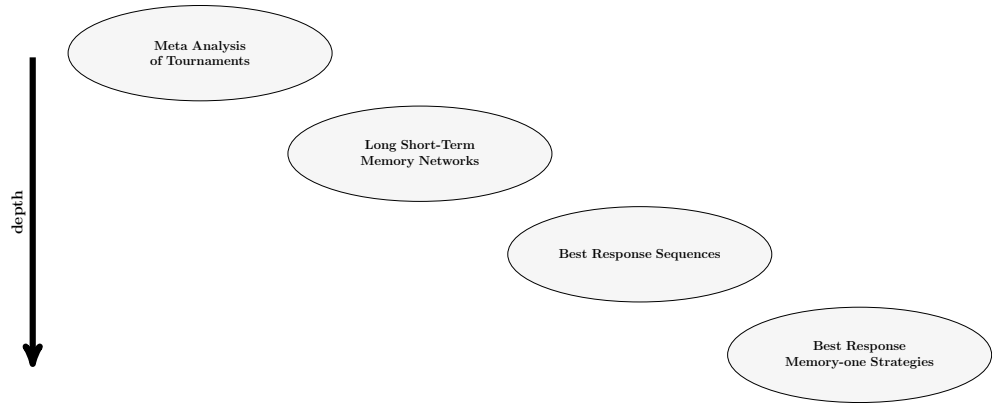


Figure 1.2: The depth of exploration whilst reporting on research question 1.

Most of Chapters of the thesis make use of parameters. There are instances that the same symbol is used at different Chapters to denote different parameters with different meanings. The parameters of each Chapter alongside a brief exploration per Chapter is given in the Appendix D.

1.4 Software development & Best practices

A survey conducted by the Software Sustainability Institute at 15 Russell Group Universities showed that 92% of the researchers questioned use software intensively in their work, and 70% said that "It would not be practical to conduct my work without software" [128]. Similarly, the research of this thesis heavily relies on software. As with all research there is an obligation to ensuring the correctness and reproducibility of the results and the software decisions throughout this thesis have been driven by these requirements.

For the research of each Chapter (excluding Chapters 2 and 8) source code and analysis code have been developed. All code is written in the an open source language Python, has been made public via GitHub and has one of the most flexible and permissive licences, the MIT licence. Essentially, the code developed for the thesis is available for inspection, testing, and modification which enables and encourages greater understanding of the underlying methodology, increases model confidence, and provides an

extendable framework which can be used by others.

Two themes arise as vital in research software development: reproducibility and sustainability. To reassure the reproducibility and sustainability of the software, and subsequently the research described in the thesis, several methods of *best practice* [11, 48, 69, 135] were considered and implemented during development. Namely:

- Version control
- Virtual environments
- Automated testing
- Documentation

These will be discussed in the following subsections.

1.4.1 Version control

Version control, is a system which records all files that make up a project (down to the line) over time, tracking their development. It also provides the ability to recall previous versions of files. This type of system is essential for ensuring reproducibility of scientific research [258, 294].

A good version control system has the following features as stated in [254]:

- Backup and restore: Files can be saved as they are edited and have the facility to jump to a previous version.
- Synchronisation: Source code files can be shared and users can update their codebase with the latest version.
- Undo changes: Changes made to the code can be undone by going back to a version that was committed in the past.
- Track changes: Messages are attacked to file changes in order to track the how and why the code evolved over time.
- Track ownership: File changes are tagged with the user's name who made the changes.
- Sandboxing: The ability to make temporary changes in an isolated area, called a sandbox, to test and try out code before it is checked in.

- Branching and merging: This is akin to a larger sandbox. Users can branch a copy of the code into a separate area and modify it in isolation (tracking changes separately). Later, the work can be merged back into the original codebase.

There are a number of popular tools for version control, these include Git [93], Subversion [17], and Mercurial [202]. The version control system chosen to carry out the software development here is Git.

There are several services that host Git servers online and allow users to work with Git publicly. These services are essential for reproducibility as they make not only the source code for the computer programmes available online but also the history of its development. Such services are GitHub [142], SourceForge [200], GitLab [94], and BitBucket [30]. GitHub is the chosen service for the thesis which integrates well with Git.

An important feature of GitHub is that it fosters collaboration between users. It is a social service which allows users to comment and raise issues on each other's repositories. Moreover, it encourages collaboration and code contributions by other users through pull request features. An example of a pull request on a GitHub repository is given by Figure 1.3. In Figure 1.3 it is shown how changes are tagged with a user and a short message describing the alternations made to the codebase.

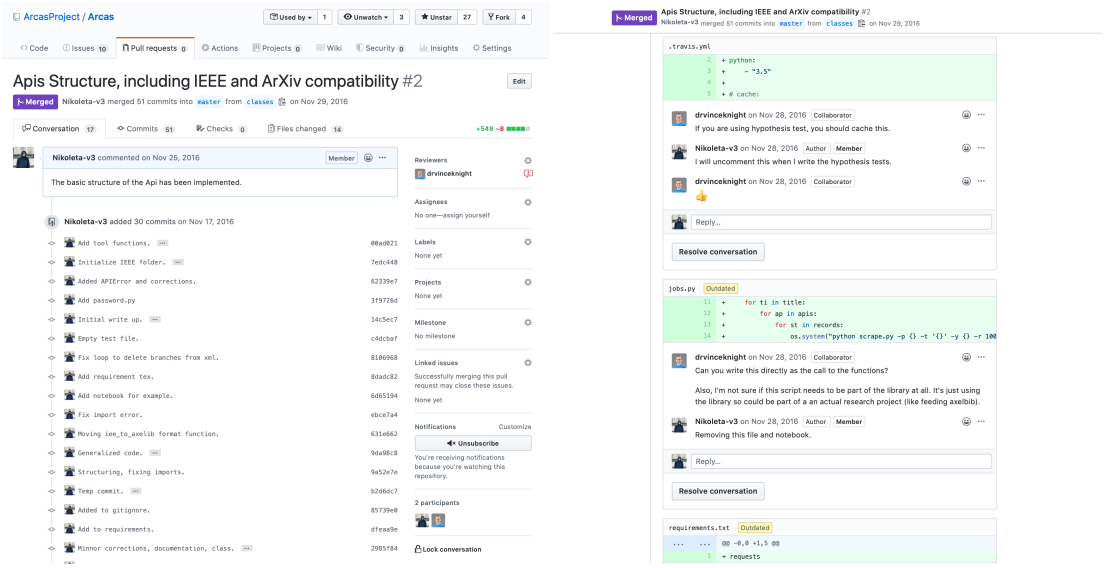


Figure 1.3: An example of a pull request on GitHub.

The code for Chapters' 3-7 is hosted on individual GitHub repositories, Table 1.2. The source code for each repository has been packaged and has been archived on Zenodo [58]. Zenodo is a platform where code, data and other project elements can be permanently

archived. Zenodo does this by assigning projects a Digital Object Identifier (DOI) which makes the work citable.

1.4.2 Virtual environments

The source code of each repository has Python libraries as dependencies. Though several of these projects use the same libraries, the versions of these libraries can differ. Tracking software dependencies is of paramount importance in order to ensure the reproducibility of computer code.

There are several tools for keeping dependencies required by different projects separated. The tool used here are Python *virtual environments*. More specifically, the Anaconda virtual environments which integrate easily with the programming language Python. Anaconda [141] is a free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. Package versions are managed by the package management system conda.

The Anaconda distribution manager: conda allows users to create, export, list, remove, and update environments that have different versions of Python and/or packages installed in them. Switching or moving between environments is called activating the environment. An environment can be shared and kept under version control as a file. An example of such as file is given by Figure 1.4.

Each Chapter's repository includes an environment file detailing the dependencies of the source code and their versions.

1.4.3 Automated testing

Testing code is of considerable importance in order to ensure the robustness, correctness and sustainability of the computer code. The standard method of testing code is through *automated testing* using test suits that run parts of the code and assert whether they are behaving as expected.

Two types of tests are described in [235], *functional tests* that assert the code's functionality, and *unit tests* that help ensure the code is clean and free of bugs.

Functional tests aim to test how the whole application functions from the perspective of the outside user. They feed in basic input and test whether the end product/final

```yaml
name: opt-mo
channels:
  - defaults
dependencies:
  - python=3.6.7
  - numpy=1.15.4
  - pandas=0.23.4
  - pip:
    - attrs==19.1.0
    - axelrod==4.4.0
    - black==18.9b0
    - sympy==1.2.0
    - scikit-optimize==0.5.2
    - jupyter==1.0.0
    - jupyter-console==5.2.0
    - ipython==6.4.0
    - pytest==4.0.1
    - pytest-cov==2.7.1
    - sqlalchemy==1.2.17
    - fsspec==0.3.3
```

Figure 1.4: An example of an environment file. The name of the specific environment is called `opt-mo` and it corresponds to the environment associated with Chapter 5.

behaviour is as expected. Unit tests assert that small chunks of code behave as expected, and the test the application from the point of the programmer. Unit tests are isolated from the rest of the code and are modular. There are two types of unit tests: *pure* and *integrated* tests.

Pure unit tests are written to test only one function or method. Thus, if a pure unit test was to fail then it should be due to problems with the specific part of the code it is testing only, and not any other bit of code. Pure unit tests are fast and readable, however, they do not test how well functions and methods integrate with one another. This is tested by writing unit tests that rely on other parts of the code that are not explicitly being tested. This type of unit test is called integrated tests.

Automated tests, which include functional and unit tests, have been implemented for the repositories associated with the thesis. This was done using the Python library `pytest` which makes it easy to write automated tests in a few lines and to check for code coverage. Coverage is a measure used to describe how much of the code is executed (covered) by the testing suite. Coverage is tested using a plugin to `pytest`, the `pytest-cov`.

To regularly test code which aims to be merged back into the original codebase continuous integration (CI) systems are used. CIs perform the tests suite and the cover-

age checks every time a new version of the codebase is made available ("pushed") on GitHub. The benefits of using a CI are identifying bugs quickly, reducing problems when merging in contributions from collaborators, and adding transparency to the development process. There are two CIs that have been used in the repositories listed in Table 1.2. These are Travis [65] and GitHub Actions [143].

1.4.4 Documentation

Software *documentation* is written text or illustration that accompanies computer software or is embedded in the source code. The documentation either explains how the software operates or how to use it.

Each repository associated with the thesis includes a detailed README file. These contain installation instructions for the corresponding packaged source code and demonstrate how to run the associated test suite. The source code for each repository has been written in a modular way and meaningful names have been given to all variables, functions, methods and classes. Each function, method and class includes a docstring. A docstring is a series of sentences used to document a specific segment of code.

The repositories also include a series of Jupyter Notebooks [154] that are used to carry out the analysis of each Chapter, and serve as demonstration of the source code's usage.

1.4.5 Summary of software written

As previously stated the codebases for Chapters 3-7 have been written following best practices, have been packaged, are available on GitHub and have been archived on Zenodo. These practices have been followed to ensure the correctness, reproducibility and sustainability of the source code and research described throughout the thesis.

To ensure the reproducibility of the work the data sets used in several of the following Chapters have also been archived and are available online. The details for the source code and data sets for each Chapter are summarised in Table 1.2.

Throughout the thesis, parts of the source code and examples of the code's usage are going to be presented in the corresponding Chapters. Two types of code snippets are used in this thesis to present code. Code snippets that demonstrate the source code of a specific piece of software, Figure 1.5, and code snippets that demonstrate the usage, Figure 1.6. These can be distinguish by the three arrows, >>>, which are only

	GitHub url	Source code archive	Data archive
Chapter 3	https://github.com/Nikoleta-v3/bibliometric-study-of-the-prisoners-dilemma	[107]	[98, 99, 100]
Chapter 4	https://github.com/Nikoleta-v3/meta-analysis-of-prisoners-dilemma-tournaments	[96]	[97, 102]
Chapter 5	https://github.com/Nikoleta-v3/Memory-size-in-the-prisoners-dilemma	[108]	[101]
Chapter 6 & 7	https://github.com/Nikoleta-v3/Training-IPD-strategies-with-RNN	[95]	[103, 105, 106]

Table 1.2: A summary of the GitHub repositories, source code and data archives associated with the thesis.

found in the usage code snippets. The three arrows are followed by a command. It demonstrates that the command is executed in a Python interpreter, and the result of executing the command is the one in the following lines without the arrows. The code snippets can also be distinguish by their background color. The usage snippets have a lighter background.

```
import axelrod as axl

def simulate_match_utility(player, opponent, turns=500, repetitions=200):
    """
    Returns the simulated utility of a memory one player against a single opponent.
    """
    total = 0
    players = [axl.MemoryOnePlayer(vector) for vector in [player, opponent]]
    for rep in range(repetitions):
        match = axl.Match(players=players, turns=turns)
        _ = match.play()

        total += match.final_score_per_turn()[0]

    return total / repetitions
```

Figure 1.5: Example of a function implemented withing the package `opt_mo` which is the package that has been developed to carry out the research of Chapter 5.

```
>>> import opt_mo
>>> opt_mo.utility.simulate_match_utility([1, 0, 1, 0], [1, 1, 1, 1])
3.0
```

Figure 1.6: An example of using the function `simulate_match_utility` given by Figure 1.5.

The results of this thesis heavily rely not only on the projects of Table 1.2 but also on the open source package Axelrod-Python library (APL). APL [7] is an open source project for simulating rounds of the IPD which contains a large collections of strategies. APL has several capabilities which include performing different types of tournaments. Its documentation is found at `http://axelrod.readthedocs.io/`. The specific version

of APL used in each Chapter will be mentioned at the start of each Chapter.

This thesis itself is hosted on a GitHub repository at `https://github.com/Nikoleta-v3/Thesis`. It is written in the document preparation system LaTeX, and automated tests have been setup to test that the document compiles, spelling is correct, every time an updated version of the document is pushed to GitHub. The usage code examples of this thesis are also automatically tested. Each command beginning with the symbol `>>>` is executed each time the document is pushed to GitHub. The test executes the commands and checks that the outcome is the same as the one following the command in the code snippets.

1.5 Chapter summary

This Chapter has introduced the IPD which is the strategic game used in this thesis. It has presented a review of the Axelrod's tournaments in the 1980s, and presented a list of tournaments that have been performed ever since.

The research questions of this thesis and how each Chapter contributes to these questions have been outlined. The research of this thesis heavily reliefs on software. The software includes already established packages and packages that have been developed specifically for this thesis. These have been developed following best practices. A number of best practices were introduced in section 1.4.

The software packages and the data sets used in the following Chapters have been archived and made available online. This reassures that all the results presented in the following Chapters are reproducible. The developed packages as well as this thesis are being hosted on GitHub repositories and are being tested using automated tests.

Chapter 2

A literature review of the Prisoner's Dilemma.

2.1 Introduction

Chapter 1 introduced the PD as the main game theoretic model that will be used throughout this thesis, and presented a brief literature review of the research this thesis is building upon. This Chapter provides a more detailed literature on the PD. The aim of this Chapter is to provide a concrete summary of the existing literature and to identify research topics in the field of the PD. This is achieved by partitioning the literature in five different sections each reviewing a different aspect of research. The Chapter is structured as follows:

- section 2.2 presents the origin of the PD and reviews the early publications in the field and the use of human subject research.
- section 2.3 presents the pioneering computer tournaments of Axelrod and reviews IPD strategies of intelligent design.
- section 2.4 discusses the emergence, or not, of cooperative behaviour in evolutionary dynamics.
- section 2.5 defines structured strategies in the IPD, the notion of training and discusses related papers.
- section 2.6 reports on educational and research software used for simulating the PD game.

2.2 Origins of the Prisoner's Dilemma

The origin of the PD goes back to the 1950s in early experiments conducted at RAND [82] to test the applicability of games described in [215]. The game received its name later the same year. According to [282], Albert W. Tucker (the PhD supervisor of John Nash [214]), in an attempt to deliver the game with a story during a talk described the players as prisoners and the game has been known as the Prisoner's Dilemma ever since.

The early research on the IPD was limited. The only source of experimental results was through human subject research where pairs of participants simulated plays of the game, and human subject research had disadvantages. Humans could behave randomly and in several experiments both the size and the background of the individuals were different, thus comparing results of two or more studies became difficult.

The main aim of these early research experiments was to understand how conditions such as the gender of the participants [80, 184, 187], the physical distance between the participants [263], the effect of their opening moves [281] and even how the experimenter, by varying the tone of their voice and facial expressions [90], could influence the outcomes and subsequently the emergence of cooperation. An early figure that sought out to understand several of these conditions was the mathematical psychologist Anatol Rapoport. The results of his work are summarised in [242].

Rapoport was also interested in conceptualising strategies that could promote international cooperation. Decades later he would submit the winning strategy (Tit For Tat) of the first computer tournament, run by Axelrod. These tournaments and several strategies that were designed by researchers, such as Rapoport, are introduced in the following section.

2.3 Axelrod's tournaments and intelligently designed strategies

As discussed in section 2.2, before 1980 a great deal of research was done in the field, however, as described in [36], the political scientist Axelrod believed that there was no clear answer to the question of how to avoid conflict, or even how an individual should play the game. Combining his interest in artificial intelligence and political science Axelrod created a framework for exploring these questions using computer tournaments

and made the study of cooperation of critical interest. As described in [243], "Axelrod's new approach has been extremely successful and immensely influential in casting light on the conflict between an individual and the collective rationality reflected in the choices of a population whose members are unknown and its size unspecified, thereby opening a new avenue of research".

The first reported computer tournament took place in 1980 [32]. Axelrod asked researchers to design a strategy with the purpose of winning an IPD tournament. A total of 13 strategies were submitted, written in the programming languages Fortran or Basic. Each competed in a 200 turn match against all 12 opponents, itself and a player that played randomly (called Random). This type of tournament is referred to as a *round robin*. The tournament was repeated 5 times to get a more stable estimate of the scores for each pair of play. Each participant knew the exact number of turns and had access to the full history of each match. Furthermore, Axelrod performed a preliminary tournament and the results were known to the participants. This preliminary tournament is mentioned in [32] but no details were given.

The winner of the tournament was determined by the total average score and not by the number of matches won. The strategy that was announced the winner was the strategy submitted by Rapoport, Tit For Tat. The success of Tit For Tat came as a surprise. It was not only the simplest submitted strategy, it would always cooperates on the first round and then mimic the opponent's previous move, but it had also won the tournament even though it could never beat any player it was interacting with.

In order to further test the results Axelrod performed a second tournament in 1980 [33]. The second tournament received much more attention and had a total of 62 entries. The participants knew the results of the previous tournament and the rules were similar with only a few alterations. The tournament was repeated 5 times and the length of each match was not known to the participants. Axelrod intended to use a fixed probability (referred to as 'shadow of the future' [37]) of the game ending on the next move. However, 5 different number of turns were selected for each match 63, 77, 151, 308 and 401, such that the average length would be around 200 turns.

Nine of the original participants competed again in the second tournament. Two strategies that remained the same were Tit For Tat and Grudger. Grudger is a strategy that will cooperate as long as the opponent does not defect, submitted by James W. Friedman. The name Grudger was given to the strategy in [181]. The

strategy goes by many names in the literature such as **Spite** [43], **GrimTrigger** [41] and **Grim** [286]. New entries in the second tournament included **Tit for Two Tats** submitted by John Maynard Smith and **KPavlovC**. **KPavlovC**, is also known as **Simpleton** [242] or just **Pavlov** [220]. The strategy is based on the fundamental behavioural mechanism win-stay, lose-shift. **Pavlov** is heavily studied in the literature and similarly to **Tit For Tat** had many variants trying to build upon its success, for example **PavlovD** and **Adaptive Pavlov** [177].

Despite the larger size of the second tournament none of the new entries managed to outperform the simpler designed strategy. The winner was once again **Tit For Tat**. Axelrod concluded that the strategy's robustness was due to four properties, which he adapted into four suggestions on doing well in an IPD:

- Do not be envious by striving for a payoff larger than the opponent's payoff.
- Be "nice"; Do not be the first to defect.
- Reciprocate both cooperation and defection; Be provocable to retaliation and to apologies.
- Do not be too clever by scheming to exploit the opponent.

Being provocable to apologies refers to a strategy's ability to go from a DC to C, which is also referred to as forgiveness. The only way **Tit For Tat** would end up in DC is if it had received a defection and then retaliated. Subsequently, **Tit For Tat** would forgive an opponent that apologises (in a DC round) by returning to cooperation.

The success of **Tit For Tat**, however, was not unquestionable. Several papers showed that stochastic uncertainties severely undercut the effectiveness of reciprocating strategies and such stochastic uncertainties have to be expected in real life situations [204]. For example, in [210] it is proven that in an environment where *noise* (a probability that a player's move will be flipped) is introduced two strategies playing **Tit For Tat** receive the same average payoff as two **Random** players. Hammerstein, pointed out that if by mistake, one of two **Tit For Tat** players makes a wrong move, this locks the two opponents into a hopeless sequence of alternating defections and cooperations [262]. The poor performance of the strategy in noisy environments was also demonstrated in tournaments. In [46, 74] round robin tournaments with noise were performed, and **Tit For Tat** did not win. The authors concluded that to overcome the noise more generous strategies than **Tit For Tat** were needed. They introduced the strategies **Nice and For-**

giving and **Omega Tit For Tat** respectively. A second type of stochastic uncertainty is misperception, where a player's action is made correctly but it is recorded incorrectly by the opponent. In [295], a strategy called **Contrite Tit For Tat** was introduced that was more successful than **Tit For Tat** in such environments. The difference between the strategies was that **Contrite Tit For Tat** was not so fast to retaliate against a defection.

Several works extended the reciprocity based approach which has led to new strategies. For example **Gradual** [43] which was constructed to have the same qualities as those of **Tit For Tat** except one, **Gradual** had a memory of the game since the beginning of it. **Gradual** recorded the number of defections by the opponent and punished them with a growing number of defections. It would then enter a calming state in which it would cooperates for two rounds. In a tournament of 12 strategies, including both **Tit For Tat** and **Pavlov**, **Gradual** managed to outperformed them all. A strategy with the same intuition as **Gradual** is **Adaptive Tit For Tat** [284]. **Adaptive Tit For Tat** does not keep a permanent count of past defections, it maintains a continually updated estimate of the opponent's behaviour, and uses this estimate to condition its future actions. In the exact same tournament as in [43] with now 13 strategies **Adaptive Tit For Tat** ranked first.

Another extension of strategies was that of teams of strategies [72, 73, 250] that collude to increase one member's score. In 2004 Graham Kendall led the Anniversary Iterated Prisoner's Dilemma Tournament with a total of 223 entries. In this tournament participants were allowed to submit multiple strategies. A team from the University of Southampton submitted a total of 60 strategies [250]. All these were strategies that had been programmed with a recognition mechanism by default. Once the strategies recognised one another, one would act as leader and the other as a follower. The follower plays as a **Cooperator**, cooperates unconditionally and the leader would play as a **Defector** gaining the highest achievable score. The followers would defect unconditionally against other strategies to lower their score and help the leader. The result was that Southampton had the top three performers. Nick Jennings, who was part of the team, said that "We developed ways of looking at the Prisoner's Dilemma in a more realistic environment and we devised a way for computer agents to recognise and collude with one another despite the noise. Our solution beats the standard **Tit For Tat** strategy" [226].

2.3.1 Memory-one strategies

A set of strategies that have received a lot of attention in the literature are *memory-one* strategies. In [221], Nowak and Sigmund proposed a structure for studying simple strategies that remembered only the previous turn, and moreover, only recorded the move of the opponent. These are called *reactive* strategies and they can be represented by using three parameters (y, p_1, p_2), where y is the probability to cooperate in the first move, and p_1 and p_2 are the conditional probabilities to cooperate given that the opponent's last move was a cooperation or a defection. For example Tit For Tat is a reactive strategy and it can be written as $(1, 1, 0)$. Another reactive strategy well known in the literature is Generous Tit For Tat [223] $(1, 1, \frac{1}{3})$.

In [222], Nowak and Sigmund extended their work to include strategies which consider the entire history of the previous turn to make a decision. These are called memory-one strategies. If only a single turn of the game is taken into account and depending on the simultaneous moves of the two players there are only four possible states that the players could be in. These are:

- Both players cooperated, denoted as CC.
- First player cooperated while the second one defected, denoted as CD.
- First player defected while the second one cooperated, denoted as DC.
- Both players defected, denoted as DD.

A memory-one strategy can be denoted by the probabilities of cooperating after each state and the probability of cooperating in the first round, (y, p_1, p_2, p_3, p_4). For example Pavlov's memory-one representation is $(1, 1, 0, 0, 1)$. Though reactive and memory-one strategies have to specify their move in the first round, the opening move is a transient effect and has no affect on the game in long run [266]. Consequently, reactive strategies can be described as an element $\in R^2$ and memory-one strategies as an element $\in R^4$.

Memory-one strategies made an impact when a specific subset of memory-one strategies were introduced called *zero-determinant* strategies (ZDs) [237]. The American Mathematical Society's news section [133] stated that "the world of game theory is currently on fire" and in [274] it was stated that "Press and Dyson have fundamentally changed the viewpoint on the Prisoner's Dilemma". ZDs are a set of extortionate strategies that can force a linear relationship between the long-run scores of both themselves and the

opponent, therefore ensuring that the opponent will never do better than them. Press and Dyson's suggested that the ZDs were the dominant set of strategies in the IPD, and as memory did not benefit them then they argued that memory is not beneficial for any strategy. In [13, 121, 130, 131, 132, 133, 165, 166, 175, 233, 274, 275, 276] the effectiveness of ZDs is questioned. Namely, [275, 276] showed that memory-one strategies must be forgiving to be evolutionarily stable and [68, 121, 130, 165, 166, 175, 233] demonstrated that longer-memory strategies have an advantage over short memory strategies. Chapter 5, studies the set of memory-one strategies, and more specifically, best response memory-one strategies and reinforces the discussion that the best action is adaptability and not manipulation, and short memory can be limiting.

This section of the literature review covered the original computer tournaments of Axelrod, the early success of Tit For Tat in these tournaments and excessive amounts of IPD strategies. Though Tit For Tat was considered to be the most robust basic strategy, reciprocity was found to not be enough in environments with uncertainties. There are at least two properties, that have been discussed in this section, for coping with such uncertainties; generosity and contrition. Generosity is letting a percentage of defections go unpunished, and contrition is lowering a strategy's readiness to defect following an opponent's defection. The strategies covered in this section are all strategies of intelligent design. They have been designed by researchers and not surfaced from an indirect process, such strategies are covered in section 2.5.

In the later part of this section a series of new strategies which were built on the basic reciprocal approaches were presented, followed by the infamous memory-one strategies, the zero-determinant strategies. Though the ZDs can be proven to be robust in pairwise interactions they were found to be lacking in evolutionary settings and in computer tournaments. Evolutionary settings and the emergence of cooperation under natural selection are covered in the next section.

2.4 Evolutionary dynamics

As yet, the emergence of cooperation has been discussed in the contexts of the one shot PD game (Chapter 1) and the IPD round robin tournaments (section 2.3). In the PD it is known that cooperation will not emerge, furthermore, in a series of influential works Axelrod demonstrated that reciprocal behaviour favours cooperation when individuals interact repeatedly. But does natural selection favour cooperation? Understanding the

conditions under which natural selection can favour cooperative behaviour is important in understanding social behaviour amongst intelligent agents [54].

Imagine a mixed population of cooperators and defectors where every time two individuals meet they play a game of PD. In such population the average payoff for defectors is always higher than cooperators. Under natural selection the frequency of defectors will steadily increase until cooperators become extinct (Figure 2.1). Natural selection favours defection in the PD, however, there are several mechanisms that allow the emergence of cooperation in an evolutionary context which will be covered in this section.

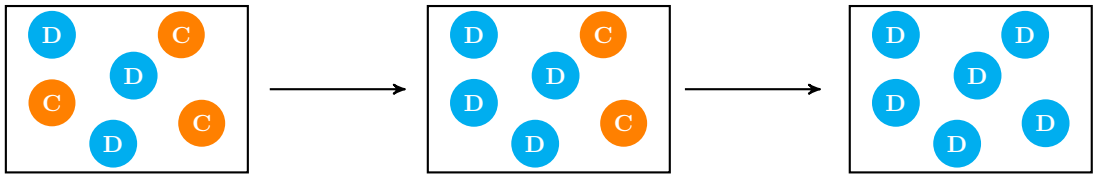


Figure 2.1: Natural selection favours defection in a mixed population of **Cooperator** (s) and **Defector** (s).

In the later sections of [33], Axelrod discusses an ecological tournament that he performed using the 62 strategies of the second tournament to understand the reproductive success of **Tit For Tat**. In an ecological tournament the prevalence of each type of strategy in each round is determined by that strategy's success in the previous round. The competition in each round becomes stronger as weaker performers are reduced and eliminated. The ecological simulation concluded with a handful of nice strategies dominating the population whilst exploitative strategies had died off. That was because the weaker strategies which were being exploitative were becoming extinct, and exploitative strategies were loosing their prey.

This new result led Axelrod to study the IPD in an evolutionary context based on several of the approaches established by the biologist John M. Smith [269, 270, 271]. Smith was a fundamental figure in evolutionary game theory and a participant of Axelrod's second tournament. The biological applications of the new evolutionary approach [34] won Axelrod and his co-author William Donald Hamilton the Newcomb-Cleveland prize of the American Association for the Advancement of Science. In [34] pairs of individuals from a population played the IPD. The number of interactions between the pairs were not fixed, but there was a probability defined w, where $0 < w < 1$, that the pair would interact again. This was referred to as the *importance of the future* of the game. It was shown that for a sufficient high w **Tit For Tat** strategies

would become common and remain common because they were "collectively stable". Axelrod argued that collective stability implied evolutionary stability (ESS) and that when a collectively stable strategy is common in a population and individuals are paired randomly, no other rare strategy can invade. However, Boyd and Lorderbaum in [54] proved that if w is large enough then no pure strategy is ESS because it can always be invaded by any pair of other strategies. This was also independently proven in [239].

These conclusions were made in populations where the individuals could all interact with each other. In 1992, Nowak and May considered a structured population where an individual's interactions were limited to its neighbours. More specifically, in [185] they explored how local interaction alone can facilitate population wide cooperation in a one shot PD game. The two deterministic strategies Defector and Cooperator were placed onto a two dimensional square array where the individuals could interact only with the immediate neighbours. The number of immediate neighbours could be either fourth, six or eight, as shown in Figure 2.2, where each node represents a player and the edges denote whether two players will interact. This topology is referred to as *spatial topology*. Each cell of the lattice is occupied by a Cooperator or a Defector and at each generation step each cell owner interacts with its immediate neighbours. The score of each player is calculated as the sum of all the scores the player achieved at each generation. At the start of the next generation, each lattice cell is occupied by the player with the highest score among the previous owner and their immediate neighbours.

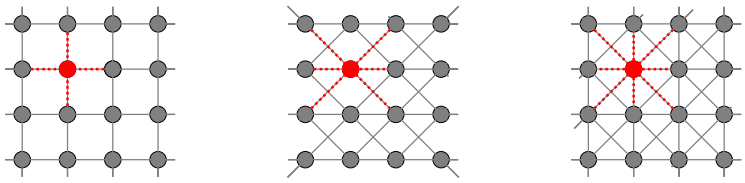


Figure 2.2: Spatial neighbourhoods

Limited/Local interactions proved that as long as small clusters of cooperators form, where they can benefit from interactions with other cooperators while avoiding interactions with defectors, global cooperation will continue. Thus, local interactions proved that even for the PD cooperation can emerge. Moreover in [228], whilst using the donation game (Equation (1.4)), it was shown that cooperation will evolve in a struc-

tured population as long as the benefit to cost ratio b/c is higher than the number of neighbours.

In structured populations local interactions that can dynamically change were considered in [300]. Graphs with a probability of rewiring connections were considered, and the rewire could be with any given node in the graphs and not just with immediate neighbours. Perc et al. concluded that "making new friends" may be an important activity for the successful evolution of cooperation, but also they must be selected carefully and one should keep their number limited. Finally, in [68] it was shown that cooperation can evolve as a stable strategy in large social groups when there are sufficient interactions on each round and players are able to base future play on their observations of other players' past actions (have a large memory).

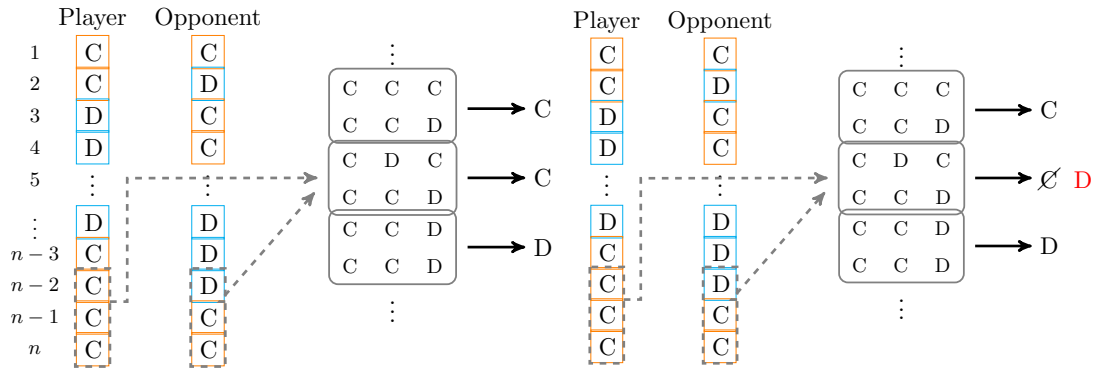
Another approach for increasing the likelihood of cooperation by increasing of assortative interactions among cooperative agents, include partner identification methods such as reputation [148, 224, 279], communication tokens [206] and tags [62, 120, 206, 246].

This section considered papers on evolutionary dynamics and mechanisms that ensure the emergence, or not, of cooperation. The emergence of cooperation using such mechanisms has not only been explored using only the PD but also other games such as the public goods game [155]. The following section focuses on strategy archetypes, training methods and strategies obtained from training.

2.5 Structured strategies and training

This section covers strategies that are different to that of intelligent design discussed in section 2.3. These are strategies that have been through a *training process* using generic strategy archetypes. For example, in [35] Axelrod explored deterministic strategies that took into account the last 3 plays of both players. As discussed in section 2.3.1, for each turn there are 4 possible outcomes, CC, CD, DC, DD, thus for 3 turns there are a total of $4 \times 4 \times 4 = 64$ possible combinations. Therefore, the strategy can be defined by a series of 64 C's/D's, corresponding to each combination; this type of strategy is called a *lookup table*. A graphical representation of the look up table strategy in [35] is given by Figure 2.3a. In [35] lookup tables were trained using a genetic algorithm [168]. A training process includes making random changes to a given instant of the lookup table, Figure 2.3b. The strategy which corresponds to the new altered instant is evaluated in

a given setting set by the experiment, and if the utility of the strategy has increased this change is kept and its genes are passed on to a new generation of strategies. A genetic algorithm is not the only heuristic method which can be used for training strategies, realistically any heuristic method can be used.



(a) A graphical representation of a look up table player which considers 3 plays of both players.

(b) Training a look up table player includes making changes to the strategy's responses to a history.

Figure 2.3: A graphical representation of the lookup table strategy described in [35], and a demonstration of the changes a strategy exhibits during training.

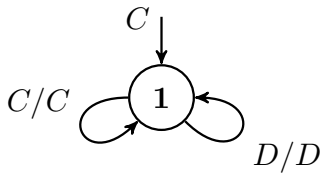
In 1996 John Miller considered finite state automata as an archetype [205], more specifically, Moore machines [212]. The training process used a genetic algorithm and the strategies were evaluated in a tournament with noise. Miller's results showed that even a small difference in noise (from 1% to 3%) significantly changed the characteristics of the evolving strategies. The strategies he introduced were **Punish Twice**, **Punish Once For Two Tats** and **Punish Twice and Wait**. A training combination of finite state automata and a genetic algorithm was also considered in [28]. In a series of experiments where the size of the population varied, there were two strategies frequently developed by the training process, and moreover, they were developed only after the evolution had gone on for many generations. These were **Fortress3** and **Fortress4**.

Also, in 1996 the first structured strategies based on neural networks that had been trained using a genetic algorithm were introduced in [123] by Harrald and Fogel. Harrald and Fogel considered a single layered neural network which had 6 inputs. These were the last 3 moves of the player and the opponent, similar to [35]. Neural networks have broadly been used since 1996 to train IPD strategies [23, 26, 70, 84] with training methods such as genetic algorithms [26, 64, 191] and particle swarm optimisation [84]. Chapter 7 of this thesis discusses the training of strategies using neural network in

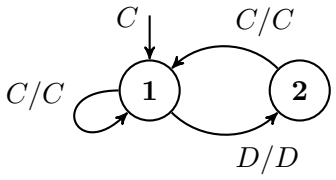
more details, as the aim of the chapter is to use an extension of a neural network, a *recurrent neural network*, to train an IPD strategy.

In [121, 165] both genetic algorithm and particle swarm optimisation were used to introduce a series of structured strategies based on lookup tables, finite state machines, neural networks, hidden Markov models [78] and Gambler. Hidden Markov models, are a stochastic variant of a finite state machine and Gamblers are stochastic variants of lookup tables. The structured strategies that arised from the training were put up against a large number of strategies in (1) a Moran process, which is an evolutionary model of invasion and resistance across time during which high performing individuals are more likely to be replicated and (2) a round robin tournament with 200 strategies. The top spots were dominated by the trained strategies of all the archetypes. The top three strategies were **Evolved LookerUp 2 2 2**, **Evolved HMM 5** and **Evolved FSM 16**. In [165] it was demonstrated that these trained strategies would overtake the population in a Moran process. The strategies evolved an ability to recognise themselves by using a handshake. This recognition mechanism allowed the strategies to resist invasion by increasing the interactions between themselves, an approach similar to the one described in section 2.4.

Throughout the different methods of training that have been discussed in this section, a spectrum of structured strategies can be found. Differentiating between strategies is not always straightforward. It is not obvious looking at a finite state diagram how a machine will behave, and many different machines, or neural networks can represent the same strategy. For example Figure 2.4 shows two finite automata and both are a representation of **Tit For Tat**.



(a) **Tit For Tat** as a finite state machine with 1 state.



(b) **Tit For Tat** as a finite state machine with 2 states.

Figure 2.4: Finite state machine representations of **Tit For Tat**. A machine consists of transition arrows associated with the states. Each arrow is labelled with A/R where A is the opponent's last action and R is the player's response. Finite state machines consist of a set of internal states. In (a) **Tit For Tat** finite state machine consists of 1 state and in (b) of 2.

To allow for identification of similar strategies a method called *fingerprinting* was

introduced in [22] by Daniel Ashlock. The method of fingerprinting is a technique for generating a functional signature for a strategy [23]. This is achieved by computing the score of a strategy against a spectrum of opponents. The basic method is to play the strategy against a probe strategy with varying noise parameters. In [22] Tit For Tat is used as the probe strategy. In Figure 2.5 an example of Pavlov's fingerprint is given. Fingerprinting has been studied in depth in [23, 24, 25, 26]. Another type of fingerprinting is the *transitive fingerprint* [7]. The method represents the cooperation rate of a strategy against a set of opponents over a number of turns. An example of a transitive fingerprint is given in Figure 2.6.

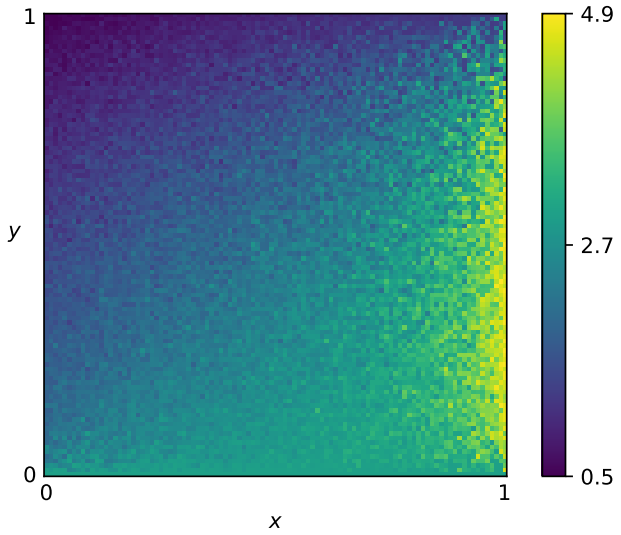


Figure 2.5: Pavlov fingerprinting with Tit For Tat used as the probe strategy. Figure was generated using [7].

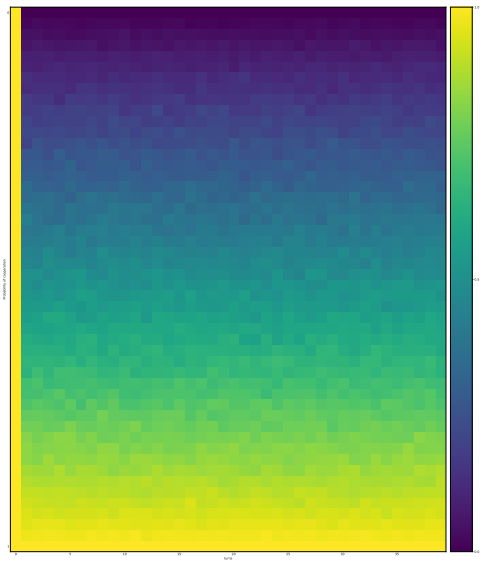


Figure 2.6: Transitive fingerprint of Tit For Tat against a set of 50 random opponents with varying cooperation rate.

This section covered a series of structured strategies based on different archetypes which have been trained via different training methods. The works discussed in this section has demonstrated that through these indirect training processes successful IPD strategies can emerge. This thesis explores both strategies of intelligent design and trained strategies in more details. The next section covers software that has been developed with the main aim of simulating the IPD interactions.

2.6 Software

Aside from human subject research the research of the IPD heavily relies on software. Many academic fields suffer from lack of source code availability and the IPD is not an exception. Several of the tournaments that have been discussed so far were generated using computer code, though not all of the source code is available. The code for Axelrod's original tournament is known to be lost, and moreover, for the second tournament the only source code available is the code for the 62 strategies (found on Axelrod's personal website [1]).

Several projects, however, are open source and available. Two educational platforms include [2] and [3]. The "Game of Trust" [2] is an on-line educational platform with a graphical user interface that introduces the basics of game theory, the IPD and the notion of strategies. It attracted a lot of attention due to being "well-presented with scribble-y hand drawn characters" [136] and "a whole heap of fun" [161]. Secondly, [3] is a personal project written in PHP with also a graphical user interface. The project offers a big collection of strategies and allows users to simulate their own tournaments using the listed strategies.

Two open source projects used for research include [4] and [7]. PRISON [4] is written in the programming language Java and it was launched in 1998. The project includes a good number of strategies and has been used in several publications [43, 44, 192, 194]. Axelrod-Python library [7] is another software used in a number of publications [89, 114, 121, 164, 165, 229, 289]. It is written in the programming language Python and contains the largest collection of strategies in the field. The strategy list of the project itself has been cited by publications [16, 126, 216]. The Axelrod-Python project has been implemented following best practices and the tools used in this thesis to simulate IPD interactions.

2.7 Chapter summary

This Chapter presented a literature review of the IPD. The opening sections focused on research trends and published works of the field, followed by a presentation of research and educational software. More specifically, section 2.2 covered the early years of research. This was when simulating turns of the game was only possible with human subject research. Following the early years, the pioneering tournaments of Axelrod were introduced in section 2.3. Axelrod's work offered the field an agent based game theoretic framework to study the IPD. In his original papers he asked researchers to design strategies to test their performance with the new framework. The winning strategy of both his tournaments was Tit For Tat. The strategy however came with limitations which were explored by other researchers, and new intelligently designed strategies were introduced in order to surpass Tit For Tat with some contributions such as Pavlov and Gradual.

Soon researchers came to realise that strategies should not just do well in a tournament setting but should also be evolutionary robust. Evolutionary dynamic methods were applied to many works in the field, and factors under which cooperation emerges were explored, as described in section 2.4. This was not done only for unstructured populations, where all strategies in the population can interact with each other, but also in population where interactions were limited to only strategies that were close to each other. In such topologies it was proven that even in the one shot game, cooperation can indeed emerge.

Evolutionary approaches can offer many insights in the study of the PD. In evolutionary settings strategies can learn to adapt and take over population by adjusting their actions; such algorithms can be applied so that evolutionarily robust strategies can emerge. Algorithms and structures used to train strategies in the literature were covered in section 2.5. From these training methods several strategies are found, and to be able to differentiate between them fingerprinting was introduced. The research of best play and cooperation has been going on since the 1950s, and for simulating the game software has been developed along the way. This software has been briefly discussed in section 2.6.

The study of the PD is still an ongoing field research where new variants and new structures of strategies are continuously being explored [227]. The game now serves as a model in a wide range of applications, for example in medicine and the study of

cancer cells [19, 158], as well as in social situations and how they can be driven by rewards [76]. This thesis aims to contribute to the continued understanding of this well known and widely applied game theoretic model. Many of the papers reviewed in this Chapter have served as motivation to the research presented in the following Chapters. In Chapter 4 the performance of several of the strategies mentioned in this Chapter is evaluated in a large number of tournaments. Chapter 5 explores the set of memory-one strategies, and Chapters 6 and 7 explore trained strategies based on archetypes such as sequences and recurrent neural networks.

Chapter 3

A bibliometric study of research topics, collaboration and influence in the field of the Iterated Prisoner's Dilemma

The research reported in this Chapter has lead to a manuscript, entitled:

"A bibliometric study of research topics, collaboration and influence in the field of the Iterated Prisoner's Dilemma"

Available at: `arxiv.org/abs/1911.06128`

Associated data sets: [98, 99, 100]

Associated codebase: [107]

The manuscript's abstract is the following:

This manuscript explores the research topics and collaborative behaviour of authors in the field of the Prisoner's Dilemma using topic modelling and a graph theoretic analysis of the co-authorship network. The analysis identified five research topics in the Prisoner's Dilemma which have been relevant of the course of time. These are human subject research, biological studies, strategies, evolutionary dynamics on networks and modelling problems as a Prisoner's Dilemma game. Moreover, the results demonstrated the Prisoner's Dilemma is a field of continued interest, and although it is a collaborative field, it is not necessarily more collaborative than other scientific fields.

The co-authorship network suggests that authors are focused on their communities and not many connections across the communities are made. The Prisoner Dilemma authors also do not influence or gain much information by their connections, unless they are connected to a "main" group of authors.

The differences between the Chapter and the manuscript include details on the open source package Arcas which was used for the data collection. The Chapter includes snippets of its source code, its unit tests and examples of its usage.

3.1 Introduction

This Chapter presents a bibliometric analysis of the data set "Articles' meta data on the Prisoner's Dilemma" [100]. Chapter 2 presented a review of published works on the PD, and manually assigned them to different topics. To complement that manual identification of topics this Chapter presents an automatic approach using natural language processing. More specifically, the 2,422 articles' metadata in [100] are used to extract a list of research topics in the field. Moreover, the list of authors in [100] is also used to generate a co-authorship network and explore their collaborative behaviour. The Chapter is structured as follows:

- section 3.2 covers the data collection progress, an introduction to topic modelling and to co-authorship networks.
- section 3.3 presents a preliminary analysis of the collected data set.
- section 3.4, identifies research topics in the field using natural language processing.
- section 3.5 evaluates the collaborative behaviour of the field based on the publications of 4,226 authors.

3.2 Methodology

As discussed in [299], bibliometrics (the statistical analysis of published works originally described by [238]) has been used to support historical assumptions about the development of fields [240], identify connections between scientific growth and policy changes

[71], develop a quantitative understanding of author order [261], and to investigate the collaborative structure of an interdisciplinary field [183]. Most academic research is undertaken in the form of collaborative effort and as [173] points out, it is rational that two or more people have the potential to do better as a group than individually. Indeed this is the very premise of the PD itself. Collaboration in groups has a long tradition in experimental sciences and it has be proven to be productive according to [79]. The number of collaborations can be different between research fields and understanding how collaborative a field is not always an easy task. Several studies tend to consider academic citations as a measure. A blog post published by Nature [219] argues that depending on citations can often be misleading because the true number of citations can not be known. Citations can be missed due to data entry errors, academics are influenced by many more papers than they actually cite and several of the citations are superficial.

A more recent approach to measuring collaborative behaviour, and to studying the development of a field is to use the co-authorship network, as described in [183]. The co-authorship network has many advantages as several graph theoretic measures can be used as proxies to explain author relationships. For example the average degree of a node corresponds to the average number of an authors' collaborators, and clustering coefficient corresponds to the extent that two collaborators of an author also collaborate with each other. In [183], the approach was applied to analyse the development of the field "evolution of cooperation", and in [299] to identify the subdisciplines of the interdisciplinary field of "cultural evolution" and investigate trends in collaboration and productivity between these subdisciplines. This Chapter builds on the works of [183] and [299], and extends their methodology as it will be described in section 3.2.2.

Latent Dirichlet Allocation (LDA) is a topic modelling technique proposed in [52] as a generative probabilistic model for discovering underlying topics in collections of data. Applications of the technique include detection in image data [14, 67] and detection in video [217, 291]. Nevertheless, LDA has been applied by several works on publication data for identifying the topic structure of a subject area. In [144] it was applied to the publications on mathematical education of the journals "Educational Studies in Mathematics" and "Journal for Research in Mathematics Education", in [278] to the dissertations of the North American library and Information Science and in [50] to conference papers presented at EvoLang conferences. LDA is the topic modelling technique used in this thesis to identify research topics. An introduction to the technique

is presented in section 3.2.3.

The individual techniques of the methodology and their applications to research fields have been done before [50, 183, 278, 299]. The novelty of this Chapter is the combination of these techniques and their application to a new data set. A data set that has been collected from five different sources, whereas the data sets of previous works [183, 299] were from a single source (Web of Science). A bespoke open source software was used to carry out the data collection. Details of the software and the data collection process are presented in section 3.2.1.

3.2.1 Data collection

Academic articles are accessible through scholarly databases. Several databases and collections today offer access through an open application protocol interface (API). An API allows users to query directly a journal's database and bypass the graphical user interface. Interacting with an API has two phases: requesting and receiving. The request phase includes composing a url with the details of the request. For example,

```
http://export.arxiv.org/api/query?search_query=abs:prisoner'sdilemma&max_
results=1
```

represents a request message. The first part of the request is the address of the API. In this example the address corresponds to the API of arXiv. The second part of the request contains the search arguments. In this example it is requested a single article that the word "Prisoner's Dilemma" exists within the article's title. The format of the request message is different from API to API. The receive phase includes receiving a number of raw metadata of articles that satisfies the request message. The raw metadata are commonly received in extensive markup language (xml) or Javascript object notation (json) formats [225]. Similarly to the request message, the structure of the received data differs from journal to journal.

To ensure that the research reported in this Chapter can be reproduced all code used to query the different APIs has been packaged as a Python library called Arcas. The source code of the library has been made available online and the package includes documentation of usage which is available at: `http://arcas.readthedocs.io/en/latest/`. Arcas allows users to communicate with a list of APIs by specifying a single keyword whilst not considering the differences between the requesting and receiving phases of the APIs.

Consider the example of retrieving a single article with the word "Prisoner's Dilemma" in the title. Figure 3.1 demonstrates the Python code needed to query the publisher PLOS and Figure 3.2 demonstrates code for querying the API of Nature. The only distinction between the two code snippets is their respective line 2 where the API is specified by creating an instance of a class corresponding to the publisher's API. The differences between querying the two APIs are visible from lines 6 and 14-onwards. Lines 6 show the requesting message and lines 14-onwards show the metadata of the article received by each source.

```
1  >>> import arcas
2  >>> api = arcas.Plos()
3  >>> parameters = api.parameters_fix(title="Prisoner's Dilemma", records=1)
4  >>> url = api.create_url_search(parameters)
5  >>> url
6  'http://api.plos.org/search?q=title:"Prisoner\'s Dilemma"&rows=1'
7
8  >>> request = api.make_request(url)
9  >>> root = api.get_root(request)
10 >>> article = api.parse(root)
11
12 >>> for key in article[0].keys():
13 ...     print(key)
14 id
15 journal
16 eissn
17 publication_date
18 article_type
19 author_display
20 abstract
21 title_display
22 score
```

Figure 3.1: Example of using the library Arcas to communicate the API of the publisher PLOS. The query is for a single article with the word "Prisoner's Dilemma" in the title.

There are differences and similarities between the retrievable metadata of each API. Arcas includes a function which standarises the format of querying results. Figure 3.3 demonstrates the usage of the function.

At the time of writing there are a total of five different APIs implemented within the project. These five include APIs of four prominent publishers in the field and a preprint server. Namely these are:

- arXiv [198]; a repository of electronic preprints. It consists of scientific papers in the fields of mathematics, physics, astronomy, electrical engineering, computer science, quantitative biology, statistics, and quanti-

```
>>> import arcas
>>> api = arcas.Nature()
>>> parameters = api.parameters_fix(title="Prisoner's Dilemma", records=1)
>>> url = api.create_url_search(parameters)
>>> url
"http://www.nature.com/opensearch/request?&query=dc.title adj Prisoner's Dilemma&maximumRecords=1"

>>> request = api.make_request(url)
>>> root = api.get_root(request)
>>> article = api.parse(root)

>>> for key in article[0].keys():
...     print(key)
records
record
recordSchema
recordPacking
recordData
message
article
head
identifier
title
creator
productCode
publicationName
issn
eIssn
doi
publisher
publicationDate
volume
number
startingPage
endingPage
url
genre
description
copyright
aggregationType
```

Figure 3.2: Example of using the library Arcas to communicate the API of the publisher Nature. The query is for a single article with the word "Prisoner's Dilemma" in the title.

```
>>> import arcas
>>> api = arcas.Plos()
>>> parameters = api.parameters_fix(title="Prisoner's Dilemma", records=1)
>>> url = api.create_url_search(parameters)

>>> request = api.make_request(url)
>>> root = api.get_root(request)
>>> article = api.parse(root)

>>> meta_data = api.to_dataframe(article[0])
>>> meta_data.columns
Index(['url', 'key', 'unique_key', 'title', 'author', 'abstract', 'doi',
       'date', 'journal', 'provenance', 'category', 'score', 'open_access'],
      dtype='object')
```

Figure 3.3: Python Code. Arcas includes a function which standarises the results of the queries regarding the API.

tative finance, which all can be accessed online.

- PLOS [5]; a library of open access journals and other scientific literature under an open content license. It launched its first journal, PLOS Biology, in October 2003 and publishes seven journals, as of October 2015.
- IEEE Xplore Digital Library (IEEE) [140]; a research database for discovery and access to journal articles, conference proceedings, technical standards, and related materials on computer science, electrical engineering and electronics, and allied fields. It contains material published mainly by the Institute of Electrical and Electronics Engineers and other partner publishers.
- Nature [116]; a multidisciplinary scientific journal, first published on 4 November 1869. It was ranked the world's most cited scientific journal by the Science Edition of the 2010 Journal Citation Reports and is ascribed an impact factor of 40.137, making it one of the world's top academic journals.
- Springer [199]; a leading global scientific publisher of books and journals. It publishes close to 500 academic and professional society journals.

Each API has a corresponding class implemented in Arcas. The classes include a series of methods which allow Arcas to communicate with the APIs. An example of an API class is given by both Figure 3.4 and Figure 3.5. These include the classes for the APIs of arXiv and IEEE. Note that IEEE is an example of an API which requires

a user to have an access key (line 7 in Figure 3.5). An access key can be required from the publisher's website and for the APIs of this Chapter they can be acquired for free.

As mentioned in Chapter 1 the source code associated with the research projects of this thesis have been written following a set of best practices. These best practices include unit testing. There are a series of unit tests that test the functionality and correctness of each API class. For example, Figure 3.6 displays a test case for the method `to_dataframe` of the `Arxiv` class. Moreover, Figure 3.7 shows several unit tests which ensure that the request url for IEEE, with different search arguments, is being generated correctly.

The 2,422 articles metadata explored in this Chapter has been collected using Arcas. More specifically, articles for which any of the terms "prisoner's dilemma", "prisoners dilemma", "prisoner dilemma", "prisoners evolution", "prisoner game theory" existed within the title, the abstract or the text are included in the analysis. The data set has been archived and is available at [100]. Note that the latest data collection was performed on the 30$^{\text{th}}$ November 2018.

3.2.2 Co-authorship network

The relationship between the authors within a field will be modelled as a graph $G = (V_G, E_G)$ where V_G is the set of nodes and E_G is the set of edges. The set V_G represents the authors and an edge connects two authors if and only if those authors have written together. This co-authorship network is constructed using the data set [100] and the open source package [119]. The PD network is denoted as G where the number of unique authors $|V(G)|$ is 4,226 and $|E(G)|$ is 7,642 . All authors' names were formatted as their first name and last name (i.e. Martin A. Nowak to Martin Nowak). This was done to avoid errors such as Martin A. Nowak and Martin Nowak being treated as a different person. There are some authors for which only their first initial was found. These entries are left as such.

The collaborativeness of the authors will be analysed using measures such as, isolated nodes, connected components, clustering coefficient, communities, modularity and average degree. These measures show the number of connections authors can have and how strongly connected these people are. The number of isolated nodes is the number of nodes that are not connected to another node, thus the number of authors that

```python
class Arxiv(Api):
    def __init__(self):
        self.standard = 'http://export.arxiv.org/api/query?search_query='

    def to_dataframe(self, raw_article):
        """A function which takes a dictionary with structure of the arXiv results,
        transforms it to a standardised format and returns a dataframe."""
        raw_article['url'] = raw_article.get('id', None)

        for key_one, key_two in [['author', 'name'], ['category', 'category']]:
            raw_article[key_one] = raw_article.get(key_two, None)
            if raw_article[key_one] is not None:
                raw_article[key_one] = raw_article[key_one].split(',')

        raw_article['abstract'] = raw_article.get('summary', None)
        raw_article['date'] = int(raw_article.get('published', '0').split('-')[0])
        raw_article['journal'] = raw_article.get('journal_ref', None)
        if raw_article['journal'] is None:
            raw_article['journal'] = "arXiv"

        raw_article['provenance'] = 'arXiv'
        raw_article['title'] = raw_article.get('title', None)
        raw_article['doi'] = raw_article.get('doi', None)
        raw_article['key'], raw_article['unique_key'] = self.create_keys(raw_article)

        raw_article['open_access'] = True
        raw_article['score'] = 'Not available'
        return self.dict_to_dataframe(raw_article)

    def parse(self, root):
        """Removing unwanted branches."""
        branches = root.getchildren()
        raw_articles = []
        for record in branches:
            if 'entry' in record.tag:
                raw_articles.append(self.xml_to_dict(record))
        if not raw_articles:
            raw_articles = False
        return raw_articles

    @staticmethod
    def parameters_fix(author=None, title=None, abstract=None, year=None, records=None,
                       start=None, category=None, journal=None, keyword=None):
        parameters = []
        if author is not None:
            parameters.append('au:{}'.format(author))
        if title is not None:
            parameters.append('ti:{}'.format(title))
        if abstract is not None:
            parameters.append('abs:{}'.format(abstract))
        if category is not None:
            parameters.append('cat:{}'.format(category))
        if journal is not None:
            parameters.append('jr:{}'.format(journal))
        if keyword is not None:
            parameters.append('all:{}'.format(keyword))
        if records is not None:
            parameters.append('max_results={}'.format(records))
        if start is not None:
            parameters.append('start={}'.format(start))
        if year is not None:
            print('arXiv does not support argument year.')

        return parameters

    @staticmethod
    def get_root(response):
        root = ElementTree.fromstring(response.text)
        return root
```

Figure 3.4: `Arxiv` class implementation in Arcas. It includes the code necessary for Arcas to query the API of arXiv.

```python
class Ieee(Api):
    """
        API argument is 'ieee'.
    """
    def __init__(self):
        self.standard = 'https://ieeexploreapi.ieee.org/api/v1/search/articles?'
        self.key_api = api_key

    def create_url_search(self, parameters):
        """Creates the search url, combining the standard url and various
        search parameters."""
        url = self.standard
        url += parameters[0]
        for i in parameters[1:]:
            url += '&{}'.format(i)
        url += '&apikey={}'.format(self.key_api)
        return url

    @staticmethod
    @ratelimit.rate_limited(3)
    def make_request(url):
        """Request from an API and returns response."""
        response = requests.get(url, stream=True, verify=False)
        if response.status_code != 200:
            raise APIError(response.status_code)
        return response

    def parse(self, root):
        """Parsing the xml file"""
        if root['total_records'] == 0:
            return False
        return root['articles']

    @staticmethod
    def parameters_fix(author=None, title=None, abstract=None, year=None,
                       records=None, start=None, category=None, journal=None,
                       keyword=None):
        parameters = []
        if author is not None:
            parameters.append('author={}'.format(author))
        if title is not None:
            parameters.append('article_title={}'.format(title))
        if abstract is not None:
            parameters.append('abstract={}'.format(abstract))
        if year is not None:
            parameters.append('publication_year={}'.format(year))
        if category is not None:
            parameters.append('index_terms={}'.format(category))
        if journal is not None:
            parameters.append('publication_title={}'.format(journal))
        if keyword is not None:
            parameters.append('querytext={}'.format(keyword))
        if records is not None:
            parameters.append('max_records={}'.format(records))
        if start is not None:
            parameters.append('start_record={}'.format(start))

        return parameters

    @staticmethod
    def get_root(response):
        root = response.json()
        return root
```

Figure 3.5: `Ieee` class implementation in Arcas. It includes the code necessary for Arcas to query the API of IEEE.

```python
import arcas

def test_to_dataframe():
    dummy_article = {'entry': '\n', 'id': 'http://arxiv.org/abs/0000',
                     'updated': '2011', 'published': '2010', 'title': 'Title',
                     'summary': "Abstract", 'author': '\n', 'name': 'E Glynatsi, V Knight',
                     'doi': '10.0000', 'comment': 'This is a comment.',
                     'journal_ref': 'Awesome Journal', 'primary_category': 'Dummy',
                     'category': None}
    api = arcas.Arxiv()
    article = api.to_dataframe(dummy_article)

    assert isinstance(article, pandas.core.frame.DataFrame)
    assert list(article.columns) == api.keys()
    assert len(article['url']) == 2

    assert article['url'].unique()[0] == 'http://arxiv.org/abs/0000'
    assert article['key'].unique()[0] == 'Glynatsi2010'
    assert article['title'].unique()[0] == 'Title'
    assert article['abstract'].unique()[0] == 'Abstract'
    assert article['journal'].unique()[0] == 'Awesome Journal'
    assert article['primary_category'].unique()[0] == 'Dummy'
    assert article['category'].unique()[0] == None
    assert article['score'].unique()[0] == 'Not available'
    assert article['open_access'].unique()[0] == True
```

Figure 3.6: Unit tests for the `Arxiv` class. Tests the functionality of the method `to_dataframe`.

```python
import arcas

def test_setup():
    api = arcas.Ieee()
    assert api.standard == 'https://ieeexploreapi.ieee.org/api/v1/search/articles?'

def test_parameters_and_url_author():
    api = arcas.Ieee()
    parameters = api.parameters_fix(author='Glynatsi')
    assert parameters == ['author=Glynatsi']

    url = api.create_url_search(parameters)
    assert (
    url ==
    "https://ieeexploreapi.ieee.org/api/v1/search/articles?author=Glynatsi&apikey=Your key here"
    )

def test_parameters_and_url_title():
    api = arcas.Ieee()
    parameters = api.parameters_fix(title="Game")
    assert parameters == ["article_title=Game"]
```

Figure 3.7: Unit tests for the `Ieee` class.

have published alone. The average degree denotes the average number of neighbours for each nodes, i.e. the average number of collaborations between the authors. A connected component is a maximal set of nodes such that each pair of nodes is connected by a path [77]. The number of connected components as well as the size of the largest connected component in the network are reported. The size of the largest connected component represents the scale of the central cluster of the entire network, as will be discussed in later parts. Clustering coefficient and modularity are also calculated. The clustering coefficient, defined as 3 times the number of triangles on the graph divided by the number of connected triples of nodes, is a local measure of the degree to which nodes in a graph tend to cluster together in a clique [77]. It shows to which extent the collaborators of an author also write together.

In comparison, modularity is a global measure designed to measure the strength of division of a network into communities. The number of communities will be reported using the Clauset-Newman-Moore method [66]. Also the modularity index is calculated using the Louvain method described in [53]. The value of the modularity index can vary between $[-1, 1]$, a high value of modularity corresponds to a structure where there are dense connections between the nodes within communities but sparse connections between nodes in different communities. That means that there are many sub communities of authors that write together but not across communities.

Two further points are aimed to be explored in this thesis, (1) which people control the flow of information; as in which people influence the field the most and (2) which are the authors that gain the most from the influence of the field. To measure these concepts centrality measures are going to be used. Centrality measures are often used to understand different aspects of social networks [174]. The two centrality measures chosen here are closeness and betweenness centrality.

1. In networks some nodes have a short distance to a lot of nodes and consequently are able to spread information on the network very effectively. A representative of this idea is *closeness centrality*, where a node is seen as centrally involved in the network if it requires only few intermediaries to contact others and thus is structurally relatively independent. Closeness centrality is interpreted as influence. Authors with a high value of closeness centrality, are the authors that spread scientific knowledge easier on the network and they have high influence.

2. Another centrality measure is the *betweenness centrality*, where the determination

of an author's centrality is based on the quotient of the number of all shortest paths between nodes in the network that include the node in question and the number of all shortest paths in the network. In betweenness centrality the position of the node matters. Nodes with a higher value of betweenness centrality are located in positions that a lot of information pass through, this is interpreted as the gain from the influence, thus these authors gain the most from their networks.

3.2.3 Topic modelling

The articles contained in the data set will be classified into research topics using LDA, a topic modelling technique designed to summarise large collections of documents by a small number of conceptually connected topics or themes [52, 115]. LDA is carried out using [244].

The input to an LDA is a collection of documents, and the collection of documents considered here are the articles' abstracts. The output of an LDA is an $N \times n$ matrix - N rows for N abstracts and n columns for n topics. The cells contain the percentage contributions for each topic for each abstract, c_i^j for $i \in \{1, 2, \ldots, n\}$ for $j \in \{1, 2, \ldots, N\}$. Thus each document/abstract is represented by a distribution over topics, and the topics themselves are represented by a distribution over words. More specifically, each topic is described by weights associated with words. For example assume two topics A and B. Their associated words and weights are:

- Topic A: 0.039× "cooperation", 0.028× "study" and 0.026× "human".
- Topic B: 0.020× "cooperation", 0.028× "agents" and 0.026× "strategies".

A document with abstract "The study of cooperation in humans" has a $c_A = 0.039 + 0.028 + 0.026 = 0.093$ and $c_B = 0.020 + 0.0 + 0.0 = 0.020$. In essence, LDA maps every paper to a vector. In this example the document is mapped to $[0.093, 0.020]$. Each document has a dominant topic to which is going to be assigned in. The dominant topic is the topic with the highest percentage contribution denoted as c^*. For the given example the dominant topic is Topic A, and $c^* = c_A$.

LAD requires that the number of topics is specified in advance before running the algorithm. The number of topics can be chosen using the coherence value [248] or through subjective minimisation of the overlapping keywords between two topics. Both these approaches will be used here. Preceding the analysis of research topics, the next section presents a preliminary analysis of the collected data set.

3.3 Preliminary analysis

The data set consists of 2,422 articles with unique titles. In case of duplicates the preprint version of an article (collected from arXiv) was dropped. Similarly to [183], 76 articles have been manually added throughout the writing of Chapter 2 because they were of specific interest. These papers include [82] the first publication on the PD, [228, 274] two well cited articles in the field, and a series of works of Axelrod [32, 33, 34, 35, 246].

A more detailed summary of the articles' provenance is given by Table 3.1. Only 3% of the data set consists of articles that were manually added and 27% of the articles were collected from arXiv. The average number of publications is also included in Table 3.1. Overall an average of 43 articles are published per year on the topic. The most significant contribution to this appears to be from arXiv with 11 articles per year, followed by Springer with 9, and PLOS and Nature with 8.

	Number of Articles	Percentage %	Year of first publication	Average number of publications per year
IEEE	294	12.14%	1973	5
Manual	76	3.14%	1951	1
Nature	436	18.00%	1959	8
PLOS	477	19.69%	2005	8
Springer	533	22.01%	1966	9
arXiv	654	27.00%	1993	11

Table 3.1: Summary of [100] per provenance.

The data handled here is in fact a time series from the 1950s, the formulation of the game, until 2018 (Figure 3.8). Two observations can be made from Figure 3.8.

1. There is a steady increase of the number of publications since the 1970s.
2. There is a decrease in 2017-2018. This is due to the data set being incomplete. Articles that have been written in 2017-2018 have either not being published or were not retrievable by the APIs at the time of the last data collection.

These observations can be confirmed by studying the time series. Using [151], an exponential distribution is fitted to the data. The fitted model can be used to forecast the behaviour of the field for the next 5 years. Even though the time series has indicated a slight decrease, the model forecasts that the number of publications will keep increasing, thus demonstrating that the field of the PD continues to attract academic attention.

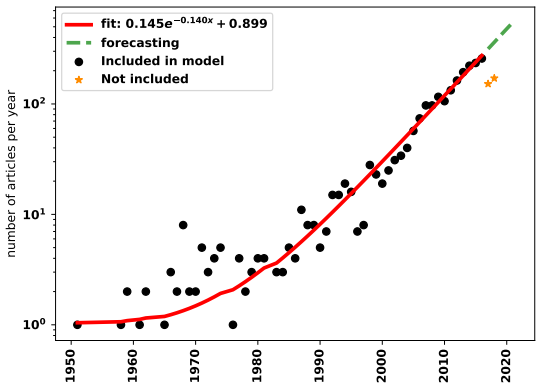


Figure 3.8: Number of articles published on the PD 1951-2018 (on a log scale), with a fitted exponential line, and a forecast for 2017-2022.

There are a total of 4,226 authors in the data set and several of these authors have had multiple publications collected from the data collection process. The highest number of articles collected for an author is 83 publications for Matjaz Perc. The distribution of the number of papers per author is given by Figure 3.9, and it can be seen that Perc is an outlier. More specifically, most authors have 1 to 6 publications in the data set.

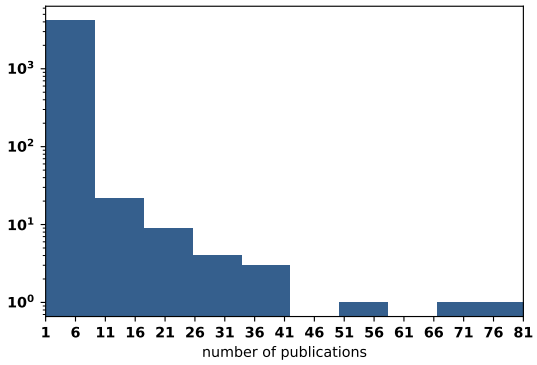


Figure 3.9: Distribution of number of papers per author (on a log scale).

The overall collaboration index or the average number of authors on multi-authored papers is 3.2, thus on average a non single author publication in the PD has 3 authors. This appears to be quite standard compared to other fields such as cultural evolution [299], Astronomy and Astrophysics, Genetics and Heredity, Nuclear and Particle Physics as reported by [190]. There are only a total of 545 publications with a single author, which corresponds to the 22% of the papers. It appears that academic publications tend to be undertaken in the form of collaborative effort, which is in line with the claim of [173]. From Figure 3.10 the trend of collaboration index over the years is given. There are some peaks in the early years 1969 and 1980, however, a steady

increase appears to happen after 2004. This could be an effect of better communication tools being introduced around that time which enabled more collaborations between researchers.

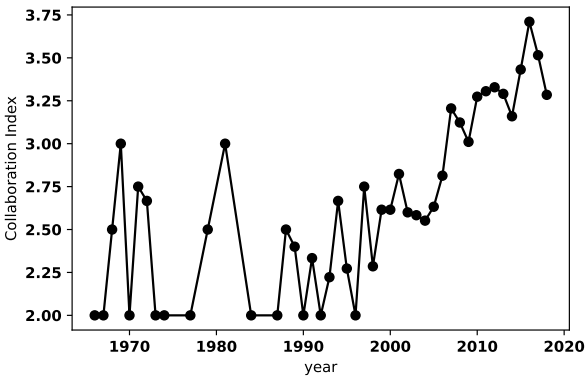


Figure 3.10: Collaboration index over time.

The collaborativeness of the authors is explored in more detail in section 3.5 using the co-authorship network. The collaborative behaviour and relative influence of authors will also be explored in co-authorship networks which correspond to their publications research topics. These topics are presented in the next section.

3.4 Research topics in the Prisoner's Dilemma research

In order to identify the topics which are being discussed in the field of the PD, the LDA algorithm implemented in [244] is applied to the abstracts of the data set. As mentioned before, the number of topics, which will be denoted as n, needs to be specified before running the algorithm. The appropriate number of topics is chosen based on the coherence value [248]. Figure 3.11 gives the coherence values of 18 models where $n \in \{2, 3, \ldots, 19\}$, and it can be seen than the most appropriate number of topics is 6 with a coherence value of 0.418.

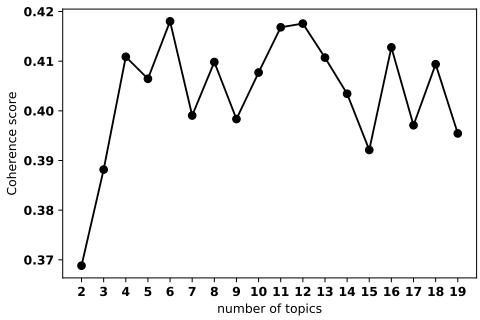


Figure 3.11: Coherence for LDA models over the number of topics.

The keywords associated with each topic for $n = 6$ are given by Table 3.2. Though $n = 6$ has the highest coherence score, from Table 3.2 it can be observed that there are overlapping keywords between the topics. Further manual investigation has revealed that the separation of topics are the most clear when an n of 5 is considered. The LDA model for $n = 5$ has a coherence value 0.406 which is close to 0.418 (the score for $n = 6$). Thus, $n = 5$ is chosen to carry out the LDA.

Topic	Topic Keywords
A	model, theory, system, base, paper, problem, propose, present, approach, provide, analysis, framework, method, develop, solution
B	behavior, social, human, decision, study, experiment, make, suggest, result, behaviour, effect, partner, participant, subject, experimental
C	individual, group, good, social, punishment, level, cost, mechanism, dilemma, cooperative, show, base, public, high, society
D	game, strategy, player, agent, play, dilemma, state, prisoner, payoff, equilibrium, result, iterate, set, probability, show
E	population, evolutionary, dynamic, model, selection, result, evolution, evolve, show, process, size, interaction, cooperator, change, system
F	cooperation, network, interaction, structure, study, evolution, find, behavior, cooperative, simulation, rule, spatial, cooperator, promote, result

Table 3.2: Keywords for each topic when $n = 6$. The highlighted keywords are overlapping keywords between topics.

For $n = 5$ the articles are clustered and assigned to their dominant topic based on the highest percentage contribution. The keywords associated with a topic, the most representative article of the topic (based on the percentage contribution) and its academic reference are given by Table 3.3. The topics are labelled as A, B, C, D and E, and more specifically:

- Based on the keywords associated with Topic A, and the most representative article, Topic A appears to be about human subject research. Several publications assigned to the topic study the PD by setting experiments and having human participants simulate the game instead of computer simulations. These articles include [195] which showed that prosocial behaviour increased with the age of the participants, [180] which studied the difference in cooperation between high-functioning autistic and typically developing children, [211] explored the gender effect in high school students and [45] explored the effect of facial expressions of individuals.

- Though it is not immediate from the keywords associated with Topic B, investigating the papers assigned to the topic indicate that it is focused on biological studies. Papers assigned to the topic include papers which apply the PD to genetics [259, 268], to the study of tumours [18, 260] and viruses [283]. Other works include how phenotype affinity can affect the emergence of cooperation [296] and modelling bacterial communities as a spatial structured social dilemma.

- Based on the keywords and the most representative article Topic C appears to include publications on PD strategies. Publications in the topic include the introduction of new strategies [275], the search of optimality in strategies [40] and the training of strategies [145] with different representation methods. Moreover, publications that study the evolutionary stability of strategies [13] and introduced methods of differentiating between them [23] are also assigned to C.

- The keywords associated with Topic D clearly show that the topic is focused on evolutionary dynamics on networks. Publications include [139] which explored the robustness of cooperation on networks, [288] which studied the effect of a strategy's neighbourhood on the emergence of cooperation and [61] which explored the fixation probabilities of any two strategies in spatial structures.

- The publications assigned to Topic E are on modelling problems as a PD game. Though Topic B is also concerned with problems being formulated as a PD, it includes only biological problems. In comparison, the problems in Topic E include decision making in operational research [230], information sharing among members in a virtual team [81], the measurement of influence in articles based on citations [138] and the price spikes in electric power markets [117], and not on biological studies.

Dominant Topic	Topic Keywords	Most Representative Article Title	Reference	# Documents	% Documents
A	social, behavior, human, study, experiment, cooperative, cooperation, suggest, find, behaviour	Facing Aggression: Cues Differ for Female versus Male Faces	[92]	496.0	0.2008
B	individual, group, good, show, high, increase, punishment, cost, result, benefit	Genomic and Gene-Expression Comparisons among Phage-Resistant Type-IV Pilus Mutants of Pseudomonas syringae pathovar phaseolicola	[268]	309.0	0.1251
C	game, strategy, player, agent, dilemma, play, payoff, state, prisoner, equilibrium	Fingerprinting: Visualization and Automatic Analysis of Prisoner's Dilemma Strategies	[268]	561.0	0.2271
D	cooperation, network, population, evolutionary, evolution, interaction, dynamic, structure, cooperator, study	Influence of initial distributions on robust cooperation in evolutionary Prisoner's Dilemma	[60]	556.0	0.2251
E	model, theory, base, system, problem, paper, propose, information, provide, approach	Gaming and price spikes in electric power markets and possible remedies	[117]	548.0	0.2219

Table 3.3: Keywords for each topic and the document with the most representative article for each topic.

Note that whilst for the choice of 5 topics the actual clustering is not subjective (the algorithm is determining the output) the interpretation above is.

Thus, the five topics in the PD publications identified by the data set of using an LDA are:

1. human subject research,
2. biological studies,
3. strategies,
4. evolutionary dynamics on networks,
5. modelling problems as a PD.

These topics nicely summarise the PD research. They highlight the interdisciplinarity of the field; how it brings together applied modelling of real world situations (Topic B and E) and more theoretical notions such as evolutionary dynamics and optimality of strategies.

Figure 3.12 gives the number of articles per topic over time. The topics appear to have had a similar trend over the years, with topics B and D having a later start. Following the introduction of a topic its publications have been increasing. There is no decreasing trend in any of the topics. All topics have been publishing for years and they still attract the interest of academics. Thus, there does not seem to be any given topic more or less in fashion.

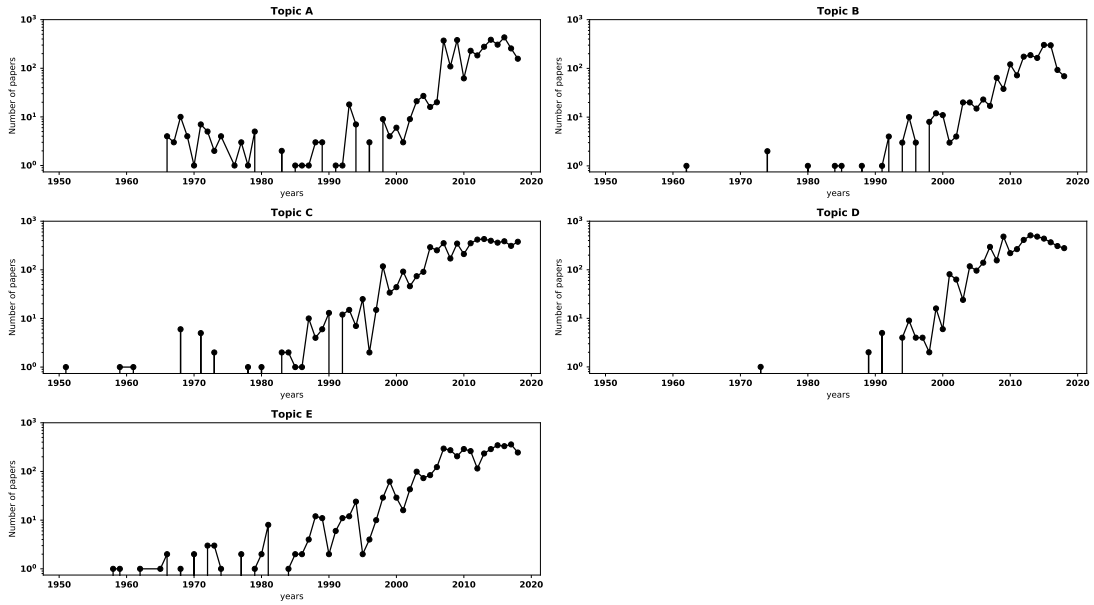


Figure 3.12: Number of articles per topic over the years (on a logged scale).

To gain a better understanding regarding the change in the topics over the years, LDA is applied to the cumulative data set over 8 time periods. These periods are 1951-1965,

1951-1973, 1951-1980, 1951-1988, 1951-1995, 1951-2003, 1951-2010, 1951-2018. The number of topics for each cumulative subset is chosen based on the coherence value and no objective approach is used. As a result, the period 1951-2018 has been assigned $n = 6$ which had the highest coherence value instead of 5. The chosen models for each period including the number of topics, their keywords and number of articles assigned to them are given by Table 3.4.

But how well do the five topics which were presented earlier fit the publications over time? This is answered by comparing the performance of three LDA models over the cumulative periods' publications. The three models are LDA models for the entire data set for n equal to 5, 6 and the optimal number of topics over time. For each model the c^* is estimated for each document in the cumulative data sets. The performance of the models are then compared based on:

$$\bar{c^*} \times n \tag{3.1}$$

where $\bar{c^*}$ is the median highest percentage contribution and n is the number of topics of a given period. A model with more topics will have more difficulty to assign papers. Thus, Equation (3.1) is a measure of confidence in assigning a given paper to its topic weighted by the number of topics. The performances are given by Figure 3.13.

The five topics of the PD presented in this manuscript appear to always be less good at fitting the publications compared to the six topics of LDA $n = 6$. Moreover, there are less good than the topics of the optimal number of topics from 1951 to 1995. The difference in the performance values, Equation (3.1), however are small. The relevance of the five topics has been increasing over time, and though, the topics did not always fit the majority of published work over time, there were still papers being published on those topics.

In the following section the collaborative behaviour of authors in the field, and within the field's topics as were presented in this section, are explored using a network theoretic approach.

3.5 Analysis of co-authorship network

The collaborative behaviour of authors in the field of the PD is assessed using the co-authorship network, which as introduced in section 3.2 is denoted as G. There are

Period	Topic	Topic Keywords	Num of Documents	Percentage of Documents
1951-1965	1	problem, technology, divert, euler, subsystem, requirement, trace, technique, system, untried	3	0.375
1951-1965	2	interpret, requirement, programme, evolution, article, increase, policy, system, trace, technology	2	0.25
1951-1965	3	equipment, agency, conjecture, development, untried, programme, trend, technology, weapon, technique	1	0.125
1951-1965	4	variation, celebrated, trend, untried, change, involve, month, technique, subsystem, research	1	0.125
1951-1965	5	give, good, modern, trace, technique, ambiguity, problem, trend, technology, system	1	0.125
1951-1973	1	study, shock, cooperative, money, part, vary, investigate, good, receive, equipment	12	0.3243
1951-1973	2	cooperation, level, significantly, sequence, reward, provoke, descriptive, principal, display, argue	4	0.1081
1951-1973	3	player, make, effect, triad, experimental, motivation, dominate, hypothesis, instruction, trend	3	0.0811
1951-1973	4	ss, sex, male, female, dyad, design, suggest, college, factor, tend	3	0.0811
1951-1973	5	result, research, format, change, operational, analysis, relate, understanding, decision, money	2	0.0541
1951-1973	6	condition, give, high, treatment, conflict, cc, real, original, replication, promote	2	0.0541
1951-1973	7	group, competitive, show, interpret, scale, compete, escalation, free, variable, individualistic	2	0.0541
1951-1973	8	outcome, strategy, choice, type, pdg, difference, dummy, conclude, compare, consistent	2	0.0541
1951-1973	9	game, difference, pair, approach, behavior, person, weapon, occur, advantaged, differential	2	0.0541
1951-1973	10	response, present, dilemma, influence, cooperate, bias, point, amount, participate, factor	2	0.0541
1951-1973	11	trial, problem, previous, involve, prisoner, experiment, follow, tit, increase, initial	1	0.027
1951-1973	12	matrix, behavior, rational, black, model, research, broad, distance, complex, trace	1	0.027
1951-1973	13	play, finding, individual, noncooperative, white, nature, race, ratio, represent, prisoner	1	0.027
1951-1980	1	play, trial, group, follow, white, interpret, scale, black, trend, small	14	0.25
1951-1980	2	outcome, level, effect, type, dyad, vary, pdg, participate, understanding, arise	9	0.1607
1951-1980	3	game, strategy, cooperation, significant, difference, sentence, text, occur, differential, hypothesis	4	0.0714
1951-1980	4	male, female, find, result, sex, subject, experimental, situation, treatment, computer	4	0.0714
1951-1980	5	research, problem, influence, matrix, format, model, analysis, year, crime, equipment	4	0.0714
1951-1980	6	condition, dilemma, bias, free, attempt, book, year, dummy, prison, design	4	0.0714
1951-1980	7	variable, result, factor, individual, ability, triad, half, migration, change, investigate	3	0.0536
1951-1980	8	show, present, suggest, rational, compete, approach, characteristic, examine, person, conduct	3	0.0536
1951-1980	9	behavior, high, finding, relate, obtain, assistance, ratio, good, weapon, competition	3	0.0536
1951-1980	10	ss, shock, money, competitive, part, difference, pair, amount, man, information	3	0.0536
1951-1980	11	player, conflict, theory, decision, determine, produce, maker, cooperate, specialist, programming	2	0.0357
1951-1980	12	study, prisoner, make, response, experiment, noncooperative, standard, separate, conclude, initial	2	0.0357
1951-1980	13	give, cooperative, choice, cognitive, real, operational, set, subject, ascribe, concern	1	0.0179
1951-1988	1	trial, difference, find, choice, significant, competitive, effect, triad, interact, occur	24	0.2553
1951-1988	2	ss, shock, money, pair, response, part, high, tit, receive, amount	13	0.1383
1951-1988	3	suggest, paper, case, debate, view, achieve, framework, natural, assumption, finitely	10	0.1064
1951-1988	4	prisoner, dilemma, behavior, model, present, involve, person, increase, trust, experiment	8	0.0851
1951-1988	5	game, player, show, approach, repeat, previous, move, tat, related, include	8	0.0851
1951-1988	6	cooperation, level, mutual, equilibrium, standard, provide, information, human, real, question	6	0.0638
1951-1988	7	play, result, male, subject, female, cooperative, sex, experimental, treatment, computer	5	0.0532
1951-1988	8	research, study, variable, ability, factor, conflict, matrix, year, student, interpret	4	0.0426
1951-1988	9	problem, group, small, scale, social, issue, large, base, bias, party	4	0.0426
1951-1988	10	game, strategy, outcome, type, cooperate, ethical, pdg, explain, dependent, separate	4	0.0426
1951-1988	11	give, condition, individual, major, dyad, behaviour, produce, conflict, assistance, collectively	3	0.0319
1951-1988	12	situation, iterate, statement, rational, card, side, paradox, true, consequence, front	2	0.0213
1951-1988	13	inflation, hypothesis, rate, run, change, demand, nominal, cost, output, growth	2	0.0213
1951-1988	14	theory, make, analysis, decision, system, examine, work, soft, lead, hard	1	0.0106
1951-1995	1	strategy, population, evolution, iterate, tit, opponent, evolve, dynamic, set, tat	31	0.1732
1951-1995	2	game, repeat, assumption, rule, person, equilibrium, general, finitely, indefinitely, analyze	24	0.1341
1951-1995	3	inflation, long, rate, hypothesis, run, policy, cost, nominal, demand, programming	20	0.1117
1951-1995	4	condition, outcome, trial, find, difference, cooperation, experiment, level, significant, response	15	0.0838
1951-1995	5	rational, result, receive, statement, money, paradox, shock, iterate, consequence, common	14	0.0782
1951-1995	6	cooperation, show, competitive, high, probability, conflict, simulation, altruism, yield, natural	14	0.0782
1951-1995	7	prisoner, dilemma, give, point, defect, form, cooperator, increase, relate, ethical	10	0.0559
1951-1995	8	player, give, decision, provide, cooperative, game, previous, pair, determine, interact	9	0.0503
1951-1995	9	play, cooperate, result, male, subject, female, time, relationship, suggest, student	8	0.0447
1951-1995	10	problem, group, theory, good, approach, society, large, scale, issue, level	8	0.0447
1951-1995	11	study, situation, behaviour, computer, argue, change, implication, characteristic, real, associate	8	0.0447
1951-1995	12	model, paper, behavior, examine, present, mutual, expectation, develop, type, variable	7	0.0391
1951-1995	13	make, research, system, analysis, choice, work, base, relation, world, wide	6	0.0335
1951-1995	14	individual, social, behavior, standard, choose, evolutionary, partner, payoff, defection, small	5	0.0279
1951-2003	1	game, player, dilemma, prisoner, theory, give, paper, make, group, problem	151	0.4266
1951-2003	2	cooperation, result, play, show, cooperate, condition, cooperative, high, level, time	106	0.2994
1951-2003	3	strategy, model, agent, study, behavior, individual, population, evolutionary, state, player	97	0.274
1951-2010	1	model, theory, paper, base, make, present, problem, provide, human, decision	325	0.3454
1951-2010	2	game, strategy, player, agent, play, dilemma, system, behavior, show, state	322	0.3422
1951-2010	3	cooperation, network, study, population, individual, evolutionary, social, evolution, interaction, structure	294	0.3124
1951-2018	1	model, theory, system, base, paper, problem, propose, present, approach, provide	556	0.2251
1951-2018	2	behavior, social, human, decision, study, experiment, make, suggest, result, behaviour	482	0.1951
1951-2018	3	individual, group, good, social, punishment, level, cost, mechanism, dilemma, cooperative	428	0.1733
1951-2018	4	game, strategy, player, agent, play, dilemma, state, prisoner, payoff, equilibrium	380	0.1538
1951-2018	5	population, evolutionary, dynamic, model, selection, result, evolution, evolve, show, process	351	0.1421
1951-2018	6	cooperation, network, interaction, structure, study, evolution, find, behavior, cooperative, simulation	273	0.1105

Table 3.4: Topic modelling result for the cumulative data set over the periods

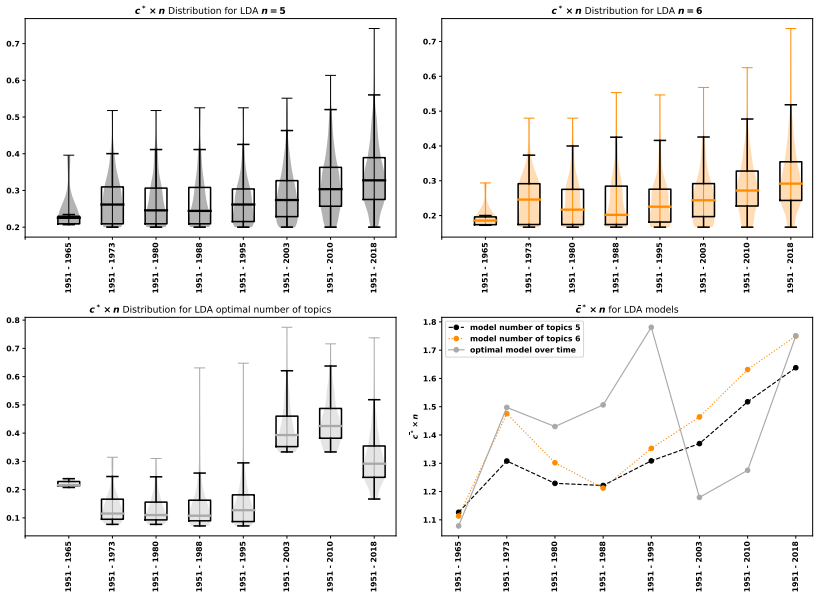


Figure 3.13: Maximum percentage contributions (c^*) over the time periods, for the LDA models for the entire data set for n equal to 5, 6 and the optimal number of topics over time.

a total of 947 connected components in G and the largest component has a size of 796 nodes. The largest connected component is going to be referred to as the main cluster of the network and is denoted as $\bar{G}$. A graphical representation of both networks is shown in Figures 3.14-3.15 and a metrics summary is given by Table 3.5.

Based on Table 3.5 an author in G has on average 4 collaborators and a 70% probability of collaborating with a collaborator's co-author. An author of $\bar{G}$ on average is 7% more likely to write with a collaborator's co-author and on average has 2 more collaborators. Moreover, there are only 3.2 % of authors in the PD that has no connection to any other author.

	# Nodes	# Edges	% Isolated nodes	# Connected components	Size of largest component	Av. degree	# Communities	Modularity	Clustering coeff
G	4011	7642	3.2	947	796	3.811	967	0.96491	0.701
$\bar{G}$	796	2214	0.0	1	796	5.563	25	0.84406	0.773

Table 3.5: Network metrics for G and $\bar{G}$ respectively.

But how do these compare to other fields? Two more data sets for the topics "price of anarchy" and "auction Games" have been collected in order to compare the collaborative behaviour of the PD to other game theoretic fields. A total of 3,444 publications have been collected for auction games and 748 for price of anarchy. Price of anarchy is relatively a new field, with the first publication on the topic being [167] in 1999. This explains the small number of articles that have been retrieved. Both data sets have

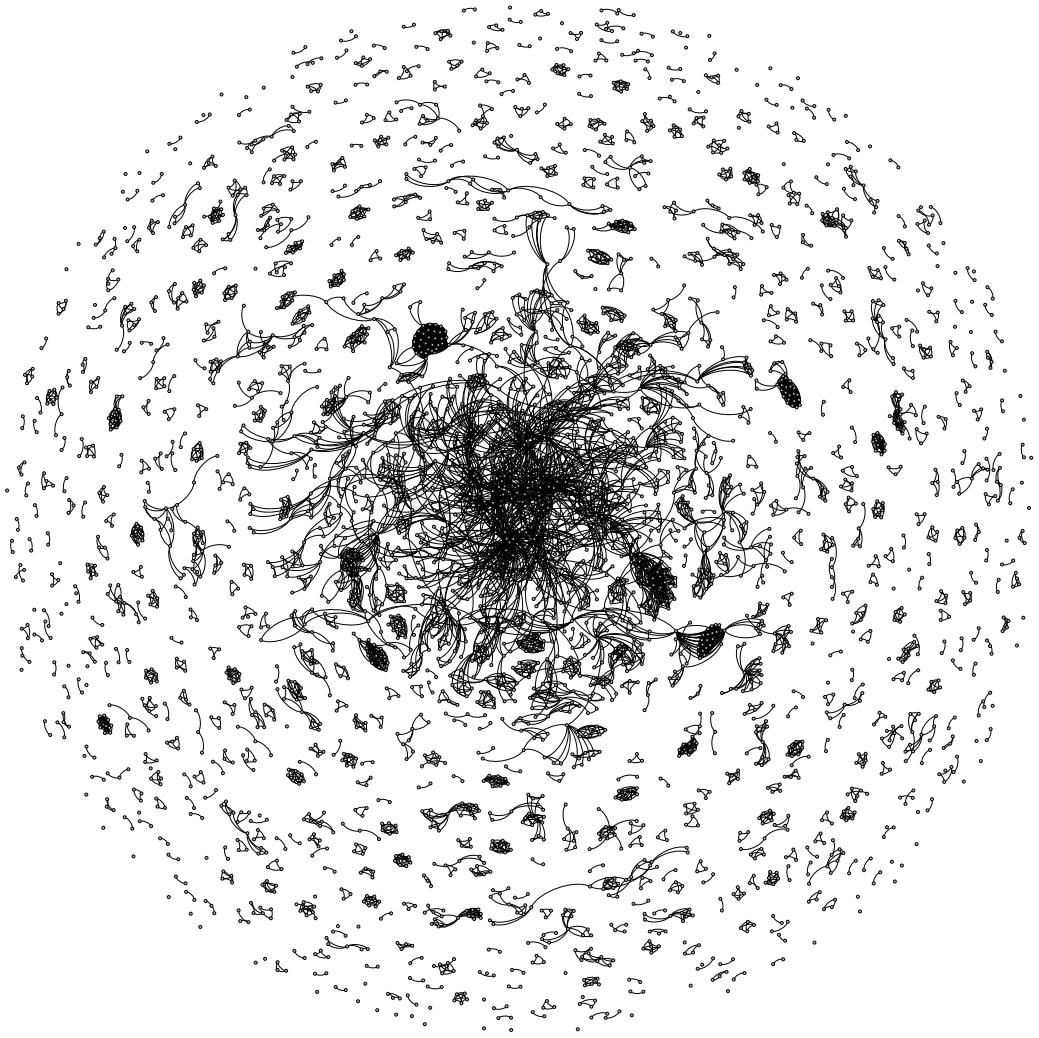


Figure 3.14: G the co-authorship network for the IPD.

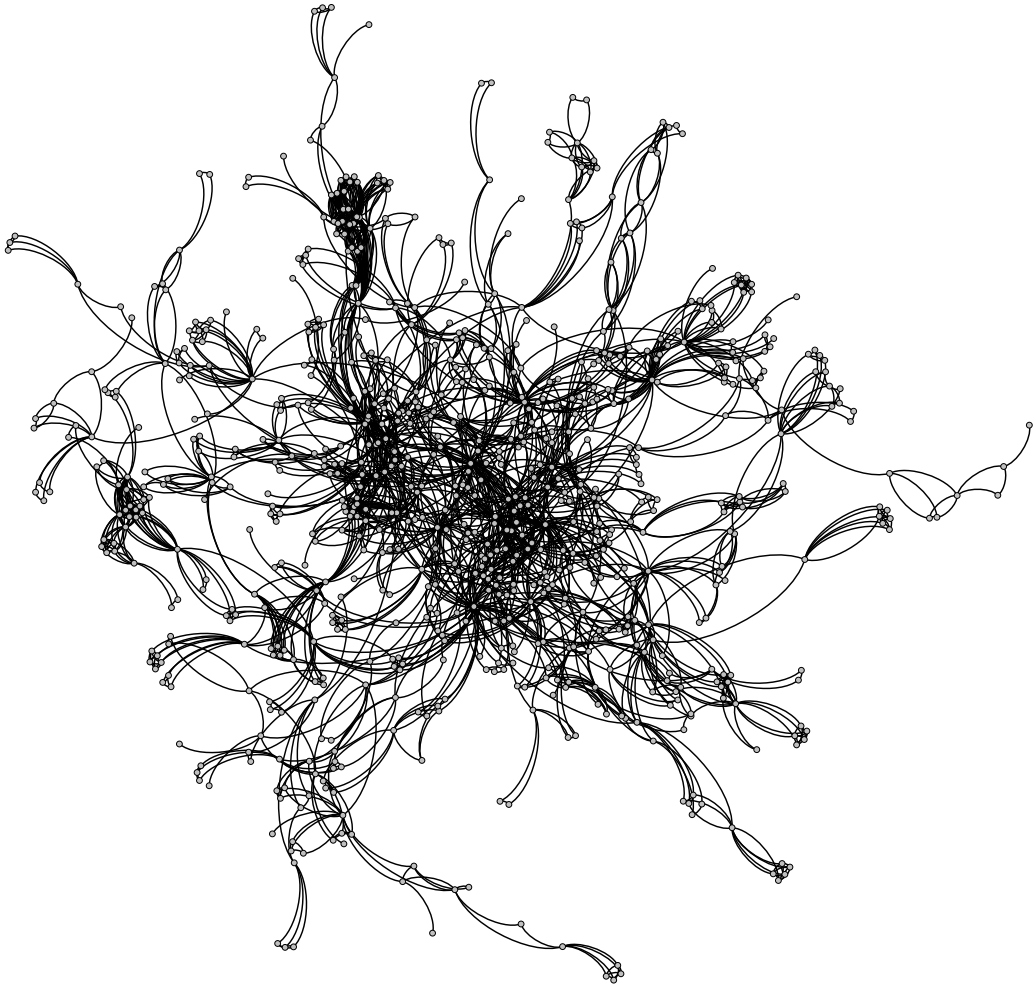


Figure 3.15: $\bar{G}$ the largest connected component of G.

been archived and are available in [98, 99]. The networks for both data sets have been generated in the same way as G. A summary of the networks' metrics are given by Table 3.6.

	# Nodes	# Edges	# Isolated nodes	% Isolated nodes	# Connected components	Size of largest component	Av. degree	# Communities	Modularity	Clustering coeff
auction games	5165	7861	256	5.0	1272	1348	3.044	1294	0.957	0.622
price of anarchy	1155	1953	4	0.3	245	222	3.382	253	0.965	0.712

Table 3.6: Network metrics for auction games and price of anarchy networks respectively.

The average degrees for the price of anarchy and for auction games are lower than the PD's. In auction games an author is more likely to have no collaborators, and in the price of anarchy there are almost no authors that are not connected to someone. This could be an effect of the field being introduced in more modern days. Overall, an author in the PD has on average more collaborators and there are less isolated authors compared to another well established game theoretic field. These results seem to indicate that the PD is a *relatively* collaborative field.

However, both G and $\bar{G}$ have a high modularity (larger than 0.84) and a large number of communities (967 and 25 respectively). A high modularity implies that authors create their own publishing communities but not many publications from authors from different communities occur. Thus, author tends to collaborate with authors in their communities but not many efforts are made to create new connections to other communities and spread the knowledge of the field across academic teams. The fields of both price of anarchy and auction games also have high modularity, and that could indicate that is in fact how academic publications are. Thus, the PD is indeed a collaborative field but perhaps it is not more collaborative than other fields, as there is no effort from the authors to write with people outside their community.

The evolution of the networks was also explored over time by constructing the network cumulatively over 51 periods. Except from the first period 1951-1966 the rest of the periods have a yearly interval (data for the years 1975 and 1982 were not retrieved by the collection data process). The metrics of each sub network are given by Tables 3.7 and 3.8.

The results, similarly to the results of [183], confirm that the networks grow over time and that the networks always had a high modularity. Since the first publications authors tend to write with people from their communities, and that is not an effect of

Period	# Nodes	# Edges	% Isolated nodes	# Connected components	Size of largest component	Av. degree	# Communities	Modularity	Clustering coeff
1951 - 1966	6	3	0.0	3	2	1.000	3	0.667	0.000
1951 - 1967	8	4	0.0	4	2	1.000	4	0.750	0.000
1951 - 1968	19	15	0.0	8	5	1.579	8	0.684	0.228
1951 - 1969	20	17	0.0	8	6	1.700	8	0.630	0.250
1951 - 1970	22	18	0.0	9	6	1.636	9	0.667	0.227
1951 - 1971	33	28	0.0	13	6	1.697	13	0.827	0.424
1951 - 1972	39	34	0.0	15	6	1.744	15	0.867	0.513
1951 - 1973	42	35	2.4	17	6	1.667	17	0.873	0.476
1951 - 1974	42	35	2.4	17	6	1.667	17	0.873	0.476
1951 - 1976	42	35	2.4	17	6	1.667	17	0.873	0.476
1951 - 1977	44	36	2.3	18	6	1.636	18	0.880	0.455
1951 - 1978	44	36	2.3	18	6	1.636	18	0.880	0.455
1951 - 1979	47	40	2.1	18	6	1.702	18	0.884	0.454
1951 - 1980	47	40	2.1	18	6	1.702	18	0.884	0.454
1951 - 1981	50	46	2.0	18	6	1.840	18	0.889	0.497
1951 - 1983	51	46	3.9	19	6	1.804	19	0.889	0.487
1951 - 1984	53	47	3.8	20	6	1.774	20	0.894	0.469
1951 - 1985	53	47	3.8	20	6	1.774	20	0.894	0.469
1951 - 1986	53	47	3.8	20	6	1.774	20	0.894	0.469
1951 - 1987	56	48	5.4	22	6	1.714	22	0.898	0.443
1951 - 1988	62	52	6.5	25	6	1.677	25	0.909	0.449
1951 - 1989	75	62	6.7	31	6	1.653	31	0.926	0.424
1951 - 1990	79	64	6.3	33	6	1.620	33	0.930	0.403
1951 - 1991	87	69	6.9	37	6	1.586	37	0.937	0.400
1951 - 1992	95	72	10.5	42	6	1.516	42	0.941	0.367
1951 - 1993	106	81	11.3	47	6	1.528	47	0.947	0.366
1951 - 1994	124	95	12.9	56	6	1.532	56	0.955	0.394
1951 - 1995	135	102	12.6	61	6	1.511	61	0.960	0.384
1951 - 1996	142	105	12.7	65	6	1.479	65	0.962	0.365
1951 - 1997	155	115	12.9	71	6	1.484	71	0.966	0.392
1951 - 1998	191	140	11.0	87	6	1.466	87	0.973	0.367
1951 - 1999	221	169	11.3	99	6	1.529	99	0.977	0.397
1951 - 2000	250	195	10.8	110	6	1.560	110	0.979	0.418
1951 - 2001	287	235	10.5	125	7	1.638	125	0.977	0.419
1951 - 2002	335	278	10.7	146	7	1.660	146	0.979	0.428
1951 - 2003	381	310	10.5	168	7	1.627	168	0.982	0.413
1951 - 2004	437	370	9.2	185	10	1.693	185	0.983	0.424
1951 - 2005	532	476	7.7	214	19	1.789	214	0.985	0.458
1951 - 2006	640	603	6.7	246	22	1.884	246	0.987	0.486
1951 - 2007	793	877	5.8	283	25	2.212	283	0.985	0.532
1951 - 2008	948	1170	5.3	318	33	2.468	319	0.985	0.558
1951 - 2009	1108	1442	4.9	356	71	2.603	358	0.982	0.573
1951 - 2010	1300	1936	5.1	402	133	2.978	405	0.965	0.592
1951 - 2011	1560	2375	5.1	472	157	3.045	475	0.970	0.613
1951 - 2012	1837	2865	4.4	534	209	3.119	537	0.969	0.634
1951 - 2013	2149	3420	4.3	603	322	3.183	609	0.965	0.644
1951 - 2014	2481	3971	4.2	683	399	3.201	694	0.962	0.658
1951 - 2015	2938	4877	3.7	765	504	3.320	779	0.965	0.675
1951 - 2016	3469	6532	3.3	850	613	3.766	863	0.964	0.696
1951 - 2017	3735	7072	3.2	895	706	3.787	912	0.964	0.700
1951 - 2018	4011	7642	3.2	947	796	3.811	967	0.966	0.701

Table 3.7: Collaborativeness metrics for cumulative graphs, $\tilde{G} \subseteq G$

Period	# Nodes	# Edges	% Isolated nodes	Size of largest component	Av. degree	# Communities	Modularity	Clustering coeff
1951 - 1966	2	1	0.0	2	1.000	1	0.000	0.000
1951 - 1967	2	1	0.0	2	1.000	1	0.000	0.000
1951 - 1968	5	8	0.0	5	3.200	1	0.000	0.867
1951 - 2002	7	21	0.0	7	6.000	1	0.000	1.000
1951 - 2003	7	21	0.0	7	6.000	1	0.000	1.000
1951 - 2004	10	13	0.0	10	2.600	2	0.376	0.553
1951 - 2005	19	28	0.0	19	2.947	3	0.544	0.730
1951 - 2006	22	35	0.0	22	3.182	4	0.527	0.720
1951 - 2007	25	39	0.0	25	3.120	5	0.558	0.686
1951 - 2008	33	62	0.0	33	3.758	4	0.623	0.736
1951 - 2009	71	148	0.0	71	4.169	6	0.697	0.698
1951 - 2010	133	387	0.0	133	5.820	7	0.726	0.749
1951 - 2011	157	465	0.0	157	5.924	8	0.727	0.725
1951 - 2012	209	611	0.0	209	5.847	11	0.733	0.737
1951 - 2013	322	892	0.0	322	5.540	12	0.780	0.743
1951 - 2014	399	1109	0.0	399	5.559	15	0.794	0.742
1951 - 2015	504	1368	0.0	504	5.429	24	0.811	0.751
1951 - 2016	613	1677	0.0	613	5.471	21	0.819	0.761
1951 - 2017	706	1935	0.0	706	5.482	29	0.830	0.772
1951 - 2018	796	2214	0.0	796	5.563	25	0.845	0.773

Table 3.8: Collaborativeness metrics for cumulative graphs' main clusters, $\tilde{G} \subseteq \bar{G}$

a specific time period.

The networks corresponding to the topics of section 3.3 have also been generated similarly to G. Note that authors with publications in more than one topic exist, and these authors are included in all the corresponding networks. A metrics' summary for all five topic networks is given by Table 3.9.

	# Nodes	# Edges	# Isolated nodes	% Isolated nodes	# Connected components	Size of largest component	Av. degree	# Communities	Modularity	Clustering coeff
Topic A	1124	2137	15	1.3	264	56	3.802	265	0.983	0.759
Topic B	695	1382	13	1.9	157	80	3.977	158	0.950	0.773
Topic C	900	1141	41	4.6	281	29	2.536	281	0.981	0.636
Topic D	880	1509	17	1.9	174	312	3.430	183	0.918	0.701
Topic E	1045	1964	59	5.6	354	31	3.759	354	0.926	0.664

Table 3.9: Network metrics for topic networks.

Topic B is the network with the highest average degree followed by Topic A. The topic with the smallest average degree, 2.5, is Topic C. In topics A and B the number of isolated nodes is very small $< (0.2)$ compared to Topic E where the percentage of isolated nodes is approximately 6%. Moreover, in topics C and E an author is 10% more likely to collaborate with a collaborator's co-author. Thus, topics "human subject research" and "biological studies" tend to be more collaborative than the topic of "strategies", and an authors in these are less likely to have at least one collaborator compared to the topic of "modelling problems as a PD".

"Evolutionary dynamics on networks" also appear to be a collaborative topic. In fact the network of the topic is a sub graph of $\bar{G}$, the main cluster of G and it will be demonstrated in the following section that authors in this network are more like to gain from the influence of the network compared to any other topic network.

The two centrality measures reported in this thesis are closeness and betweenness centrality. Closeness centrality is a measure of how easy it is for an author to contact others, and consequently affect them; influence them. Thus closeness centrality is a measure of influence. Betweenness centrality is a measure of how many paths pass through a specific node, thus the amount of information this person has access to. Betweenness centrality is interpreted as a measure of how much an author gains from the field. The values of the centralities can range between 0 and 1. Influence and the amount of information an author has access to are proxies to understand if/which authors benefit more from their position.

For G and $\bar{G}$ the most central authors based on closeness and betweenness centralities are given by Table 3.10. The most central authors in G and $\bar{G}$ are the same. This implies that the results on centrality heavily rely on the main cluster (as expected). Matjaz Perc is an author with 83 publications in the data set and the most central authors based on both centrality measures. The most central authors are fairly similar between the two measures. The author that appear to be central based on one measure and not the other are Martin Nowak, Franz Weissing, Jianye Hao, Angel Sanchez and Valerio Capraro which have access to information due to their positioning but do not influence the network as much, and the opposite is true for Attila Szolnoki, Luo-Luo Jiang Sandro Meloni, Cheng-Yi Xia and Xiaojie Chen.

	G				$\bar{G}$			
	Name	Betweenness	Name	Closeness	Name	Betweenness	Name	Closeness
1	Matjaz Perc	0.015	Matjaz Perc	0.066	Matjaz Perc	0.373	Matjaz Perc	0.330
2	Zhen Wang	0.011	Long Wang	0.060	Zhen Wang	0.279	Long Wang	0.301
3	Long Wang	0.007	Yamir Moreno	0.059	Long Wang	0.170	Yamir Moreno	0.299
4	Martin Nowak	0.006	Attila Szolnoki	0.059	Martin Nowak	0.159	Attila Szolnoki	0.297
5	Angel Sanchez	0.004	Zhen Wang	0.059	Angel Sanchez	0.114	Zhen Wang	0.296
6	Yamir Moreno	0.004	Arne Traulsen	0.056	Yamir Moreno	0.110	Arne Traulsen	0.281
7	Arne Traulsen	0.004	Luo-Luo Jiang	0.055	Arne Traulsen	0.107	Luo-Luo Jiang	0.280
8	Franz Weissing	0.004	Sandro Meloni	0.055	Franz Weissing	0.101	Sandro Meloni	0.278
9	Jianye Hao	0.004	Cheng-Yi Xia	0.055	Jianye Hao	0.094	Cheng-Yi Xia	0.276
10	Valerio Capraro	0.004	Xiaojie Chen	0.055	Valerio Capraro	0.093	Xiaojie Chen	0.276

Table 3.10: The 10 most central authors based on betweenness and closeness centralities for G and $\bar{G}$.

It is obvious that in G the centrality values are low which suggests that in the PD authors do not benefit from their positions. This could be an effect of information not flowing from one community to another as authors tend to write with people from their communities. Nevertheless, there are authors that do benefit from their position, but these are only the authors connected to the main cluster.

The centrality measures for the topic networks have also been estimated and are given in Tables 3.11-3.12. If information was flowing between the communities of the research topics then there would be an increase to the values of centralities for the sub networks. However, the only topic where authors gain from their positions are the authors of Topic D (topic on evolutionary dynamics on network). From the list of names it is obvious that these authors are part of $\bar{G}$, and that the network of Topic D is a sub network of $\bar{G}$. This confirms the results. The people benefiting from their position in the co-authorship networks corresponding to research topics of the PD are only the people from the main cluster of G.

	Topic A		Topic B		Topic C		Topic D		Topic E	
	Name	Betweeness	Name	Betweeness	Name	Betweeness	Name	Betweeness	Name	Betweeness
1	David Rand	0.002	Long Wang	0.006	Daniel Ashlock	0.001	Matjaz Perc	0.064	Zengru Di	0.0
2	Valerio Capraro	0.001	Luo-Luo Jiang	0.005	Matjaz Perc	0.000	Luo-Luo Jiang	0.037	Jian Yang	0.0
3	Angel Sanchez	0.001	Martin Nowak	0.004	Karl Tuyls	0.000	Yamir Moreno	0.031	Yevgeniy Vorobeychik	0.0
4	Feng Fu	0.001	Matjaz Perc	0.003	Philip Hingston	0.000	Christoph Hauert	0.027	Otavio Teixeira	0.0
5	Martin Nowak	0.000	Attila Szolnoki	0.003	Eun-Youn Kim	0.000	Long Wang	0.024	Roberto Oliveira	0.0
6	Nicholas Christakis	0.000	Christian Hilbe	0.002	Wendy Ashlock	0.000	Zhen Wang	0.024	M. Nowak	0.0
7	Pablo Branas-Garza	0.000	Yamir Moreno	0.002	Attila Szolnoki	0.000	Han-Xin Yang	0.023	M. Harper	0.0
8	Toshio Yamagishi	0.000	Xiaojie Chen	0.002	Seung Baek	0.000	Martin Nowak	0.020	Xiao Han	0.0
9	James Fowler	0.000	Arne Traulsen	0.002	Martin Nowak	0.000	Angel Sanchez	0.017	Zhesi Shen	0.0
10	Long Wang	0.000	Zhen Wang	0.002	Thore Graepel	0.000	Zhihai Rong	0.016	Wen-Xu Wang	0.0

Table 3.11: The 10 most central authors based on betweenness centrality for topics' networks.

	Topic A		Topic B		Topic C		Topic D		Topic E	
	Name	Closeness	Name	Closeness	Name	Closeness	Name	Closeness	Name	Closeness
1	David Rand	0.027	Long Wang	0.043	Karl Tuyls	0.022	Matjaz Perc	0.123	Stefanie Widder	0.029
2	Valerio Capraro	0.023	Matjaz Perc	0.041	Thore Graepel	0.019	Zhen Wang	0.109	Rosalind Allen	0.029
3	Jillian Jordan	0.022	Attila Szolnoki	0.040	Joel Leibo	0.018	Long Wang	0.107	Thomas Pfeiffer	0.029
4	Nicholas Christakis	0.021	Martin Nowak	0.040	Edward Hughes	0.017	Yamir Moreno	0.105	Thomas Curtis	0.029
5	James Fowler	0.020	Olivier Tenaillon	0.038	Matthew Phillips	0.017	Luo-Luo Jiang	0.104	Carsten Wiuf	0.029
6	Martin Nowak	0.020	Xiaojie Chen	0.038	Edgar Duenez-Guzman	0.017	Attila Szolnoki	0.103	William Sloan	0.029
7	Angel Sanchez	0.019	Bin Wu	0.038	Antonio Castaneda	0.017	Gyorgy Szabo	0.102	Otto Cordero	0.029
8	Gordon Kraft-Todd	0.019	Yanling Zhang	0.037	Iain Dunning	0.017	Xiaojie Chen	0.102	Sam Brown	0.029
9	Akihiro Nishi	0.019	Feng Fu	0.037	Tina Zhu	0.017	Guangming Xie	0.101	Babak Momeni	0.029
10	Anthony Evans	0.019	David Rand	0.037	Kevin Mckee	0.017	Lucas Wardil	0.101	Wenying Shou	0.029

Table 3.12: The 10 most central authors based on closeness centrality for topics' networks.

The fact that most authors of the main cluster are primarily publishing in evolutionary dynamics on networks indicates that publishing in this specific topic differs from the other topics covered in this manuscript. There appears to be more collaboration and

influence in the publications on evolutionary dynamics and authors are more likely to gain from their position, though it is not clear as to why.

The distributions of both centrality measures for all the networks are given in the Appendix A.2.

3.6 Chapter summary

This Chapter explored the research topics from a collection of 2,422 publications of the IPD, and moreover, the authors' collaborative behaviour and their influence in the research field. This was achieved by applying network theoretic approaches and a LDA algorithm to the collection of publications. Both the software [107] and the main data set [100] associated with the Chapter have been archived and are available to be used by other researchers. In fact Arcas has been used by [189] and [280].

Arcas, its development and the data collection were covered in section 3.2, as well as an introduction to the co-authorship network and to LDA. Section 3.3 covered an initial analysis of the data set which demonstrated that the PD is a field that continues to attract academic attention and publications. In section 3.4 LDA was applied to the data set to identify topics on which researchers have been publishing. The LDA analysis showed that the articles could be classified into 5 topics associated with human subject research, biological studies, strategies, evolutionary dynamics on networks and modelling problems as a PD. These topics summarise the field of the PD well, as they demonstrate its interdisciplinarity and applications to a variety of problems. A temporal analysis explored how relevant these topics have been over the course of time, and it revealed that even though there were not the necessarily always the most discussed topics they were still being explored by researchers.

The collaborative behaviour of the field was explored in section 3.5 by constructing the co-authorship network. It was concluded that the field is a collaborative field, where authors are likely to write with a collaborator's co-authors and on average an author has 4 co-authors, however, it not necessarily more collaborative than other fields. The authors tend to collaborate with authors from one community, but not many authors are involved in multiple communities. This however might be an effect of academic research, and it might not be true just for the field of the PD. Exploring the influence of authors and their gain from being in the network of the field demonstrated that authors do not gain much, and the authors with influence are only the ones connected

to the main cluster, to a "main" group of authors. This 'main" group of authors consists of authors publishing in evolutionary dynamics on networks. Thus, an author would be aiming to publish on this topic if they were interested in gaining from their position in the publications of the PD.

The study of the PD is the study of cooperation and investigating the cooperative behaviours of authors is what this Chapter has aimed to achieve. The following Chapters focus on best responses in changing environments of the PD, and more specifically Chapter 4 studies best responses from a collection of strategies in a large number of IPD tournaments.

Chapter 4

A meta analysis of tournaments and an evaluation of performance in the Iterated Prisoner's Dilemma.

The research reported in this Chapter has lead to a manuscript, entitled:

"Properties of winning Iterated Prisoner's Dilemma strategies"

Available at: `https://arxiv.org/abs/2001.05911`

Associated data sets: [97, 102]

Associated codebase: [96]

Axelrod-Python library (APL) version: 3.0.0

The manuscript's abstract is the following:

Researchers have explored the performance of Iterated Prisoner's Dilemma strategies for decades: from the celebrated performance of Tit For Tat, to the introduction of the zero-determinant strategies, to the use of sophisticated learning structures such as neural networks, many new strategies have been introduced and tested in a variety of tournaments and population dynamics. Typical results in the literature, however, rely on performance against a small number of somewhat arbitrarily selected strategies in a small number of tournaments, casting doubt on the generalisability of conclusions. We analyse a large collection of 195 strategies in 45,600 tournaments, present the top per-

forming strategies across multiple tournament types, and distill their salient features. The results show that there is not yet a single strategy that performs well in diverse Iterated Prisoner's Dilemma scenarios, nevertheless there are several properties that heavily influence the best performing strategies. This refines the properties described by R. Axelrod in light of recent and more diverse opponent populations to: be nice, be provocable and generous, be a little envious, be clever, and adapt to the environment. More precisely, we find that strategies perform best when their probability of cooperation matches the total tournament population's aggregate cooperation probabilities, or a proportion thereof in the case of noisy and probabilistically ending tournaments, and that the manner in which a strategy achieves the ideal cooperation rate is crucial. The features of high performing strategies help cast some light on why strategies such as Tit For Tat performed historically well in tournaments and why zero-determinant strategies typically do not fare well in tournament settings.

The differences between the Chapter and the manuscript include the introduction to the PD and the discussion of previous literature. Both the introduction to the PD and the discussion of previous tournaments and strategies are excluded from the Chapter. That is because the introduction to the PD was presented in Chapter 1 and the previous literature was discussed in Chapter 2.

4.1 Introduction

In Chapter 1 it was discussed that conceptualising strategies and understanding the best way of playing the game has been of interest to the scientific community since the formulation of the game. In Chapter 2 it was established that following the computer tournaments of Axelrod in the 1980's a strategy's performance in a round robin computer tournament became a common evaluation technique for newly designed strategies. A large collection of works were discussed in Chapter 2 which introduced a broad collection of strategies, and new strategies and competitions are published frequently as established in Chapter 3. The question, however, still remains the same: what is the best way to play the game?

Compared to the works reviewed in Chapter 2, where typically a few selected or introduced strategies are evaluated on a small number of tournaments and/or small number of opponents, this Chapter evaluates the performance of 195 strategies in 45,600 computer tournaments. Furthermore, a large portion of these strategies are drawn from the known and named strategies in IPD literature, including many previous tournament winners, in contrast to other work that may have randomly generated many essentially arbitrary strategies (typically restrained to a class such as memory-one strategies, or those of a certain structural form such as finite state machines or deterministic memory two strategies). Additionally, the analysis of this Chapter considers tournament variations including standard tournaments, tournaments with noise, probabilistic match length, and both noise and probabilistic match length. This diversity of strategies and tournament types yields new insights and tests earlier claims in alternative settings against known powerful strategies.

The later part of the Chapter evaluates the impact of features on the performance of the strategies using modern machine learning and statistical techniques. These features include measures regarding a strategy's behaviour as well as measures regarding the tournaments. The outcomes reinforce the discussion started by Axelrod on properties of successful strategies (presented in section 2.3), and conclude that the properties are:

- ~~Do not be envious~~ Be a little bit envious
- Be "nice" in non-noisy environments or when game lengths are longer
- Reciprocate both cooperation and defection appropriately; Be provocable in tour-

naments with short matches, and generous when matches are longer

- ~~Do not be too clever~~ It is okay to be clever
- Adapt to the environment; Adjust to the mean population cooperation rate

The rest of the Chapter is structured as follows:

- section 4.2 covers the different tournament types and the data collection which are made possible due to APL.
- section 4.3 focuses on the best performing strategies for each type of tournament and overall.
- section 4.4 explores the traits which contribute to a good performance.

4.2 Data collection

The data set generated for this Chapter was created with APL version 3.0.0. APL allows for different types of IPD computer tournaments to be simulated and contains a large list of strategies. Most of these are strategies described in the literature with a few exceptions of strategies that have been contributed specifically to the package. A total of 195 strategies are used in this Chapter, a list of these is given in the Appendix B.1.

Although APL features several tournament types, only standard, noisy, probabilistic ending, and noisy probabilistic ending tournaments are considered here. *Standard tournaments* are tournaments similar to that of Axelrod's tournaments [32]. There are N strategies which all play an iterated game of n number of turns against each other. Note that self-interactions are not included. Similarly, *noisy tournaments* have N strategies and n number of turns, but at each turn there is a probability p_n that a player's action will be flipped. *Probabilistic ending tournaments*, are of size N and after each turn a match between strategies ends with a given probability p_e. Finally, *noisy probabilistic ending* tournaments have both a noise probability p_n and an ending probability p_e. For smoothing the simulated results a tournament is repeated for k number of times. This was allowed to vary in order to evaluate the effect of smoothing. The winner of each tournament is based on the median score a strategy achieved and not by the number of wins.

The process of collecting tournament results is described by Algorithm 4.1.

Algorithm 4.1: Tournaments' result summary collections algorithm

foreach *seed* $\in [0, 11, 400]$ **do**

$N \leftarrow$ randomly select integer $\in [N_{min}, N_{max}]$;

players $\leftarrow$ randomly select N players;

$k \leftarrow$ randomly select integer $\in [k_{min}, k_{max}]$;

$n \leftarrow$ randomly select integer $\in [n_{min}, n_{max}]$;

$p_n \leftarrow$ randomly select float $\in [p_{n\,min}, p_{n\,max}]$;

$p_e \leftarrow$ randomly select float $\in [p_{e\,min}, p_{e\,max}]$;

result standard $\leftarrow$ Axelrod.tournament(players, n, k);

result noisy $\leftarrow$ Axelrod.tournament(players, n, p_n, k);

result probabilistic ending $\leftarrow$ Axelrod.tournament(players, p_e, k);

result noisy probabilistic ending $\leftarrow$ Axelrod.tournament(players, p_n, p_e, k);

return *result standard, result noisy, result probabilistic ending, result noisy probabilistic ending*;

For each trial a random size N is selected, and from the 195 strategies a random list of N strategies is chosen. For the given list of strategies a standard, a noisy, a probabilistic ending and a noisy probabilistic ending tournament are performed and repeated k times. The parameters for the tournaments, as well as the number of repetitions, are selected once for each trial. The parameters and their respective minimum and maximum values are given by Table 4.1.

parameter	parameter explanation	min value	max value
N	number of strategies	3	195
k	number of repetitions	10	100
n	number of turns	1	200
p_n	probability of flipping action at each turn	0	1
p_e	probability of match ending in the next turn	0	1

Table 4.1: Data collection; parameter values.

A total of 11,400 trials of Algorithm 4.1 have been run. For each trial the results for 4 different tournaments were collected, thus a total of 45,600 $(11,400 \times 4)$ tournament results have been retrieved. Each tournament outputs a result summary in the form of Table 4.2. These have been archived and are available at [97, 102]. Each strategy has participated on average in 5,154 tournaments of each type. The strategy with the maximum participation in each tournament type is Inverse Punisher with 5,639 entries. The strategy with the minimum entries is EvolvedLookerUp 1 1 1 which was

selected in 4,693 trials.

A result summary (Table 4.2) has N rows because each row contains information for each strategy that participated in the tournament. The information includes the strategy's rank, median score, the rate with which the strategy cooperated (C_r), its match win count, and the probability that the strategy cooperated in the opening move. Moreover, the probabilities of a strategy being in any of the four states (CC, CD, DC, DD), and the rate of which the strategy cooperated after each state. The *normalised rank* is a feature that has been manually added to the result summary. The rank R of a given strategy can vary between 0 (first) and $N-1$ (last), and thus the normalised rank, denoted as r, is calculated as a strategy's rank divided by $N-1$.

						Rates							
Rank	Name	Median score	Cooperation rating (C_r)	Win	Initial C	CC	CD	DC	DD	CC to C	CD to C	DC to C	DD to C
0	EvolvedLookerUp2 2 2	2.97	0.705	28.0	1.0	0.639	0.066	0.189	0.106	0.836	0.481	0.568	0.8
1	Evolved FSMSix 16 Noise 05	2.875	0.697	21.0	1.0	0.676	0.020	0.135	0.168	0.985	0.571	0.392	0.07
2	PSO Gambler 1 1 1	2.874	0.684	23.0	1.0	0.651	0.034	0.152	0.164	1.000	0.283	0.000	0.136
3	PSO Gambler Mem1	2.861	0.706	23.0	1.0	0.663	0.042	0.145	0.150	1.000	0.510	0.000	0.122
4	Winner12	2.835	0.682	20.0	1.0	0.651	0.031	0.141	0.177	1.000	0.441	0.000	0.462
...	...	...	...	...	...	...	...	...	...	...	...	...	...

Table 4.2: Output result of a single tournament.

4.3 Top ranked strategies

The performance of each strategy is evaluated in four tournament types, as presented in section 4.2, followed by an evaluation of their performance over all the 45,600 simulated tournaments. Each strategy participated in multiple tournaments of the same type (on average 5,154). For example Tit For Tat participated in a total of 5,114 tournaments of each type. The strategy's normalised rank distribution in these is given in Figure 4.1. A value of $r = 0$ corresponds to a strategy winning the tournament where a value of $r = 1$ corresponds to the strategy coming last. Because of the strategies' multiple entries their performance is evaluated based on the *median normalised rank* denoted as $\bar{r}$.

The top 15 strategies for each tournament type based on $\bar{r}$ are given in Table 4.3. The data collection process was designed such that the probabilities of noise and ending of the match varied between 0 and 1. However, commonly used values for these probabilities are values less than 0.1. Thus, Table 4.3 also includes the top 15 strategies in noisy tournaments with $p_n < 0.1$ and probabilistic ending tournaments with $p_e < 0.1$. The r distributions for the top ranked strategies of Table 4.3 are given by Figure 4.2.

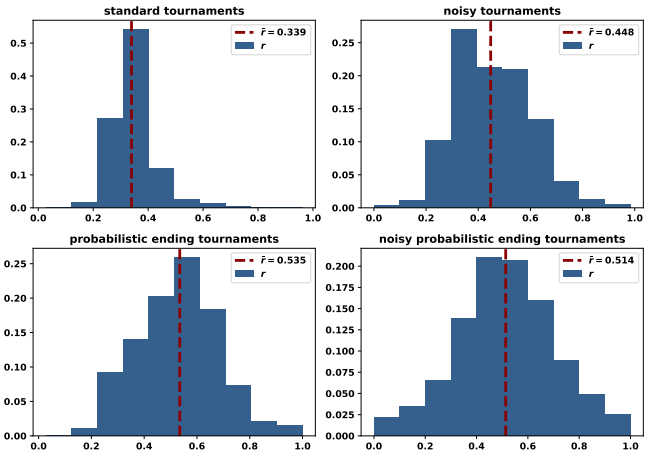
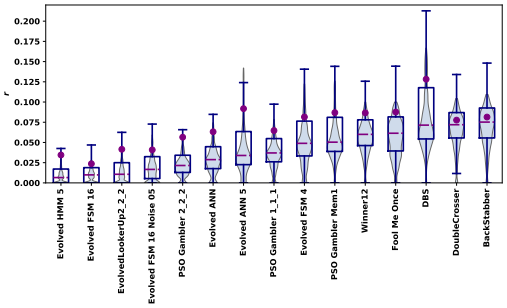


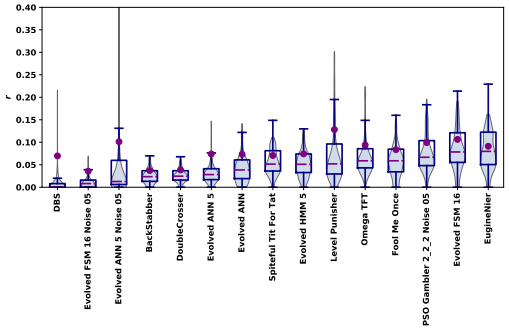
Figure 4.1: Tit For Tat's r distribution in tournaments. Lower values of r correspond to better performances. The best performance of the strategy has been in standard tournaments where it achieved a $\bar{r}$ of 0.34.

	Standard		Noisy		Noisy ($p_n < 0.1$)		Probabilistic ending		Probabilistic ending ($p_e < 0.1$)		Noisy probabilistic ending	
	Name	$\bar{r}$	Name	$\bar{r}$	Name	$\bar{r}$	Name	$\bar{r}$	Name	$\bar{r}$	Name	$\bar{r}$
0	Evolved HMM 5	0.007	Grumpy	0.140	DBS	0.000	Fortress4	0.013	Evolved FSM 16	0.000	Alternator	0.304
1	Evolved FSM 16	0.010	e	0.194	Evolved FSM 16 Noise 05	0.008	Defector	0.014	Evolved FSM 16 Noise 05	0.013	ϕ	0.310
2	EvolvedLookerUp2 2 2	0.011	Tit For 2 Tats	0.206	Evolved ANN 5 Noise 05	0.013	Better and Better	0.016	MEM2	0.027	e	0.312
3	Evolved FSM 16 Noise 05	0.017	Slow Tit For Two Tats	0.210	BackStabber	0.024	Tricky Defector	0.019	Evolved HMM 5	0.044	π	0.317
4	PSO Gambler 2 2 2	0.021	Cycle Hunter	0.215	DoubleCrosser	0.025	Fortress3	0.022	EvolvedLookerUp2 2 2	0.049	Limited Retaliate	0.353
5	Evolved ANN	0.029	Risky QLearner	0.222	Evolved ANN 5	0.028	Gradual Killer	0.025	Spiteful Tit For Tat	0.060	Anti Tit For Tat	0.354
6	Evolved ANN 5	0.034	Retaliate 3	0.229	Evolved ANN	0.038	Aggravater	0.028	Nice Meta Winner	0.068	Limited Retaliate 3	0.356
7	PSO Gambler 1 1 1	0.037	Cycler CCCCCD	0.235	Spiteful Tit For Tat	0.051	Raider	0.031	NMWE Finite Memory	0.069	Retaliate 3	0.356
8	Evolved FSM 4	0.049	Retaliate 2	0.239	Evolved HMM 5	0.051	Cycler DDC	0.045	NMWE Deterministic	0.070	Retaliate	0.357
9	PSO Gambler Mem1	0.050	Defector Hunter	0.240	Level Punisher	0.052	Hard Prober	0.051	Grudger	0.070	Retaliate 2	0.358
10	Winner12	0.060	Retaliate	0.242	Omega TFT	0.059	SolutionB1	0.060	NMWE Long Memory	0.074	Limited Retaliate 2	0.361
11	Fool Me Once	0.061	Hard Tit For 2 Tats	0.250	Fool Me Once	0.059	Meta Minority	0.061	Nice Meta Winner Ensemble	0.076	Hopeless	0.368
12	DBS	0.071	Limited Retaliate 3	0.253	PSO Gambler 2 2 2 Noise 05	0.067	Bully	0.061	EvolvedLookerUp1 1 1	0.077	Arrogant QLearner	0.407
13	DoubleCrosser	0.072	ShortMem	0.253	Evolved FSM 16	0.078	EasyGo	0.071	NMWE Memory One	0.080	Cautious QLearner	0.409
14	BackStabber	0.075	Limited Retaliate	0.257	EugineNier	0.080	Fool Me Forever	0.071	Winner12	0.085	Fool Me Forever	0.418

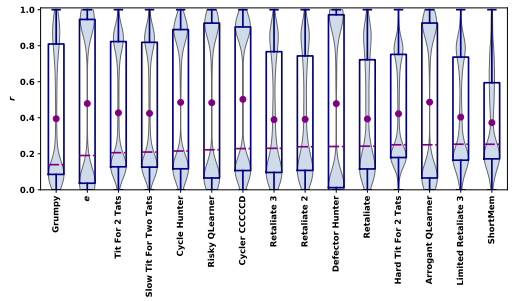
Table 4.3: Top performances for each tournament type based on $\bar{r}$. The results of each type are based on 11,420 unique tournaments. The results for noisy tournaments with $p_n < 0.1$ are based on 1,151 tournaments, and for probabilistic ending tournaments with $p_e < 0.1$ on 1,139. The top ranks indicate that trained strategies perform well in a variety of environments, but so do simple deterministic strategies. The normalised medians are close to 0 for most environments, except environments with noise not restricted to 0.1 regardless of the number of turns. Noisy and noisy probabilistic ending tournaments have the highest medians.



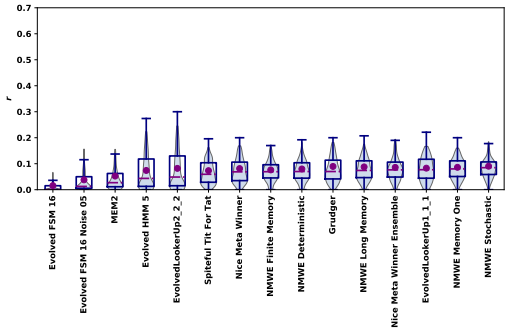
(a) r distributions of top 15 strategies in standard tournaments.



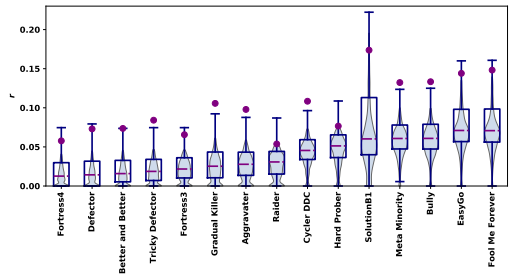
(b) r distributions of top 15 strategies in noisy tournaments with $p_n < 0.1$.



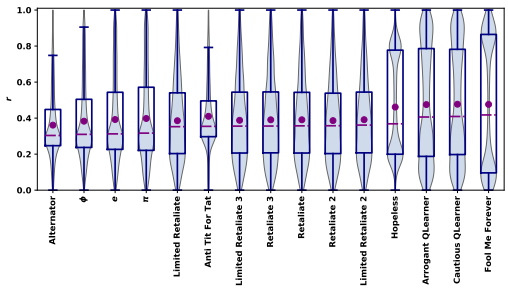
(c) r distributions of top 15 strategies in noisy tournaments.



(d) r distributions of top 15 strategies in 1139 probabilistic ending tournaments with $p_e < 0.1$.



(e) r distributions of top 15 strategies in probabilistic ending tournaments.



(f) r distributions of top 15 strategies in noisy probabilistic ending tournaments.

Figure 4.2: r distributions of the top 15 strategies in different environments. A lower value of $\bar{r}$ corresponds to a more successful performance. A strategy's r distribution skewed towards zero indicates that the strategy ranked highly in most tournaments it participated in. Most distributions are skewed towards zero except the distributions with unrestricted noise, supporting the conclusions from Table 4.3.

In standard tournaments 10 out of the 15 top strategies were introduced in [122]. These are strategies based on finite state automata (FSM), hidden Markov models (HMM), artificial neural networks (ANN), lookup tables (LookerUp) and stochastic lookup tables (Gambler) that have been trained using reinforcement learning algorithms (evolutionary and particle swarm algorithms). They have been trained to perform well against a subset of the strategies in APL in a standard tournament, thus their performance in the specific setting was anticipated although still noteworthy given the random sampling of tournament participants. **DoubleCrosser**, **BackStabber** and **Fool Me Once**, are strategies not from the literature but from APL. **DoubleCrosser** is an extension of **BackStabber** and both strategies make use of the number of turns because they are set to defect on the last two rounds. It should be noted that these strategies can be characterised as "cheaters" because the source code of the strategies allows them to know the number of turns in a match (unless the match has a probabilistic ending). These strategies were expected to not perform as well in tournaments where the number of turns is not specified. Finally, **Winner 12** [193] and **DBS** [31] are both from the literature. **DBS** is a strategy specifically designed for noisy environments, however, it ranks highly in standard tournaments as well. Similarly the fourth ranked player, **Evolved FSM 16 Noise 05**, was trained for noisy tournaments yet performs well in standard tournaments. Figure 4.2a shows that these strategies typically perform well in any standard tournament in which they participate.

In the case of noisy tournaments with smaller noise $p_n < 0.1$ the top performing strategies include strategies specifically designed for noisy tournaments. These are **DBS**, **Evolved FSM 16 Noise 05**, **Evolved ANN 5 Noise 05**, **PSO Gambler 2 2 2 Noise 05** and **Omega Tit For Tat** [159]. **Omega Tit For Tat**, another strategy designed to break the deadlocking cycles of CD and DC that **Tit For Tat** can fall into in noisy environments, places 10th. The rest of the top ranks are occupied by strategies which performed well in standard tournaments and deterministic strategies such as **Spiteful Tit For Tat** [4], **Level Punisher** [8], **Eugine Nier** [236].

In contrast, the performance of the top ranked strategies in noisy environments when $p_n \in [0, 1]$ is bimodal. The top strategies include strategies which decide their actions based on the cooperation to defection ratio, such as **ShortMem** [56], **Grumpy** [7] and **e** [7], and the **Retaliate** strategies which are designed to defect if the opponent has tricked them more often than a given percentage of the times that they have done the same. The bimodality of the r distributions is explained by Figure 4.3 which

demonstrates that the top 6 strategies were highly ranked due to the their performance in tournaments with $p_n > 0.5$, and that in tournaments with $p_n < 0.5$ they performed poorly. At a noisy level of 0.5 or greater, mostly cooperative strategies become mostly defectors and vice versa.

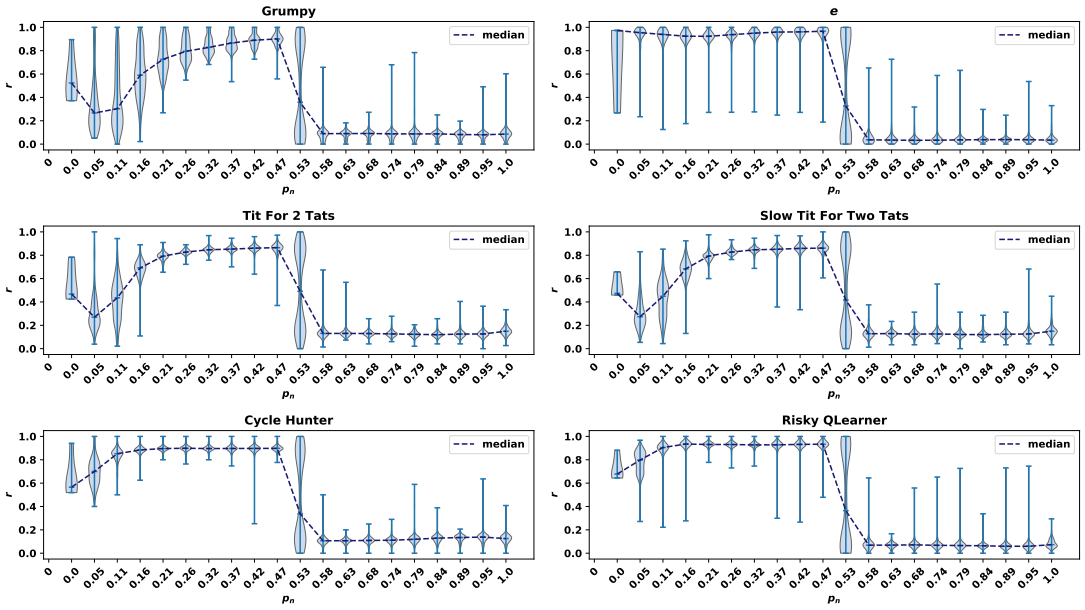


Figure 4.3: Normalised rank r distributions for top 6 strategies in noisy tournaments over the probability of noisy (p_n).

The most effective strategies in probabilistic ending tournaments with $p_e < 0.1$ are a series of ensemble Meta strategies, trained strategies which performed well in standard tournaments, and **Grudger** [7] and **Spiteful Tit For Tat** [4]. The Meta strategies [7] utilise a team of strategies and aggregate the potential actions of the team members into a single action in various ways. Figure 4.2d indicates that these strategies performed well in any probabilistic ending tournament.

In probabilistic ending tournaments with $p_e \in [0, 1]$ the top ranks are mostly occupied by defecting strategies such as **Better and Better**, **Gradual Killer**, **Hard Prober** (all from [7]), **Bully** (**Reverse Tit For Tat**) [213] and **Defector**, and a series of strategies based on finite state automata introduced by Daniel Ashlock and Wendy Ashlock: **Fortress3**, **Fortress4** (both introduced in [27]), **Raider** [29] and **Solution B1** [29]. The success of defecting strategies in probabilistic ending tournaments is due to larger values of p_e which lead to shorter matches (the expected number of rounds is $1/p_e$), so the impact of the PD being iterated is subdued. As stated by the Folk Theorem [88], defecting strategies do better when the likelihood of the game ending in the next turn increases. This is demonstrated by Figure 4.4, which gives the distributions of r for

the top 6 strategies in probabilistic ending tournaments over p_e.

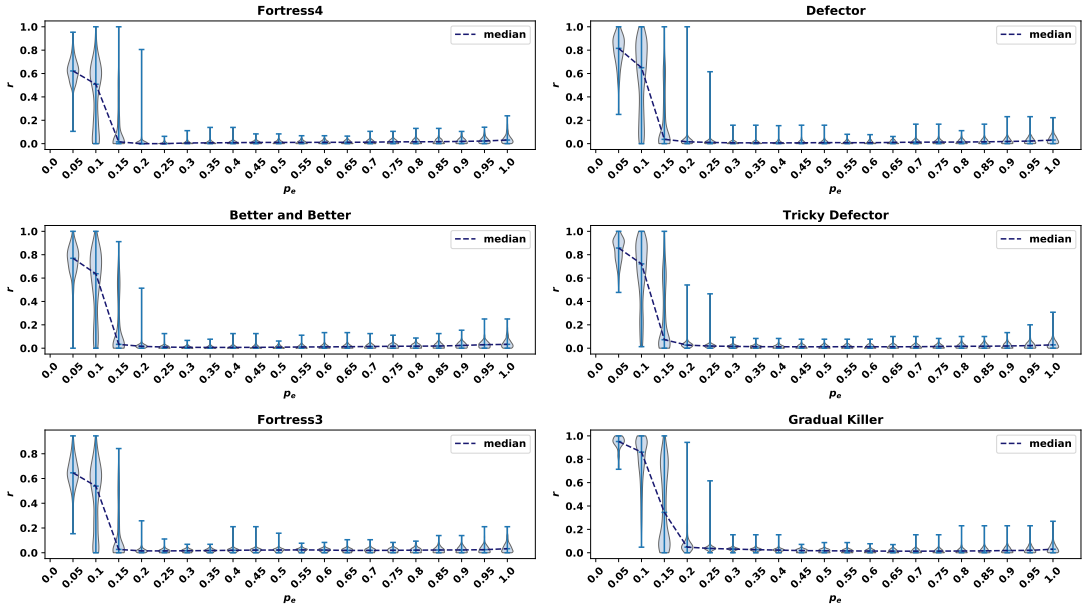


Figure 4.4: Normalised rank r distributions for top 6 strategies in probabilistic ending tournaments over p_e. The 6 strategies start of with a high median rank, however, their ranked decreased as the the probability of the game ending increased and at the point of $p_e = 0.1$.

The top performances in tournaments with both noise and a probabilistic ending and the top performances over the entire data set have the largest median values compared to the top rank strategies of the other tournament types, Figure 4.2f and Figure 4.5. The $\bar{r}$ for the top strategy is approximately at 0.3, indicating that the most successful strategy can on average just place in the top 30% of the competition.

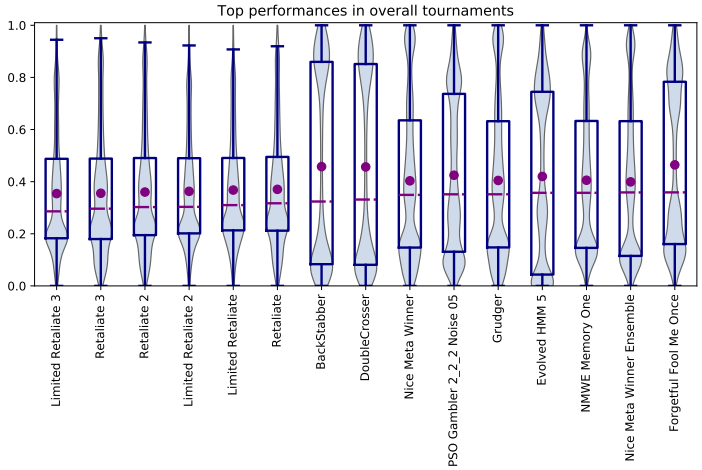


Figure 4.5: r distributions for best performed strategies in the data set [97]. A lower value of $\bar{r}$ corresponds to a more successful performance.

On the whole, the analysis of this section has shown that:

Name	$\bar{r}$
Limited Retaliate 3	0.286
Retaliate 3	0.297
Retaliate 2	0.302
Limited Retaliate 2	0.304
Limited Retaliate	0.311
Retaliate	0.317
BackStabber	0.324
DoubleCrosser	0.331
Nice Meta Winner	0.350
PSO Gambler 2 2 2 Noise 05	0.351
Grudger	0.352
NMWE Memory One	0.357
Evolved HMM 5	0.358
Nice Meta Winner Ensemble	0.359
Forgetful Fool Me Once	0.359

Table 4.4: Top performances over all the tournaments. The top ranks include strategies that have been previously mentioned. The set of **Retaliate** strategies occupy the top spots followed by **BackStabber** and **DoubleCrosser**. The distributions of the **Retaliate** strategies have no statistical difference. **PSO Gambler 2 2 2 Noise 05** and **Evolved HMM 5** are trained strategies introduced in [122] and **Nice Meta Winner** and **NMWE Memory One** are strategies based on teams. **Grudger** is a strategy from Axelrod's original tournament and Forgetful **Fool Me Once** is based on the same approach as **Grudger**.

- In standard tournaments the dominating strategies were strategies that had been trained using reinforcement learning techniques.
- In noisy environments where the noise probability strictly less than 0.1 was considered, the successful strategies were strategies specifically designed or trained for noisy environments.
- In probabilistic ending tournaments most of the highly ranked strategies were defecting strategies and trained finite state automata, all by the authors of [27, 29]. These strategies ranked high due to their performance in tournaments where the probability of the game ending after each turn was bigger than 0.1.
- In probabilistic tournaments with p_e less than 0.1 the highly ranked strategies were strategies based on the behaviour of others.
- From the collection of strategies considered here, no strategy can be consistently successful in noisy environments, except if the value of noise is constrained to less than a 0.1.

Though there is not a single strategy that repeatedly outranks all others in any of the

distinct tournament types, or even across the tournament types, there are specific types of strategies have been repeatedly ranked in the top ranks. These have been strategies that have been trained, strategies that defect, and strategies that would adapt their behaviour based on preassigned rules to achieve the highest outcome. These results contradict some of Axelrod's suggestions, and more specifically, the suggestions 'Do not be clever' and 'Do not be envious'. The features and properties contributing a strategy's success are further explored in section 4.4.

4.4 Evaluation of performance

This section examines the performance of the strategies based on features of strategies described in Table 4.5. These features are measures regarding a strategy's behaviour from the tournaments the strategies competed in as well as intrinsic properties such as whether a strategy is deterministic or stochastic.

feature	feature explanation	source	value type	min value	max value
stochastic	If a strategy is stochastic	strategy classifier from APL	boolean	Na	Na
makes use of game	If a strategy makes used of the game information	strategy classifier from APL	boolean	Na	Na
makes use of length	If a strategy makes used of the number of turns	strategy classifier from APL	boolean	Na	Na
memory usage	The memory size of a strategy divided by the number of turns	memory size from APL	float	0	1
SSE	A measure of how far a strategy is from ZD behaviour	method described in [166]	float	0	1
max cooperating rate ($C_{\max}$)	The biggest cooperating rate in a given tournament	result summary	float	0	1
min cooperating rate ($C_{\min}$)	The smallest cooperating rate in a given tournament	result summary	float	0	1
median cooperating rate (C_{median})	The median cooperating rate in a given tournament	result summary	float	0	1
mean cooperating rate (C_{mean})	The mean cooperating rate in a given tournament	result summary	float	0	1
C_r / $C_{\max}$	A strategy's cooperating rate divided by the maximum	result summary	float	0	1
$C_{\min}$ / C_r	A strategy's cooperating rate divided by the minimum	result summary	float	0	1
C_r / C_{median}	A strategy's cooperating rate divided by the median	result summary	float	0	1
C_r / C_{mean}	A strategy's cooperating rate divided by the mean	result summary	float	0	1
C_r	The cooperating ratio of a strategy	result summary	float	0	1
CC to C rate	The probability a strategy will cooperate after a mutual cooperation	result summary	float	0	1
CD to C rate	The probability a strategy will cooperate after being betrayed by the opponent	result summary	float	0	1
DC to C rate	The probability a strategy will cooperate after betraying the opponent	result summary	float	0	1
DD to C rate	The probability a strategy will cooperate after a mutual defection	result summary	float	0	1
p_n	The probability of a player's action being flip at each interaction	trial summary	float	0	1
n	The number of turns	trial summary	integer	1	200
p_e	The probability of a match ending in the next turn	trial summary	float	0	1
N	The number of strategies in the tournament	trial summary	integer	3	195
k	The number of repetitions of a given tournament	trial summary	integer	10	100

Table 4.5: The features which are included in the performance evaluation analysis. Stochastic, makes use of length and makes use of game are APL classifiers that determine whether a strategy is stochastic or deterministic, whether it makes use of the number of turns or the game's payoffs. The memory usage is calculated as the number of turns the strategy considers to make an action (which is specified in the APL) divided by the number of turns. The SSE (introduced in [166]) shows how close a strategy is to behaving as a ZDs, and subsequently, in an extortionate way. The method identifies the ZDs closest to a given strategy and calculates the algebraic distance between them, defined as SSE. More details on the measure are presented in Chapter 5. A SSE value of 1 indicates no extortionate behaviour at all whereas a value of 0 indicates that a strategy is behaving as a ZDs. The rest of the features considered include the cooperating ratio of a strategy, the minimum (C_{min}), maximum (C_{max}), mean (C_{mean}) and median (C_{median}) cooperating ratios of each tournament.

The memory usage of strategies is the number of rounds of play used by the strategy divided by the number of turns in each match. For example, **Winner 12** uses the previous two rounds of play, and if participating in a match with 100 turns its memory usage would be 2/100. For strategies with an infinite memory size, for example **Evolved FSM 16 Noise 05**, memory usage is equal to 1. Note that for tournaments with a probabilistic ending the number of turns was not collected, so the memory usage feature is not used for probabilistic ending tournaments.

The correlation coefficients between the features of Table 4.5 the median score and the median normalised rank are given by Table 4.6. The correlation coefficients between all features of Table 4.5 have been calculated and a graphical representation can be found in the Appendix C.1.

	Standard		Noisy		Probabilistic ending		Noisy probabilistic ending		Overall	
	r	median score	r	median score	r	median score	r	median score	r	median score
CC to C rate	-0.501	0.501	0.414	-0.504	0.408	-0.323	0.260	0.022	0.108	0.081
CD to C rate	0.226	-0.199	0.456	-0.330	0.320	-0.017	0.205	-0.220	0.281	-0.177
C_r	-0.323	0.384	0.711	-0.678	0.714	-0.832	0.579	-0.135	0.360	-0.124
C_r / C_{max}	-0.323	0.381	0.616	-0.551	0.714	-0.833	0.536	-0.116	0.395	-0.265
C_r / C_{mean}	-0.331	0.358	0.731	-0.740	0.721	-0.861	0.649	-0.621	0.428	-0.439
C_r / C_{median}	-0.331	0.353	0.652	-0.669	0.712	-0.852	0.330	-0.466	0.294	-0.405
C_r / C_{min}	0.109	-0.080	-0.358	0.250	-0.134	0.150	-0.368	0.113	0.000	0.280
C_{max}	-0.000	0.049	0.000	0.023	-0.000	0.046	0.000	-0.004	-0.000	0.553
C_{mean}	-0.000	0.229	-0.000	0.271	0.000	0.200	0.000	0.690	-0.000	0.544
C_{median}	0.000	0.209	-0.000	0.240	-0.000	0.187	-0.000	0.673	0.000	-0.250
C_{min}	0.000	0.084	0.000	-0.017	-0.000	0.007	-0.000	0.041	-0.161	-0.190
DC to C rate	0.127	-0.100	0.509	-0.504	-0.018	0.033	0.341	-0.016	0.173	-0.088
DD to C rate	0.412	-0.396	0.533	-0.436	-0.103	0.176	0.378	-0.263	0.237	-0.239
N	0.000	-0.009	-0.000	0.002	-0.000	0.003	-0.000	0.001	-0.000	-0.001
k	0.000	-0.002	-0.000	0.003	-0.000	0.001	-0.000	-0.008	0.000	-0.001
n	0.000	-0.125	-0.000	-0.024	-	-	-	-	0.000	-0.074
p_e	-	-	-	-	0.000	0.165	0.000	-0.058	0.000	0.055
p_n	-	-	-0.000	0.207	-	-	-0.000	-0.650	-0.000	-0.256
Make use of game	-0.003	-0.022	0.025	-0.082	-0.053	-0.108	0.013	-0.016	-0.004	-0.053
Make use of length	-0.158	0.124	0.005	-0.123	-0.025	-0.090	0.014	-0.016	-0.041	-0.026
SSE	0.473	-0.452	0.463	-0.337	-0.156	0.223	0.305	-0.259	0.233	-0.167
memory usage	-0.082	0.095	-0.007	-0.017	-	-	-	-	-0.053	0.046
stochastic	0.006	-0.024	0.022	-0.026	0.002	-0.130	0.021	-0.013	0.013	-0.048

Table 4.6: Correlations between the features of Table 4.5 and the normalised rank and the median score.

In standard tournaments the features CC to C, C_r, $C_r/C_{\max}$ and the cooperating ratio compared to C_{median} and C_{mean} have a moderately negative effect on the normalised rank (smaller rank is better), and a moderate positive on the median score. The SSE error and the DD to C rate have the opposite effects. Thus, in standard tournaments behaving cooperatively corresponds to a more successful performance. Even though being nice generally pays off that does not hold against defective strategies. Being more cooperative after a mutual defection, that is not retaliating, is associated to lesser

overall success in terms of normalised rank. Figure 4.6 confirms that the winners of standard tournaments always cooperate after a mutual cooperation and almost always defect after a mutual defection.

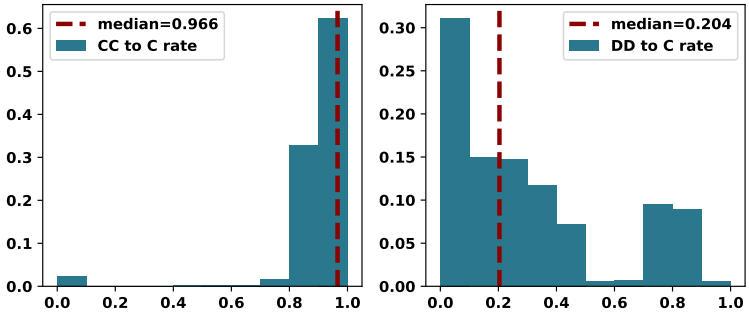


Figure 4.6: Distributions of CC to C and DD to C for the winners in standard tournaments.

Compared to standard tournaments, in both noisy and in probabilistic ending tournaments the higher the rates of cooperation the lower a strategy's success and median score. A strategy would want to cooperate less than both the mean and median cooperator in such settings. In probabilistic ending tournaments the correlation coefficients have larger values, indicating a stronger effect. Thus a strategy will be punished more by its cooperative behaviour in probabilistic ending environments, supporting the results of section 4.4 as well. The distributions of the C_r of the winners in both tournaments are given by Figure 4.7. It confirms that the winners in noisy tournaments cooperated less than 35% of the time and in probabilistic ending tournaments less than 10%. In noisy probabilistic ending tournaments and over all the tournaments' results, the only features that had a moderate effect are $C_r/C_{\text{mean}}, C_r/C_{\max}$ and C_r. In such environments cooperative behaviour appears to be punished less than in noisy and probabilistic ending tournaments.

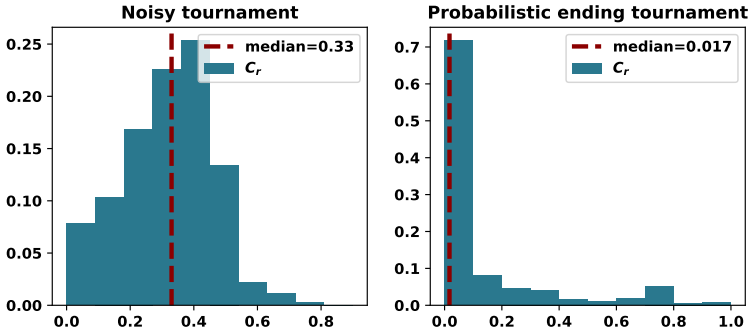


Figure 4.7: C_r distributions of the winners in noisy and in probabilistic ending tournaments.

A multivariate linear regression has been fitted to model the relationship between the features and the normalised rank. Based on the graphical representation of the correlation matrices given in Appendix C.1 several of the features are highly correlated and have been removed before fitting the linear regression model. The features included are given by Table 4.7 alongside their corresponding p values in the distinct tournaments and their regression coefficients.

	Standard		Noisy		Probabilistic ending		Noisy probabilistic ending		Overall	
	R adjusted: 0.541		R adjusted: 0.639		R adjusted: 0.587		R adjusted: 0.577		R adjusted: 0.242	
	Coefficient	p-value	Coefficient	p-value	Coefficient	p-value	Coefficient	p-value	Coefficient	p-value
CC to C rate	-0.042	0.000	-0.007	0.000	0.017	0.000	0.111	0.0	-0.099	0.0
CD to C rate	0.297	0.000	-0.068	0.000	0.182	0.000	0.023	0.0	0.129	0.0
C_r / C_{max}	-	-	1.856	0.000	-	-	1.256	0.0	-	-
C_r / C_{mean}	-0.468	0.000	-0.577	0.000	0.525	0.000	-0.120	0.0	0.300	0.0
C_{max}	-0.071	0.000	-	-	-0.022	0.391	1.130	0.0	-	-
C_{mean}	0.118	0.000	-2.558	0.000	-0.023	0.001	-1.489	0.0	-	-
C_{min}	-0.161	0.000	-1.179	0.000	-0.170	0.000	-	-	-	-
C_{min} / C_r	0.057	0.000	-0.320	0.000	0.125	0.000	-	-	-0.103	0.0
DC to C rate	0.198	0.000	0.040	0.000	-0.030	0.000	0.022	0.0	0.064	0.0
k	0.000	0.319	0.000	0.020	0.000	0.002	0.000	0.0	-	-
n	0.000	0.000	-	-	-	-	-	-	-	-
p_e	-	-	-	-	0.000	0.847	-0.083	0.0	-	-
p_n	-	-	-0.048	0.000	-	-	-	-	-	-
SSE	0.258	0.000	0.153	0.000	-0.041	0.000	0.100	0.0	0.056	0.0
constant	0.697	0.000	1.522	0.000	-0.057	0.019	-0.472	0.0	0.178	0.0
memory usage	-0.010	0.000	-0.000	0.035	-	-	-	-	-	-

Table 4.7: Results of multivariate linear regressions with r as the dependent variable. R squared is reported for each model.

A multivariate linear regression has also be fitted on the median score. The coefficients and p values of the features can be found in Appendix C.2. The results of the two methods are in agreement.

The feature C_r/C_{mean} has a statistically significant effect across all models and a high regression coefficient. It has both a positive and negative impact on the normalised rank depending on the environment. For standard tournaments, Figure 4.8 gives the distributions of several features for the winners of standard tournaments. The C_r/C_{mean} distribution of the winner is also given in Figure 4.8. A value of $C_r/C_{\text{mean}} = 1$ implies that the cooperating ratio of the winner was the same as the mean cooperating ratio of the tournament, and in standard tournaments, the median is 1. Therefore, an effective strategy in standard tournaments was the mean cooperator of its respective tournament.

The distributions of SSE and CD to C rate for the winners of standard tournaments are also given in Figure 4.8. The SSE distributions for the winners indicate that the strategy behaved in a ZD way in several tournaments, however, not constantly. The winners participated in matches where they did not try to extortionate their opponents. Furthermore, the CD to C distribution indicates that if a strategy were to defect against the winners the winners would reciprocate on average with a probability of 0.5.

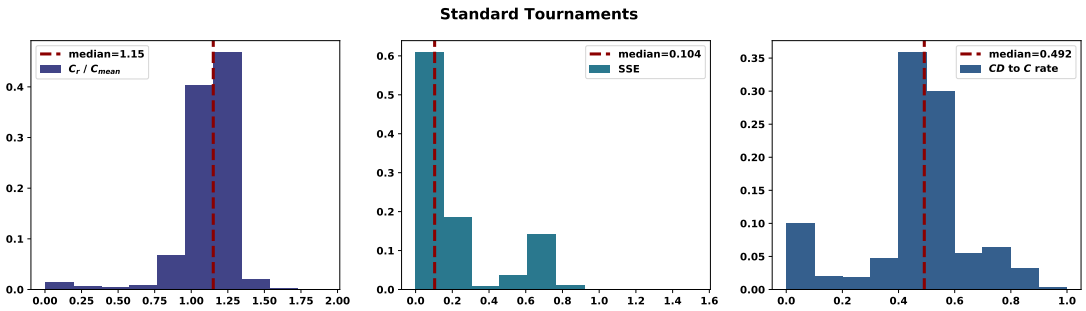


Figure 4.8: Distributions of C_r/C_{mean}, SSE and CD to C ratio for the winners of standard tournaments. A value of $C_r/C_{\text{mean}} = 1$ imply that the cooperating ratio of the winner was the same as the mean cooperating ratio of the tournament. An SSE distribution skewed towards 0 indicates a extortionate behaviour by the strategy.

Similarly for the rest of the different tournaments types, and the entire data set the distributions of C_r/C_{mean}, SSE and CD to C ratio are given by Figures 4.9, 4.11, 4.12 and 4.13.

Based on the C_r/C_{mean} distributions the successful strategies have adapted differently to the mean cooperator depending on the tournament type. In noisy tournaments where the median of the distribution is at 0.67, and thereupon the winners cooperated 67% of the time the mean cooperator did. In tournaments with noise and a probabilistic ending the winners cooperated 60%, whereas in settings that the type of the tournament can vary between all the types the winners cooperated 67% of the time the mean cooperator did. Lastly, in probabilistic ending tournaments above more defecting strategies prevail (section 4.3), and this result is reflected here.

The probability of noise has been observed to substantially affect optimal behaviour. Figure 4.10 gives the ratio C_r/C_{mean} for the winners in tournaments with noise, over the probability of noise. From Figure 4.10a it is clear that the cooperating only 67% of the time the mean cooperator did is optimal only when $p_n \in [0.2, 0.4)$ and $p_n \in [0.6, 0.7]$. In environments with $p_n < 0.1$ the winners want to be close to the mean cooperator, similarly to standard tournaments, and as the probability of noise is exceeding 0.5 (where the game is effectively inverted) strategies should aim to be less and less

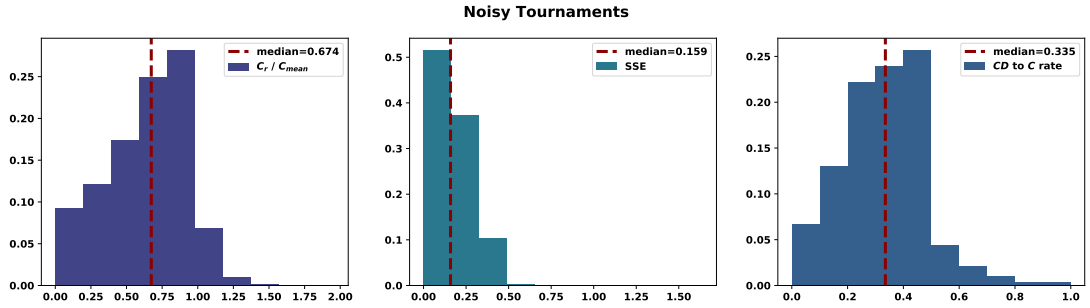
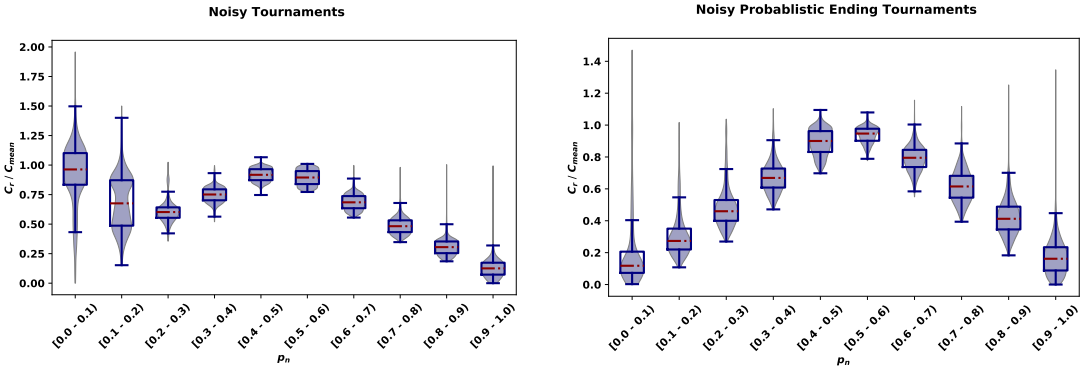


Figure 4.9: Distributions of C_r/C_{mean}, SSE and CD to C ratio for the winners of noisy tournaments.

cooperative.

Figure 4.10 gives C_r/C_{mean} for the winners over p_n in tournaments with noise and a probabilistic ending. The optimal proportions of cooperations are different now that the number of turns is not fixed, successful strategies want to be more defecting that the mean cooperator, that only changes when p_n approaches 0.5. Figure 4.10 demonstrates how the adjustments to C_r/C_{mean} change over the noise in the to the environment, and thus supports how important adapting to the environment is for a strategy to be successful.



(a) C_r/C_{mean} distribution for winners in noisy tournaments over p_n.

(b) C_r/C_{mean} distribution for winners in noisy probabilistic ending tournaments over p_n.

Figure 4.10: C_r/C_{mean} distributions over intervals of p_n. These distributions model the optimal proportion of cooperation compared to C_{mean} as a function of (p_n).

The distributions of the SSE across the tournament types suggest that successful strategies exhibit some extortionate behaviour, but not constantly. ZDs are a set of strategies that are often envious as they try to exploit their opponents. The winners of the tournaments considered in this analysis are envious, but not as much as many ZDs. Though the exact interactions between the matches have not been recorded here, the work of [122] which introduced the trained strategies that appeared in the top ranked

strategies of section 4.3 did. In [122] it was shown that clever strategies managed to achieve mutual cooperation with stronger strategies whilst exploiting the weaker strategies. This could explain the clever winners of this analysis, and would explain the SSE distributions. This could also be the reason why ZDs fail to appear in the tops ranks – they try to exploit all opponents and cannot actively adapt back to mutual cooperation against stronger strategies, which requires more depth of memory. Note that ZDs also tend to perform poorly in population games for a similar reason: they attempt to exploit other players using ZDs, failing to form a cooperative sub population [163]. This makes them good invaders but poor resisters of invasion.

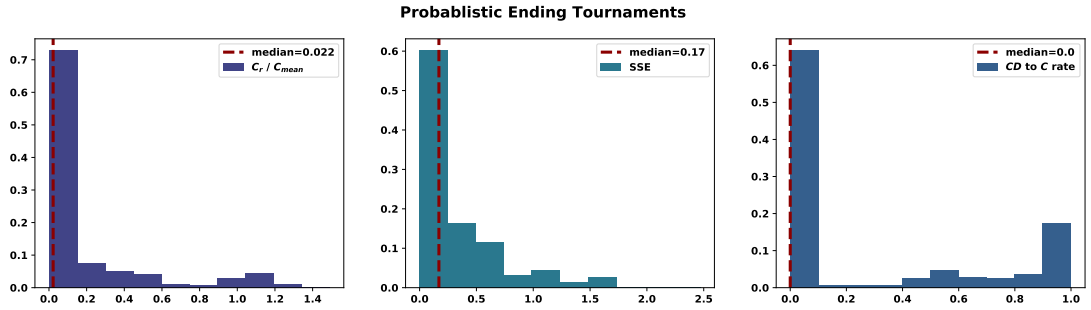


Figure 4.11: Distributions of C_r/C_{mean}, SSE and CD to C ratio for the winners of probabilistic ending tournaments.

The distributions of the CD to C rate evaluate the behaviour of a successful strategy after its opponent has defected against it. In standard tournaments it was observed that a successful strategy reciprocates with a probability of 0.5, and in a setting that the type of the tournament can vary between all the examined types a winning strategy would reciprocate on average with a probability of 0.58. In tournaments with noise a strategy is less likely to cooperate following a defection compared to standard tournaments, and in probabilistic ending tournaments a strategy will reciprocate a defection. This leads to adjusting the recommendation of being provocable to defections made by Axelrod. A strategy should be provocable in tournaments with short matches, but in the rest of the settings a strategy should be more generous.

Further statistically significant features with strong effects include C_r/C_{min}, C_r/C_{max}, C_{min} and C_{max}. These add more emphasis on how important it is for a strategy to adapt to its environment. Finally, the features number of turns, repetitions and the probabilities of noise and the game ending had no significant effects based on the multivariate regression models.

A third method that evaluates the importance of the features in Table 4.5 using clus-

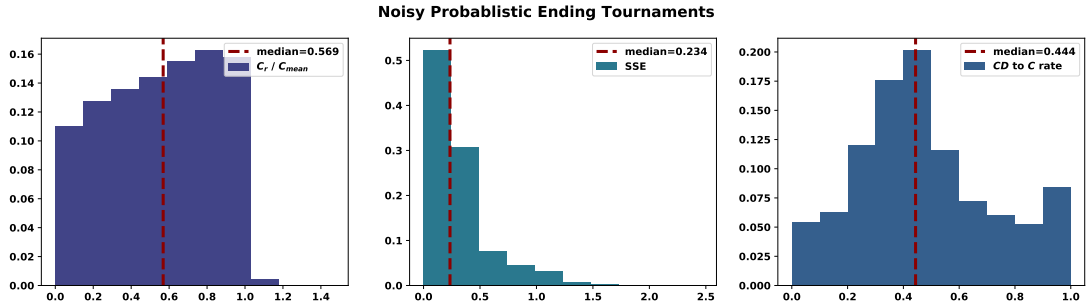


Figure 4.12: Distributions of C_r/C_{mean}, SSE and CD to C ratio for the winners of noisy probabilistic ending tournaments.

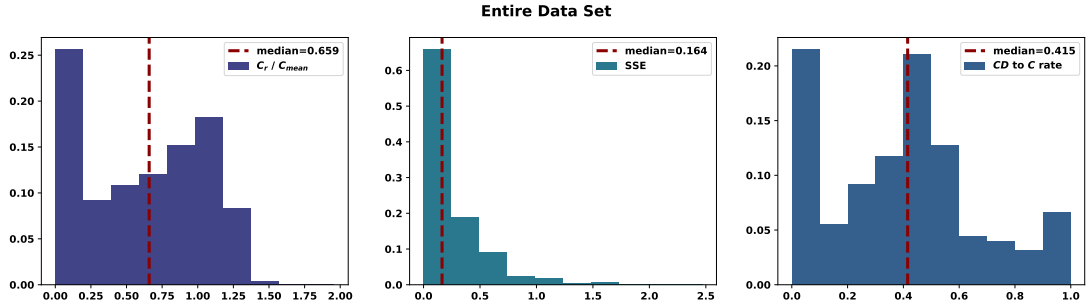


Figure 4.13: Distributions of C_r/C_{mean}, SSE and CD to C ratio for the winners over the tournaments of the entire data set.

tering and random forests can be found in the Appendix C.3. The results uphold the outcomes of the correlation and multivariate regression. It also evaluates the effects of the classifiers stochastic, make use of game, and make use of length which have not been evaluated by the methods above because there are binary variables. The results imply that they have no significant effect on a strategy's performance.

4.5 Chapter summary

This Chapter explored the performance of 195 strategies of the IPD in 45,600 computer tournaments. The collection of computer tournaments presented here is the largest and most diverse collection in the literature. The 195 strategies are drawn from the APL and include strategies from the IPD literature. The computer tournaments include tournaments of four different types.

So what is the best way of playing the IPD? And is there a single dominant strategy for the IPD?

There was not a single strategy within the collection of the 195 strategies that managed to perform well in all the tournaments variations it competed in. Even if on average

a strategy ranked highly in a specific environment this did not guarantee its success over the different tournament types. Nevertheless, in sections 4.3 and 4.4 examined the best performing strategies across various tournament types and analysed their salient features. It was demonstrated that there are properties associated with the success of strategies which in fact contradict the originally suggested properties of Axelrod [34].

It was shown that complex or clever strategies can be effective, whether trained against a corpus of possible opponents or purposely designed to mitigate the impact of noise such as the **DBS** strategy. Moreover, it was found that some strategies designed or trained for noisy environments were also highly ranked in noise-free tournaments which reinforces the idea that strategies' complexity/cleverness is not necessarily a liability, rather it can confer adaptability to a more diverse set of environments. It was also shown that while the type of exploitation attempted by ZDs is not typically effective in standard tournaments, envious strategies capable of both exploiting and not their opponents can be highly successful. Based on the results of [122] this could be because they are selectively exploiting weaker opponents while mutually cooperating with stronger opponents. Highly noisy or tournaments with short matches also favoured envious strategies. These environments mitigated the value of being nice. Uncertainty enables exploitation, reducing the ability of maintaining or enforcing mutual cooperation, while triggering grudging strategies to switch from typically cooperating to typically defecting.

The feature analysis of the best performing strategies demonstrated that a strategy should reciprocate, as suggested by Axelrod, but it should relax its readiness to do so and be more generous. For noisy environments this is inline with the results of [46, 74, 210, 262], however, it was also showed that generosity pays off even in standard settings, and that in fact the only setting a strategy would want to be too provocable is when the matches are not long. Forgiveness as defined by Axelrod was not explored in this Chapter. This was mainly because the two round states were not recorded during the data collection. This could be a topic of future work that examines the impact of considering more rounds of history. The features analysis also concluded that there is a significant importance in adapting to the environment, and more specifically, to the mean cooperator. In standard tournaments a strategy would aim to be the mean cooperator while in noisy tournaments the best performing players cooperate at a lower rate than the tournament population on average. Moreover, the manner in which

a strategy achieves a given cooperation rate relative to the tournament population average is important.

This could potentially explain the early success of Tit For Tat. Tit For Tat naturally achieves a cooperation rate near C_{mean} by virtue of copying its opponent's last move while also minimising instances where it is exploited by an opponent (cooperating while the opponent defects), at least in non-noisy tournaments. It could also explain why Tit For N Tats does not fare well for $N > 1$ – it fails to achieve the proper cooperation ratio by tolerating too many defections.

Similarly, the results could suggest an explanation regarding the intuitively unexpected effectiveness of memory-one strategies historically. Given that among the important features associated with success are the relative cooperation rate to the population average and the four memory-one probabilities of cooperating conditional on the previous round of play, these features can be optimised by a memory-one strategy such as Tit For Tat. Usage of more history becomes valuable when there are exploitable opponent patterns. This is indicated by the importance of SSE as a feature, showing that the first-approximation provided by a memory-one strategy is no longer sufficient. The limitations of memory are further explored in Chapter 5.

Overall, the five properties successful strategies need to have in a IPD competition based on the analysis that has been presented in this Chapter are:

- ~~Do not be envious~~ Be a little bit envious
- Be "nice" in non-noisy environments or when game lengths are longer
- Reciprocate both cooperation and defection appropriately; Be provocable in tournaments with short matches, and generous when matches are longer
- ~~Do not be too clever~~ It is okay to be clever
- Adapt to the environment; Adjust to the mean population cooperation rate

In this Chapter optimal behaviour was explored whilst considering a collection of pre defined strategies. In comparison, Chapter 5 does not consider pre defined strategies but estimates the exact best responses. This is done by considering strategies with a theory of mind that compete in e environments of memory-one opponents.

Chapter 5

Stability of defection, optimisation of strategies and the limits of memory in the Prisoner's Dilemma.

The research reported in this Chapter has lead to a manuscript, entitled:

"A theory of mind: Best responses to memory-one strategies. The limitations of extortion and restricted memory"

Available at: `arxiv.org/abs/1911.12112`

Associated data set: [101]

Associated codebase: [108]

Axelrod-Python library (APL) version: 4.4.0

The manuscript's abstract is the following:

Memory-one strategies are a set of Iterated Prisoner's Dilemma strategies that have been praised for their mathematical tractability and performance against single opponents. This manuscript investigates a theory of mind: *best response* memory-one strategies, as a multidimensional optimisation problem. We add to the literature that has shown that extortionate play is not always optimal by showing that optimal play is often not extortionate. We also provide evidence that memory-one strategies suffer from their limited memory in multi agent interactions and can be out performed by

optimised strategies with longer memory.

The differences between the Chapter and the manuscript include details on the bespoke open source package used to carried out the numerical experiments, details on resultant theory and an additional section on reactive strategies. These details and the additional section are only reported in this Chapter, and not in the manuscript. The Chapter also includes details an introduction to the Bayesian optimisation used to carry out the numerical experiments.

5.1 Introduction

This Chapter contributes to the question: what is the optimal behaviour an IPD strategy should adapt as a response to different environments? In [237] the authors stated that "Only a player with a theory of mind about his opponent can do better, in which case Iterated Prisoner's Dilemma is an Ultimatum Game". The purpose of this Chapter is to investigate the first part of this sentence, more specifically, to investigate the best response strategy with a theory of mind in an environment with memory-one opponents, and to understand the effects of extortion and restricted memory in those environments. Extortionate behaviour is explored using a linear algebraic approach presented in [166].

The outcomes of this Chapter reinforce and extend known results which were presented in Chapter 2. Namely that memory-one strategies must adaptable to be successful [121, 166] and that longer-memory strategies have a certain form of advantage over short memory strategies [130, 233]. The Chapter is structured as follows:

- section 5.2 describes a closed form algebraic expression for the utility of a memory-one strategy to a given group of opponents.
- section 5.3 produces a compact method of identifying the best response memory-one strategy against a given set of memory-one opponents.
- section 5.4 explains best response reactive strategies and demonstrates the usage of resultant theory in explicitly finding a reactive best response.
- section 5.5 describes a series of numerical experiments and a well designed framework that allows the comparison of an optimal memory-one strategy and a more

complex strategy which has a larger memory.

- section 5.6 presents a compact method of identifying environments for which cooperation cannot occur.

5.2 Quadratic form utility of the IPD

One specific advantage of memory-one strategies is their mathematical tractability. They can be represented completely as an element of $\mathbb{R}^4_{[0,1]}$. As previously discussed in Chapter 2, if a strategy is concerned with only the outcome of a single turn then there are four possible 'states' the strategy could be in:

- Both players cooperated, denoted as CC.
- First player cooperated while the second one defected, denoted as CD.
- First player defected while the second one cooperated, denoted as DC.
- Both players defected, denoted as DD.

Therefore, a memory-one strategy can be denoted by the probability vector of cooperating after each of these states; $p = (p_1, p_2, p_3, p_4) \in \mathbb{R}^4_{[0,1]}$.

In [221] it was shown that it is not necessary to simulate the play of a strategy p against a memory-one opponent q. Rather this exact behaviour can be modelled as a stochastic process, and more specifically as a Markov chain (Figure 5.1) whose corresponding transition matrix M is given by Equation (5.1). The long run steady state probability vector v, which is the solution to $vM = v$, can be combined with the payoff matrices of Equation (1.1) to give the expected payoffs for each player. More specifically, the utility for a memory-one strategy p against an opponent q, denoted as $u_q(p)$, is given by Equation (5.2).

$$M = \begin{bmatrix} p_1q_1 & p_1\left(-q_1+1\right) & q_1\left(-p_1+1\right) & \left(-p_1+1\right)\left(-q_1+1\right) \\ p_2q_3 & p_2\left(-q_3+1\right) & q_3\left(-p_2+1\right) & \left(-p_2+1\right)\left(-q_3+1\right) \\ p_3q_2 & p_3\left(-q_2+1\right) & q_2\left(-p_3+1\right) & \left(-p_3+1\right)\left(-q_2+1\right) \\ p_4q_4 & p_4\left(-q_4+1\right) & q_4\left(-p_4+1\right) & \left(-p_4+1\right)\left(-q_4+1\right) \end{bmatrix} \tag{5.1}$$

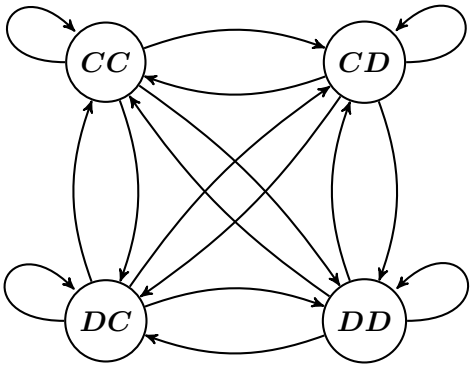


Figure 5.1: Markov Chain

$$u_q(p) = v \cdot (R, S, T, P). \tag{5.2}$$

This thesis is the first work to explore the form of $u_q(p)$. The first theoretical result of the thesis is given by Theorem 1 which states that $u_q(p)$ is given by a ratio of two quadratic forms [160].

Theorem 1. *The expected utility of a memory-one strategy $p \in \mathbb{R}^4_{[0,1]}$ against a memory-one opponent $q \in \mathbb{R}^4_{[0,1]}$, denoted as $u_q(p)$, can be written as a ratio of two quadratic forms:*

$$u_q(p) = \frac{\frac{1}{2}pQp^T + cp + a}{\frac{1}{2}p\bar{Q}p^T + \bar{c}p + \bar{a}}, \tag{5.3}$$

where $Q, \bar{Q} \in \mathbb{R}^{4\times 4}$ are square matrices defined by the transition probabilities of the opponent q_1, q_2, q_3, q_4 as follows:

$$Q = \begin{bmatrix} 0 & -(q_1-q_3)(Pq_2-P-Tq_4) & (q_1-q_2)(Pq_3-Sq_4) & (q_1-q_4)(Sq_2-S-Tq_3) \\ -(q_1-q_3)(Pq_2-P-Tq_4) & 0 & (q_2-q_3)(Pq_1-P-Rq_4) & -(q_3-q_4)(Rq_2-R-Tq_1+T) \\ (q_1-q_2)(Pq_3-Sq_4) & (q_2-q_3)(Pq_1-P-Rq_4) & 0 & (q_2-q_4)(Rq_3-Sq_1+S) \\ (q_1-q_4)(Sq_2-S-Tq_3) & -(q_3-q_4)(Rq_2-R-Tq_1+T) & (q_2-q_4)(Rq_3-Sq_1+S) & 0 \end{bmatrix}, \tag{5.4}$$

$$\bar{Q} = \begin{bmatrix} 0 & -(q_1-q_3)(q_2-q_4-1) & (q_1-q_2)(q_3-q_4) & (q_1-q_4)(q_2-q_3-1) \\ -(q_1-q_3)(q_2-q_4-1) & 0 & (q_2-q_3)(q_1-q_4-1) & (q_1-q_2)(q_3-q_4) \\ (q_1-q_2)(q_3-q_4) & (q_2-q_3)(q_1-q_4-1) & 0 & -(q_2-q_4)(q_1-q_3-1) \\ (q_1-q_4)(q_2-q_3-1) & (q_1-q_2)(q_3-q_4) & -(q_2-q_4)(q_1-q_3-1) & 0 \end{bmatrix}. \tag{5.5}$$

c and $\bar{c} \in \mathbb{R}^{4\times 1}$ are similarly defined by:

$$c = \begin{bmatrix} \boldsymbol{q_1\left(Pq_2 - P - Tq_4\right)} \\ \boldsymbol{-\left(q_3 - 1\right)\left(Pq_2 - P - Tq_4\right)} \\ \boldsymbol{-Pq_1q_2 + Pq_2q_3 + Pq_2 - Pq_3 + Rq_2q_4 - Sq_2q_4 + Sq_4} \\ \boldsymbol{-Rq_2q_4 + Rq_4 + Sq_2q_4 - Sq_2 - Sq_4 + S + Tq_1q_4 - Tq_3q_4 + Tq_3 - Tq_4} \end{bmatrix}, \tag{5.6}$$

$$\bar{c} = \begin{bmatrix} \boldsymbol{q_1\left(q_2 - q_4 - 1\right)} \\ \boldsymbol{-\left(q_3 - 1\right)\left(q_2 - q_4 - 1\right)} \\ \boldsymbol{-q_1q_2 + q_2q_3 + q_2 - q_3 + q_4} \\ \boldsymbol{q_1q_4 - q_2 - q_3q_4 + q_3 - q_4 + 1} \end{bmatrix}, \tag{5.7}$$

and the constant terms $a, \bar{a}$ are defined as $a = -Pq_2 + P + Tq_4$ and $\bar{a} = -q_2 + q_4 + 1$.

Proof. It was discussed that $u_q(p)$ it is the product of the steady state vector v and the PD payoffs,

$$u_q(p) = v \cdot (R, S, T, P).$$

The dot product of $v \cdot (R, S, T, P)$ gives,

$$u_q(p) = \begin{pmatrix} -p_1p_2(q_1-q_3)(Pq_2-P-Tq_4)+p_1p_3(q_1-q_2)(Pq_3-Sq_4)+p_1p_4(q_1-q_4)(Sq_2-S-Tq_3)+p_2p_3(q_2-q_3)(Pq_1-P-Rq_4)- \\ p_2p_4(q_3-q_4)(Rq_2-R-Tq_1+T)+p_3p_4(q_2-q_4)(Rq_3-Sq_1+S)+p_1q_1(Pq_2-P-Tq_4)-p_2(q_3-1)(Pq_2-P-Tq_4)+ \\ p_3(-Pq_1q_2+Pq_2q_3+Pq_2-Pq_3+Rq_2q_4-Sq_2q_4+Sq_4)+p_4(-Rq_2q_4+Rq_4+Sq_2q_4-Sq_2-Sq_4+S) \\ \frac{Tq_1q_4-Tq_3q_4+Tq_3-Tq_4-Pq_2+P+Tq_4}{p_1p_2(q_1q_2-q_1q_4-q_1-q_2q_3+q_3q_4+q_3)+p_1p_3(-q_1q_3+q_1q_4+q_2q_3-q_2q_4)+p_1p_4(-q_1q_2+q_1q_3+q_1+q_2q_4-q_3q_4-q_4)+} \\ p_2p_3(-q_1q_2+q_1q_3+q_2q_4+q_2-q_3q_4-q_3)+p_2p_4(-q_1q_3+q_1q_4+q_2q_3-q_2q_4)+p_3p_4(q_1q_2-q_1q_4-q_2q_3-q_2+q_3q_4+q_4)+ \\ p_1(-q_1q_2+q_1q_4+q_1)+p_2(q_2q_3-q_2-q_3q_4-q_3+q_4+1)+p_3(q_1q_2-q_2q_3-q_2+q_3-q_4)+p_4(-q_1q_4+q_2+q_3q_4-q_3+q_4-1)+ \\ q_2-q_4-1 \end{pmatrix}.$$

Consider the numerator of $u_q(p)$. The cross product terms p_ip_j are given by,

$$-p_1p_2(q_1 - q_3)(Pq_2 - P - Tq_4) + p_1p_3(q_1 - q_2)(Pq_3 - Sq_4) + p_1p_4(q_1 - q_4)(Sq_2 - S - Tq_3) +$$
$$p_2p_3(q_2 - q_3)(Pq_1 - P - Rq_4) - p_2p_4(q_3 - q_4)(Rq_2 - R - Tq_1 + T) + p_3p_4(q_2 - q_4)(Rq_3 - Sq_1 + S)$$

This can be re written in a matrix format given by Equation (5.8).

$$(p_1,p_2,p_3,p_4)^{\frac{1}{2}}\begin{bmatrix} 0 & -(q_1-q_3)(Pq_2-P-Tq_4) & (q_1-q_2)(Pq_3-Sq_4) & (q_1-q_4)(Sq_2-S-Tq_3) \\ -(q_1-q_3)(Pq_2-P-Tq_4) & 0 & (q_2-q_3)(Pq_1-P-Rq_4) & -(q_3-q_4)(Rq_2-R-Tq_1+T) \\ (q_1-q_2)(Pq_3-Sq_4) & (q_2-q_3)(Pq_1-P-Rq_4) & 0 & (q_2-q_4)(Rq_3-Sq_1+S) \\ (q_1-q_4)(Sq_2-S-Tq_3) & -(q_3-q_4)(Rq_2-R-Tq_1+T) & (q_2-q_4)(Rq_3-Sq_1+S) & 0 \end{bmatrix}\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix} \tag{5.8}$$

Similarly, the linear terms are given by,

$$p_1q_1(Pq_2-P-Tq_4)+p_4(-Rq_2q_4+Rq_4+Sq_2q_4-Sq_2-Sq_4+S+Tq_1q_4-Tq_3q_4+Tq_3-Tq_4)$$
$$-p_2(q_3-1)(Pq_2-P-Tq_4)+p_3(-Pq_1q_2+Pq_2q_3+Pq_2-Pq_3+Rq_2q_4-Sq_2q_4+Sq_4)$$

and the expression can be written using a matrix format as Equation (5.9).

$$(p_1,p_2,p_3,p_4)\begin{bmatrix} q_1(Pq_2-P-Tq_4) \\ -(q_3-1)(Pq_2-P-Tq_4) \\ -Pq_1q_2+Pq_2q_3+Pq_2-Pq_3+Rq_2q_4-Sq_2q_4+Sq_4 \\ -Rq_2q_4+Rq_4+Sq_2q_4-Sq_2-Sq_4+S+Tq_1q_4-Tq_3q_4+Tq_3-Tq_4 \end{bmatrix} \tag{5.9}$$

Finally, the constant term of the numerator, which is obtained by substituting $p = (0,0,0,0)$, is given by Equation (5.10).

$$-Pq_2+P+Tq_4 \tag{5.10}$$

Combining Equation (5.8), Equation (5.9) and Equation (5.10) gives that the numerator of $u_q(p)$ can be written as,

$$\frac{1}{2}p\begin{bmatrix} 0 & -(q_1-q_3)(Pq_2-P-Tq_4) & (q_1-q_2)(Pq_3-Sq_4) & (q_1-q_4)(Sq_2-S-Tq_3) \\ -(q_1-q_3)(Pq_2-P-Tq_4) & 0 & (q_2-q_3)(Pq_1-P-Rq_4) & -(q_3-q_4)(Rq_2-R-Tq_1+T) \\ (q_1-q_2)(Pq_3-Sq_4) & (q_2-q_3)(Pq_1-P-Rq_4) & 0 & (q_2-q_4)(Rq_3-Sq_1+S) \\ (q_1-q_4)(Sq_2-S-Tq_3) & -(q_3-q_4)(Rq_2-R-Tq_1+T) & (q_2-q_4)(Rq_3-Sq_1+S) & 0 \end{bmatrix}p^T+$$
$$\begin{bmatrix} q_1(Pq_2-P-Tq_4) \\ -(q_3-1)(Pq_2-P-Tq_4) \\ -Pq_1q_2+Pq_2q_3+Pq_2-Pq_3+Rq_2q_4-Sq_2q_4+Sq_4 \\ -Rq_2q_4+Rq_4+Sq_2q_4-Sq_2-Sq_4+S+Tq_1q_4-Tq_3q_4+Tq_3-Tq_4 \end{bmatrix}p-Pq_2+P+Tq_4$$

and equivalently as,

$$\frac{1}{2}pQp^T + cp + a$$

where $Q \in \mathbb{R}^{4\times 4}$ is a square matrix defined by the transition probabilities of the opponent q_1, q_2, q_3, q_4 as follows:

$$Q = \begin{bmatrix} 0 & -(q_1-q_3)(Pq_2-P-Tq_4) & (q_1-q_2)(Pq_3-Sq_4) & (q_1-q_4)(Sq_2-S-Tq_3) \\ -(q_1-q_3)(Pq_2-P-Tq_4) & 0 & (q_2-q_3)(Pq_1-P-Rq_4) & -(q_3-q_4)(Rq_2-R-Tq_1+T) \\ (q_1-q_2)(Pq_3-Sq_4) & (q_2-q_3)(Pq_1-P-Rq_4) & 0 & (q_2-q_4)(Rq_3-Sq_1+S) \\ (q_1-q_4)(Sq_2-S-Tq_3) & -(q_3-q_4)(Rq_2-R-Tq_1+T) & (q_2-q_4)(Rq_3-Sq_1+S) & 0 \end{bmatrix},$$

$c \in \mathbb{R}^{4\times 1}$ is similarly defined by:

$$c = \begin{bmatrix} q_1(Pq_2-P-Tq_4) \\ -(q_3-1)(Pq_2-P-Tq_4) \\ -Pq_1q_2+Pq_2q_3+Pq_2-Pq_3+Rq_2q_4-Sq_2q_4+Sq_4 \\ -Rq_2q_4+Rq_4+Sq_2q_4-Sq_2-Sq_4+S+Tq_1q_4-Tq_3q_4+Tq_3-Tq_4 \end{bmatrix},$$

and $a = -Pq_2 + P + Tq_4$.

The same process is done for the denominator. □

Numerical simulations have been carried out to validate the result. The simulated utility, which is denoted as $U_q(p)$, has been calculated with APL. For smoothing the simulated results the utility has been estimated in a tournament of 500 turns and 200 repetitions. Figure 5.2 shows two examples demonstrating that the formulation of Theorem 1 successfully captures the simulated behaviour.

Theorem 1 can be extended to consider multiple opponents. The IPD is commonly studied in tournaments and/or Moran Processes [156] where a strategy interacts with a number of opponents. The payoff of a player in such interactions is given by the average payoff the player received against each opponent. More specifically the expected utility of a memory-one strategy against N opponents is given by Theorem 2.

Theorem 2. *The expected utility of a memory-one strategy $p \in \mathbb{R}^4_{[0,1]}$ against a group*

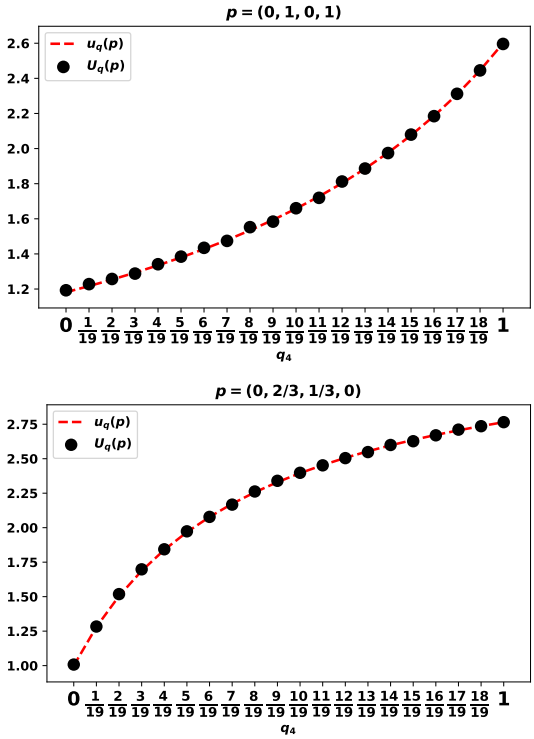


Figure 5.2: Simulated and empirical utilities for $p = (0, 1, 0, 1)$ and $p = (0, \frac{2}{3}, \frac{1}{3}, 0)$ against $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, q_4)$ for $q_4 \in \{0, \frac{1}{19}, \frac{2}{19}, \ldots, \frac{18}{19}, 1\}$. $u_q(p)$ is the theoretic value given in Theorem 1, and $U_q(p)$ is simulated numerically using APL.

of opponents $\{q^{(1)}, q^{(2)}, \ldots, q^{(N)}\}$*, denoted as* $\frac{1}{N}\sum_{i=1}^{N} {u_q}^{(i)}(p)$*, is given by:*

$$\frac{1}{N}\sum_{i=1}^{N} {u_q}^{(i)}(p) = \frac{1}{N} \frac{\sum_{i=1}^{N}(\frac{1}{2}pQ^{(i)}p^T + c^{(i)}p + a^{(i)}) \prod_{\substack{j=1 \\ j\neq i}}^{N} (\frac{1}{2}p\bar{Q}^{(j)}p^T + \bar{c}^{(j)}p + \bar{a}^{(j)})}{\prod_{i=1}^{N}(\frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)})}. \tag{5.11}$$

Equation (5.11) is the average score (using Equation (5.3)) against the set of opponents.

Similar to the previous result, the formulation of Theorem 2 is validated using numerical simulations where the 10 memory-one strategies described in [274] have been used as the opponents. Figure 5.3 shows that the simulated behaviour has been captured successfully.

The list of strategies from [274] was also used to check whether the utility against a group of strategies could be captured by the utility against the mean opponent.

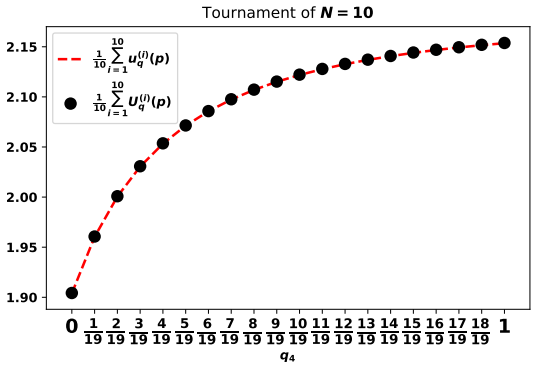


Figure 5.3: The utilities of memory-one strategies $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, p_4)$ for $p_4 \in \{0, \frac{1}{19}, \frac{2}{19}, \ldots, \frac{18}{19}, 1\}$ against the 10 memory-one strategies described in [274]. $\frac{1}{10}\sum_{i=1}^{10} u_q^{(i)}(p)$ is the theoretic value given in Theorem 1, and $\frac{1}{10}\sum_{i=1}^{10} U_q^{(i)}(p)$ is simulated numerically using APL.

Thus whether condition (5.12) holds. However condition (5.12) fails, as shown in Figure 5.4.

$$\frac{1}{N}\sum_{i=1}^{N} {u_q}^{(i)}(p) = u_{\frac{1}{N}\sum_{i=1}^{N} q^{(i)}}(p), \tag{5.12}$$

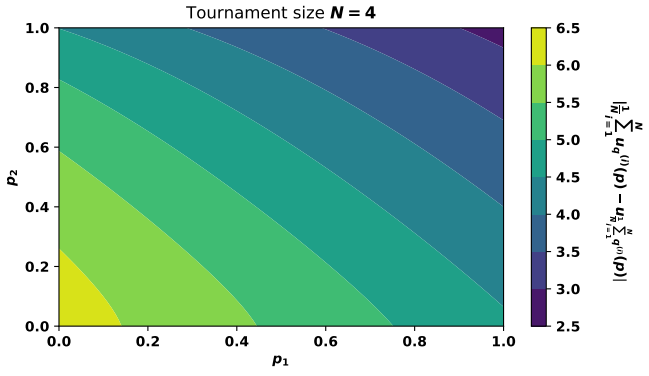


Figure 5.4: The difference between the average utility against the opponents from [274] and the utility against the average player of the strategies in [274] of a player $p = (p_1, p_2, p_1, p_2)$. A positive difference indicates that condition (5.12) does not hold.

Theorem 2 which allows for the utility of a memory-one strategy against any number of opponents to be estimated without simulating the interactions is the main result used in the rest of this Chapter. In section 5.3 it is used to define best response memory-one strategies, in section 5.4 to define best response reactive strategies and in section 5.6 to explore the conditions under which defection dominates cooperation.

5.3 Best responses to memory-one players

This section focuses on *memory-one best response* strategies. A best response is a strategy which corresponds to the most favourable outcome (Chapter 1), thus a memory-one best response to a set of opponents $q^{(1)}, q^{(2)}, \ldots, q^{(N)}$ corresponds to a strategy p^* for which Equation (5.11) is maximised. This is considered as a multi dimensional optimisation problem given by:

$$\begin{aligned} &\max_p : \sum_{i=1}^{N} {u_q}^{(i)}(p) \\ &\text{such that} : p \in \mathbb{R}_{[0,1]} \end{aligned} \tag{5.13}$$

Optimising this particular ratio of quadratic forms is not trivial. It can be verified empirically for the case of a single opponent that there exists at least one point for which the definition of concavity does not hold.

A function $f(x)$ is concave on an interval $[a, b]$ if, for any two points $x_1, x_2 \in [a, b]$ and any $\lambda \in [0, 1]$,

$$f(\lambda x_1 + (1-\lambda)x_2) \geq \lambda f(x_1) + (1-\lambda) f(x_2). \tag{5.14}$$

Let f be $u_{(\frac{1}{3},\frac{1}{3},\frac{1}{3},\frac{1}{3})}$. For $x_1 = (\frac{1}{4}, \frac{1}{2}, \frac{1}{5}, \frac{1}{2}), x_2 = (\frac{8}{10}, \frac{1}{2}, \frac{9}{10}, \frac{7}{10})$ and $\lambda = 0.1$. Direct substitution in to the left hand side of Equation (5.14) gives,

$$\begin{aligned} f(\lambda x_1 + (1-\lambda)x_2) &= u_{(\frac{1}{3},\frac{1}{3},\frac{1}{3},\frac{1}{3})}\left(0.1\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{5}, \frac{1}{2}\right) + 0.9\left(\frac{8}{10}, \frac{1}{2}, \frac{9}{10}, \frac{7}{10}\right)\right) \\ &= 1.485 \end{aligned}$$

and in to the right hand side,

$$\begin{aligned} \lambda f(x_1) + (1-\lambda) f(x_2) &= 0.1 \times u_{(\frac{1}{3},\frac{1}{3},\frac{1}{3},\frac{1}{3})}\left(\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{5}, \frac{1}{2}\right)\right) + 0.9 \times u_{(\frac{1}{3},\frac{1}{3},\frac{1}{3},\frac{1}{3})}\left(\left(\frac{8}{10}, \frac{1}{2}, \frac{9}{10}, \frac{7}{10}\right)\right) \\ &= 0.1 \times 1.790 + 0.9 \times 1.457 \\ &= 1.490. \end{aligned}$$

Thus Equation (5.14) does not hold, and thus $u_{(\frac{1}{3},\frac{1}{3},\frac{1}{3},\frac{1}{3})}$ is not concave.

The non concavity of $u(p)$ indicates multiple local optimal points. This is also intuitive. The best response against a **Cooperator**, $q = (1, 1, 1, 1)$, is a **Defector** $p^* = (0, 0, 0, 0)$. The strategies $p = (\frac{1}{2}, 0, 0, 0)$ and $p = (\frac{1}{2}, 0, 0, \frac{1}{2})$ are also best responses. The approach taken here is to introduce a compact way of constructing the discrete candidate set of all local optimal points, and evaluating the objective function Equation (5.11). This gives the best response memory-one strategy. The approach is given in Theorem 3.

Theorem 3. *The optimal behaviour of a memory-one strategy $p^* \in \mathbb{R}^4_{[0,1]}$ against a set of N opponents $\{q^{(1)}, q^{(2)}, \ldots, q^{(N)}\}$ for $q^{(i)} \in \mathbb{R}^4_{[0,1]}$ is given by:*

$$p^* = \text{argmax} \sum_{i=1}^{N} u_q(p), \ p \in S_q.$$

The set S_q is defined as all the possible combinations of:

$$S_q = \left\{ p \in \mathbb{R}^4 \left| \begin{array}{ll} \bullet & p_j \in \{0,1\} \quad \textit{and} \quad \frac{d}{dp_k} \sum_{i=1}^{N} u_q^{(i)}(p) = 0 \quad \textit{for all} \quad j \in J \quad \& \quad k \in K \quad \textit{for all} \quad J, K \\ & \qquad\qquad \textit{where} \quad J \cap K = \emptyset \quad \textit{and} \quad J \cup K = \{1,2,3,4\}. \\ \bullet & p \in \{0,1\}^4 \end{array} \right. \right\}. \quad (5.15)$$

Proof. The optimal behaviour of a memory-one strategy player $p^* \in \mathbb{R}^4_{[0,1]}$ against a set of N opponents $\{q^{(1)}, q^{(2)}, \ldots, q^{(N)}\}$ for $q^{(i)} \in \mathbb{R}^4_{[0,1]}$ is established by:

$$p^* = \text{argmax} \left(\sum_{i=1}^{N} u_q(p) \right), \ p \in S_q,$$

where S_q is given by Equation (5.15).

The optimisation problem of (5.13) can be written as:

$$\begin{aligned} \max_p : & \sum_{i=1}^{N} {u_q}^{(i)}(p) \\ \text{such that} : p_i \leq 1 \text{ for } & \in \{1,2,3,4\} \\ -p_i \leq 0 \text{ for } & \in \{1,2,3,4\} \end{aligned} \quad (5.16)$$

The optimisation problem has two inequality constraints and regarding the optimality this means that:

- either the optimum is away from the boundary of the optimization domain, and so the constraints plays no role;
- or the optimum is on the constraint boundary.

Thus, the following three cases must be considered:

Case 1: The solution is on the boundary and any of the possible combinations for $p_i \in \{0,1\}$ for $i \in \{1,2,3,4\}$ are candidate optimal solutions.

Case 2: The optimum is away from the boundary of the optimization domain and the interior solution p^* necessarily satisfies the condition $\frac{d}{dp}\sum_{i=1}^{N} u_q(p^*) = 0$.

Case 3: The optimum is away from the boundary of the optimization domain but some constraints are equalities. The candidate solutions in this case are any combinations of $p_j \in \{0,1\}$ and $\frac{d}{dp_k}\sum_{i=1}^{N} u_q^{(i)}(p) = 0$ for all $j \in J$ & $k \in K$ for all J, K where $J \cap K = \emptyset$ and $J \cup K = \{1,2,3,4\}$.

Combining cases 1-3 a set of candidate solution is constructed as:

$$S_q = \left\{ p \in \mathbb{R}^4 \;\middle|\; \begin{array}{l} \bullet \quad p_j \in \{0,1\} \quad \text{and} \quad \dfrac{d}{dp_k}\displaystyle\sum_{i=1}^{N} u_q^{(i)}(p) = 0 \quad \text{for all} \quad j \in J \quad \& \quad k \in K \quad \text{for all} \quad J, K \\ \qquad\qquad \text{where} \quad J \cap K = \emptyset \quad \text{and} \quad J \cup K = \{1,2,3,4\}. \\ \bullet \quad p \in \{0,1\}^4 \end{array} \right\}.$$

The derivative of $\sum_{i=1}^{N} u_q^{(i)}(p)$ calculated using the following property (see [10] for details):

$$\frac{dxAx^T}{dx} = 2Ax. \tag{5.17}$$

Using property (5.17):

$$\frac{d}{dp}\frac{1}{2}pQp^T + cp + a = pQ + c \text{ and } \frac{d}{dp}\frac{1}{2}p\bar{Q}p^T + \bar{c}p + \bar{a} = p\bar{Q} + \bar{c}. \tag{5.18}$$

Note that the derivative of cp is c and the constant disappears. Combining these it can be proven that:

$$\frac{d}{dp}\sum_{i=1}^{N} u_q^{(i)}(p) = \sum_{i=1}^{N} \frac{\frac{d}{dp}(\frac{1}{2}pQ^{(i)}p^T + c^{(i)}p + a^{(i)})(\frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)}) - \frac{d}{dp}(\frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)})(\frac{1}{2}pQ^{(i)}p^T + c^{(i)}p + a^{(i)})}{(\frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)})^2}$$
$$= \sum_{i=1}^{N} \frac{(pQ^{(i)} + c^{(i)}+)(\frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)})}{(\frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)})^2} - \frac{(p\bar{Q}^{(i)} + \bar{c}^{(i)})(\frac{1}{2}pQ^{(i)}p^T + c^{(i)}p + a^{(i)})}{(\frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)})^2}$$

For $\frac{d}{dp}\sum_{i=1}^{N} u_q(p)$ to equal zero then:

$$\sum_{i=1}^{N}\left(pQ^{(i)} + c^{(i)}\right)\left(\frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)}\right) - \left(p\bar{Q}^{(i)} + \bar{c}^{(i)}\right)\left(\frac{1}{2}pQ^{(i)}p^T + c^{(i)}p + a^{(i)}\right) = 0, \quad while \tag{5.19}$$

$$\sum_{i=1}^{N} \frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)} \neq 0. \tag{5.20}$$

The optimal solution to Equation 5.13 is the point from S_q for which the utility is maximised.

□

Note that there is no immediate way to find the zeros of $\frac{d}{dp}\sum_{i=1}^{N} u_q(p)$;

$$\frac{d}{dp}\sum_{i=1}^{N} u_q{}^{(i)}(p) =$$
$$= \sum_{i=1}^{N} \frac{\left(pQ^{(i)} + c^{(i)}\right)\left(\frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)}\right) - \left(p\bar{Q}^{(i)} + \bar{c}^{(i)}\right)\left(\frac{1}{2}pQ^{(i)}p^T + c^{(i)}p + a^{(i)}\right)}{\left(\frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)}\right)^2} \tag{5.21}$$

For $\frac{d}{dp}\sum_{i=1}^{N} u_q(p)$ to equal zero then:

$$\sum_{i=1}^{N}\left(\left(pQ^{(i)} + c^{(i)}\right)\left(\frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)}\right) - \left(p\bar{Q}^{(i)} + \bar{c}^{(i)}\right)\left(\frac{1}{2}pQ^{(i)}p^T + c^{(i)}p + a^{(i)}\right)\right) = 0, \quad while \tag{5.22}$$

$$\sum_{i=1}^{N} \frac{1}{2}p\bar{Q}^{(i)}p^T + \bar{c}^{(i)}p + \bar{a}^{(i)} \neq 0. \tag{5.23}$$

Finding best response memory-one strategies, more specifically constructing the subset

S_q, can be done analytically. The points for any or all of $p_i \in \{0, 1\}$ for $i \in \{1, 2, 3, 4\}$ are trivial, and finding the roots of the partial derivatives which are a set of polynomial equations (Equation (5.22)) is feasible using resultant theory [153]; however, for large systems building the resultant quickly becomes intractable.

Nevertheless, there are constrained versions of problem (5.13) for which calculating the resultant is efficient and a best response strategy can be identified explicitly. A constrained version is that of reactive strategies. Section 5.4 presents best response reactive strategies, and demonstrates the usage of resultant theory in identifying best responses.

5.4 Reactive strategies & Resultant theory

Reactive strategies are a subset of memory-one strategies discussed in Chapter 2. Well known reactive strategies include Tit For Tat and Generous Tit For Tat. As a reminder, reactive strategies only take into account the opponent's previous moves, and thus can be described as $p = (p_1, p_2) \in R^2_{[0,1]}$.

Best response reactive strategies are incorporated in the formulation of this Chapter by adding two extra constraints to the optimisation problem of (5.13),

$$\begin{aligned} \max_p : & \sum_{i=1}^{N} u_q(p) \\ \text{such that} : & \; p_1 = p_3 \\ & \; p_2 = p_4 \\ & \; p_1, p_2 \in \mathbb{R}_{[0,1]}. \end{aligned} \tag{5.24}$$

and a best response reactive strategy to a set of opponents N opponents $\{q^{(1)}, q^{(2)}, \ldots, q^{(N)}\}$ is given by Lemma 4.

Lemma 4. *The optimal behaviour of a reactive strategy $p^* \in \mathbb{R}^2_{[0,1]}$ against a set of N opponents $\{q^{(1)}, q^{(2)}, \ldots, q^{(N)}\}$ for $q^{(i)} \in \mathbb{R}^2_{[0,1]}$ is given by:*

$$p^* = \text{argmax} \sum_{i=1}^{N} u_q(p), \; p \in S_q.$$

The set S_q is defined as all the possible combinations of:

$$S_q = \left\{ p \in \mathbb{R}^2 \left| \begin{array}{ll} \bullet & p_j \in \{0,1\} \quad and \quad \frac{d}{dp_k} \sum_{i=1}^{N} u_q^{(i)}(p) = 0 \quad for\ all \quad j \in J \quad \& \quad k \in K \quad for\ all \quad J, K \\ & \qquad\qquad where \quad J \cap K = \emptyset \quad and \quad J \cup K = \{1,2\}. \\ \bullet & p \in \{0,1\}^2 \end{array} \right. \right\}. \quad (5.25)$$

Note that $\frac{d}{dp} \sum_{i=1}^{N} u_q^{(i)}(p) = 0$ corresponds to a system of 2 polynomials of 2 variables, corresponding to the partial derivatives over p_1 and p_2. Solving systems of 2 polynomials of 2 variables can be done analytically. The approach taken here to extract the roots from the partial derivatives is to use resultants.

5.4.1 Resultant theory

The *resultant* of two polynomials is a polynomial expression of their coefficients which is equal to zero if and only if the polynomials have a common root.

More specifically given a polynomial,

$$f(x) = a_n x^n + a_{(n-1)} x^{(n-1)} + \cdots + a_1 x + a_0$$

of degree n with roots $\alpha_i, i = 1, \ldots, n$ and a polynomial

$$g(x) = b_m x^m + b_{(m-1)} x^{(m-1)} + \cdots + b_1 x + b_0$$

of degree m with roots $\beta_j, j = 1, \ldots, m$, the resultant denoted as $R(f, g)$ and also called the eliminant [256], is defined by

$$R(f,g) = a_n^m b_m^n \prod_{(i=1)}^{n} \prod_{(j=1)}^{m} (\alpha_i - \beta_j). \quad (5.26)$$

Interestingly, the resultant can also be expressed as the determinant of matrices such as Sylvester's, Bezout's and Macaulay's. For systems of 2 polynomials the resultant is commonly expressed as the determinant of Sylvester's matrix [15]. The Sylvester matrix associated with f and g is the $(n + m) \times (n + m)$ matrix constructed as described by

Algorithm 5.1.

Algorithm 5.1: Construction of Sylvester matrix [15]

if $n > 0$ **then**

the 1^{st} row is $\begin{pmatrix} a_m & a_{m-1} & \cdots & a_1 & a_0 & 0 & \cdots & 0 \end{pmatrix}$;

for $i \leftarrow 1$ **to** $n-1$ **do**

the i^{th} is the previous row shifted one column to the right; the other entries of the row are 0

;

if $m > 0$ **then**

the $(n+1)^{\text{th}}$ row is: $\begin{pmatrix} b_m & b_{m-1} & \cdots & b_1 & b_0 & 0 & \cdots & 0 \end{pmatrix}$;

for $i \leftarrow n+1$ **to** $(m+n)-1$ **do**

the i^{th} is the previous row shifted one column to the right; the other entries of the row are 0

As an example consider the case of $m = 4$ and $n = 3$. The Sylvester matrix denoted as $S_{f,g}$ is given by,

$$S_{f,g} = \begin{pmatrix} a_4 & a_3 & a_2 & a_1 & a_0 & 0 & 0 \\ 0 & a_4 & a_3 & a_2 & a_1 & a_0 & 0 \\ 0 & 0 & a_4 & a_3 & a_2 & a_1 & a_0 \\ b_3 & b_2 & b_1 & b_0 & 0 & 0 & 0 \\ 0 & b_3 & b_2 & b_1 & b_0 & 0 & 0 \\ 0 & 0 & b_3 & b_2 & b_1 & b_0 & 0 \\ 0 & 0 & 0 & b_3 & b_2 & b_1 & b_0 \end{pmatrix} \tag{5.27}$$

and,

$$|S_{f,g}| = R(f,g).$$

The resultant can verify that the system has a root, but also can be used to extract

the roots. In a system of 2 polynomial equations in 2 variables, the resultant can be defined over one variable whereas the second one is kept as a coefficient. It is used in this Chapter to analytically solve for the roots of the derivative of the utility, and thus explicitly identify the best response reactive strategy against a given set of memory-one opponents.

The open source package [203] is used to calculate the Sylvester matrix and subsequently its determinant. An example is demonstrated in Figure 5.5. The approach demonstrated in Figure 5.5 is used to find the roots of the partial derivatives of $\frac{d}{dp}\sum_{i=1}^{N} u_q^{(i)}(p)$ and the candidate set of solutions S_q is constructed as defined in Equation (5.25).

```
>>> import sympy as sym
>>> from sympy.polys import subresultants_qq_zz
>>> p_1, p_2 = sym.symbols('p_1, p_2')

>>> f = p_1 ** 2 + p_1 * p_2 + 2 * p_1 + p_2 - 1
>>> g = p_1 ** 2 + 3 * p_1 - p_2 ** 2 + 2 * p_2 - 1
>>> matrix = subresultants_qq_zz.sylvester(f, g, p_2)
>>> matrix
Matrix([
[p_1 + 1, p_1**2 + 2*p_1 - 1,                 0],
[      0,            p_1 + 1, p_1**2 + 2*p_1 - 1],
[     -1,                  2, p_1**2 + 3*p_1 - 1]])
>>> matrix.det().factor()
-p_1*(p_1 - 1)*(p_1 + 3)
```

Figure 5.5: Example code for calculating the Sylvester matrix associated with $f = p_1^2 + p_1p_2 + 2p_1 + p_2 - 1$ and $g = p_1^2 + 3p_1 - p_2^2 + 2p_2 - 1$ using [203]. The matrix is calculated for p_2 whilst p_1 is handled as a coefficient, and thus the determinant is expressed in p_1. In order for the system to have a common root, p_1 must be $\in \{-3, 0, 1\}$. By substituting these values of p_1, each at a time, in f and g gives the roots for p_2.

A bespoke package has been developed to carry out the calculations for this Chapter. The package is called `opt_mo` and an example of how S_q is calculated against a given opponent $q = (0.513, 0.773, 0.870, 0.008)$ is given by Figure 5.6.

Once S_q is calculated then defining the best response is trivial. Figure 5.7 demonstrates how this is done using `opt_mo`, and the result is validated by Figure 5.8.

Sylvester's formulation can only handle systems of 2 polynomials, however, the multivariate resultants can be calculated for n homogeneous polynomials in n variables. A number of multivariate resultants can be found in the literature such as Dixon's [245] resultant and Macaulay's [186] resultant.

```
>>> import opt_mo
>>> import numpy as np
>>> import axelrod as axl

>>> axl.seed(14)
>>> opponents = [np.random.random(4)]
>>> opponents
[array([0.51394334, 0.77316505, 0.87042769, 0.00804695])]

>>> candidate_set = opt_mo.reactive_best_response.get_candidate_reactive_best_responses(
...     opponents
... )
>>> candidate_set
{(0.913428410721382+0j), 0.6964731896521483, 0.2775453690890986, 0, 1}
```

Figure 5.6: Code example of calculating S_q for a given opponent. The function `reactive_best_response.get_candidate_reactive_best_responses` retrieves the set S_q for a reactive strategy against a list of opponents. The set includes 0, 1 and to roots of the partial derivatives 0.277 and 0.696. An imaginary solution has also been calculated, however, it is ignore in the next step which calculates the best response.

```
>>> import numpy as np
>>> import opt_mo

>>> opponents = [np.array([0.51394334, 0.77316505, 0.87042769, 0.00804695])]
>>> candidate_set = opt_mo.reactive_best_response.get_candidate_reactive_best_responses(
...     opponents
... )

>>> opt_p1, opt_p2, score = opt_mo.reactive_best_response.get_argmax(
...     opponents, candidate_set
... )
>>> opt_p1, opt_p2
(0, 0.6964731842832972)
```

Figure 5.7: Code example of estimating the best response reactive strategy from a given S_q set and a given list of opponents.

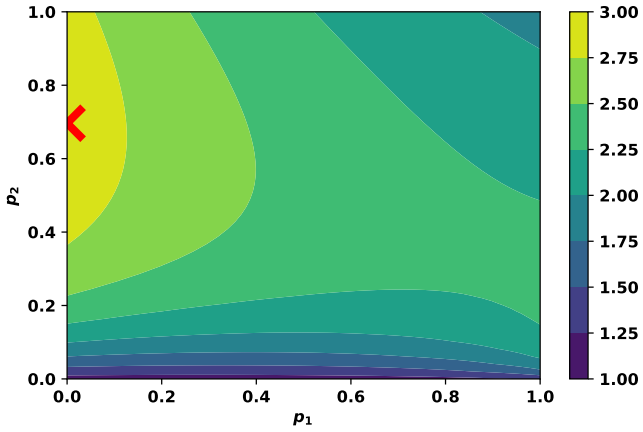


Figure 5.8: The utility of a $p = (p_1, p_2)$ reactive player against $q = (0.513, 0.773, 0.870, 0.008)$ for changing values of p_1 and p_2. The point marked with X is the point identified as the best response.

Project [203] which was used to construct Sylvester's resultant is called SymPy and it is the Pythonic package for symbolic mathematics. However, the project did not include the feature to calculate multivariate resultants. As part of this Chapter the source code for constructing both Dixon's and Macaulay's resultants was developed and was integrated into SymPy. Figure 5.9 shows the pull request made to SymPy for integrating the source code to their codebase. Figure 5.10 demonstrates an example of using [203] to calculate Dixon's resultant.

```
>>> from sympy.polys.multivariate_resultants import DixonResultant
>>> p_1, p_2 = sym.symbols('p_1, p_2')

>>> f = p_1 + p_2
>>> g = p_1 ** 2 + p_2 ** 3
>>> h = p_1 ** 2 + p_2

>>> dixon = DixonResultant(variables=[p_1, p_2], polynomials=[f, g, h])
>>> poly = dixon.get_dixon_polynomial()
>>> matrix = dixon.get_dixon_matrix(polynomial=poly)
>>> matrix
Matrix([
[ 0,  0, -1,  0, -1],
[ 0, -1,  0, -1,  0],
[-1,  0,  1,  0,  0],
[ 0, -1,  0,  0,  1],
[-1,  0,  0,  1,  0]])

>>> matrix.det()
0
```

Figure 5.10: Code example of using [203] to calculate Dixon's resultant. f, g and h have a common root $(x = 1, y = -1)$. The determinant of Dixon's matrix falls to zero which confirms that the system has a common root.

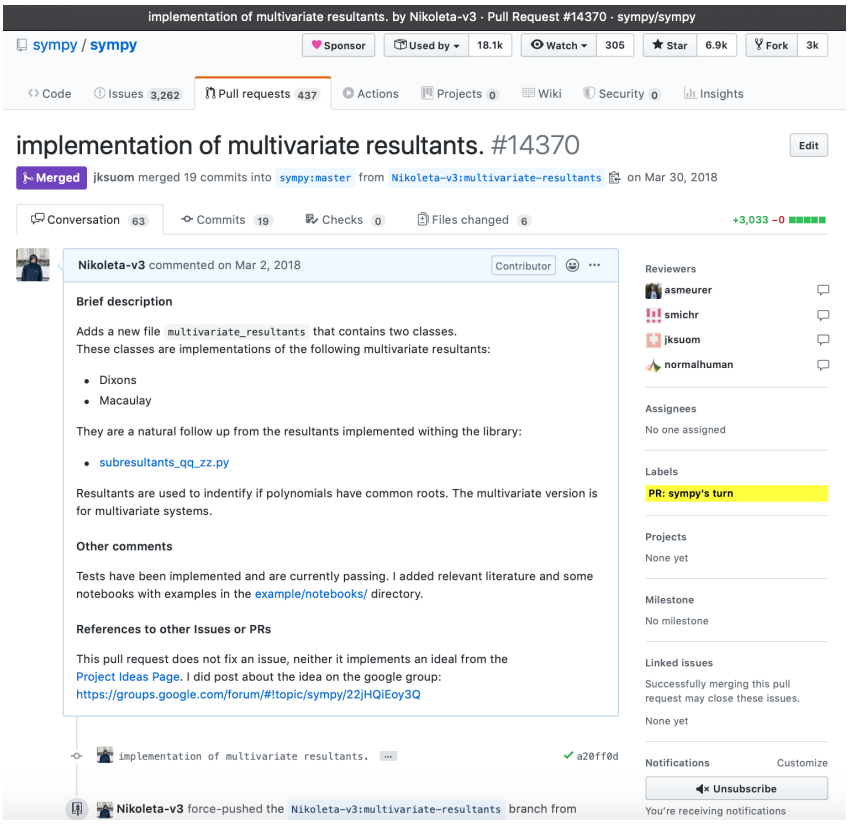


Figure 5.9: Screenshot of the pull request made to SymPy for integrating the source code of the multivariate resultants to the project's codebase. The details of the pull request as well as the conversation with the project's main contributors can be found at: `https://github.com/sympy/sympy/pull/14370`.

Multivariate resultants theoretically can be used to explicitly identify best response memory-one strategies by solving the system of 4 polynomials. However, as previously stated for large systems building the resultant quickly becomes intractable. As a result in section 5.5 a numerical approach was considered instead.

5.5 Numerical experiments

As briefly discussed in section 3.1, ZDs have been praised for their robustness against a single opponent. ZDs are evidence that extortion works in pairwise interactions. Their behaviour ensures that the strategies will never lose a game. However, this thesis argues that in multi opponent interactions, where the payoffs matter, strategies trying to exploit their opponents will suffer. Compared to ZDs, best response memory-one strategies which have a theory of mind of their opponents, utilise their behaviour in order to gain the most from their interactions. The question that arises then is whether best response strategies are optimal because they behave in an extortionate way. This is explored in section 5.5.3.

The other main finding presented in [237] was that short memory of the strategies was all that was needed. This thesis argues that the second limitation of ZDs in multi opponent interactions is that of their restricted memory. To demonstrate the effectiveness of memory in the IPD a best response longer-memory strategy against a given set of memory-one opponents is explored, and its performance is compared to that of a memory-one best response in section 5.5.4.

The results of this section rely on estimating best response memory-one strategies and understanding whether they behave in an extortionate way. Best responses will be estimated heuristically using Bayesian optimisation, which is described in section 5.5.1, and in order to investigate whether best responses behave in an extortionate matter the SSE method described in section 5.5.2 is used.

5.5.1 Bayesian optimisation

Bayesian optimisation is a global optimisation algorithm that has proven to outperform many other popular algorithms [149]. The algorithm builds a bayesian understanding of the objective function which is well suited to the potential multiple local optimas in the described search space of this Chapter. Differential evolution [277] was also considered, however, it was not selected due to Bayesian optimisation being computationally more

efficient.

As described in [86] Bayesian optimisation consists of two main components: a Bayesian statistical model for modelling the objective function, and an acquisition function for deciding where to sample next. The algorithm initially evaluates the objective according to a space-filling experimental design, often consisting of points chosen uniformly at random. They are used iteratively to allocate the remainder of a budget of I function evaluations, as shown in Algorithm 5.2.

Algorithm 5.2: Basic pseudo-code for Bayesian optimisation. As given in [86]

Place a Gaussian process prior on f;
Observe f at i_0 points according to an initial space-filling experimental design, set $i = i_0$;
while $i \leq I$ **do**
    Update the posterior probability distribution on f using all available data;
    Let x_i be a maximiser of the acquisition function over x, where the acquisition function is computed using the current posterior distribution;
    Observe $y_i = f(x_i)$;
    Increment i;
return either the point evaluated with the largest $f(x)$, or the point with the largest posterior mean.

The statistical model is invariably a Gaussian process. It provides a Bayesian posterior probability distribution that describes potential values for the objective function at a candidate point. Each time the objective function is observed at the new point the posterior distribution is updated.

The acquisition function measures the value that would be generated by evaluation of the objective function at a new point, based on the current posterior distribution over f. The most commonly used acquisition functions are:

- The expected improvement [150].
- The knowledge gradient [85].
- The entropy search and predictive entropy search [127].

As an example of the algorithm's usage consider the optimisation problem of (5.13). Figure 5.11 illustrates the change of the utility function over I. The algorithm is set to run for $I = 60$. After 60 function evaluations if the utility has not changed in the

last 10% of evaluations, then the algorithm runs for a further 20 evaluations. This is repeated until there is no change to the utility in the last 10% of evaluations.

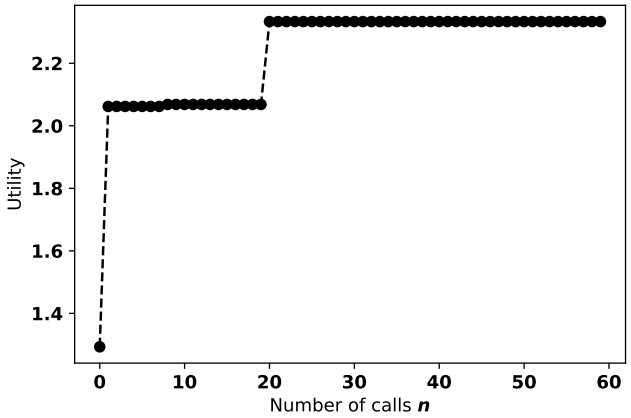


Figure 5.11: Utility over time of calls using Bayesian optimisation. The opponents are $q^{(1)} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ and $q^{(2)} = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. The best response obtained is $p^* = (0, \frac{11}{50}, 0, 0)$

5.5.2 SSE method

The SSE method is a linear algebraic approach that defines the closest ZD strategy to a given strategy p. This strategy is defined as x^* given by,

$$x^* = \left(C^T C\right)^{-1} C^T \bar{p} \tag{5.28}$$

where

$$\bar{p} = (p_1 - 1, p_2 - 1, p_3, p_4) \text{ and}$$

and

$$C = \begin{bmatrix} R - P & R - P \\ S - P & T - P \\ T - P & S - P \\ 0 & 0 \end{bmatrix}. \tag{5.29}$$

Once x^* is estimated the method calculates the squared norm of the remaining error

referred to as sum of squared errors of prediction (SSE):

$$\text{SSE} = \bar{p}^T\bar{p} - \bar{p}C\left(C^TC\right)^{-1}C^T\bar{p} = \bar{p}^T\bar{p} - \bar{p}Cx^* \tag{5.30}$$

The SSE is defined as how far a strategy is from behaving as a ZD. A small SSE implies ZD behaviour whereas a high SSE implies a non extortionate behaviour.

5.5.3 Best response memory-one strategies for $N = 2$

Bayesian optimisation was used to generate a data set of best response memory-one strategies for $N = 2$ opponents. The data set has been archived and is available at [101]. The data set contains two sets of best response memory-one strategies for each pair of opponents. These are:

- Best response memory-one strategies without self interactions.
- Best response memory-one strategies with self interactions.

In several evolutionary settings such as Moran Processes self interactions are key. Previous work has identified interesting results such as the appearance of self recognition mechanisms when training strategies using evolutionary algorithms in Moran processes [163]. This aspect of reinforcement learning can be done for best response memory-one strategies by incorporating the strategy itself in the objective function as shown in Equation (5.13). K is the number of self interactions that will take place.

$$\begin{aligned} &\max_p : \frac{1}{N}\sum_{i=1}^{N} {u_q}^{(i)}(p) + K u_p(p) \\ &\text{such that} : p \in \mathbb{R}_{[0,1]} \end{aligned} \tag{5.31}$$

For determining the memory-one best response with self interactions, an algorithmic approach is considered, called *best response dynamics*. The best response dynamics approach used in this manuscript is given by Algorithm 5.3.

Algorithm 5.3: Best response dynamics algorithm

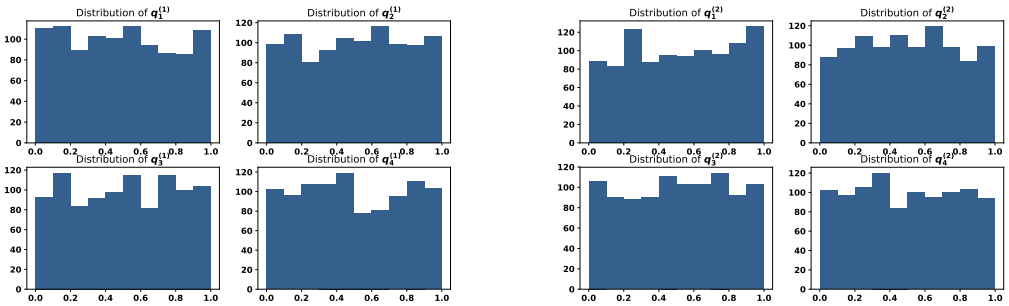
$p^{(t)} \leftarrow (1, 1, 1, 1)$;

while $p^{(t)} \neq p^{(t-1)}$ **do**

$p^{(t+1)} = \text{argmax}\frac{1}{N}\sum_{i=1}^{N} {u_q}^{(i)}(p^{(t)}) + K u_{p^{(t)}}(p^{(t)})$;

Using this approach it would be possible to create a memory-one best response strategy that updates on every generation of a Moran process to recalculate the optimal behaviour given the population.

The data set contains a total of 1,000 trials corresponding to 1,000 different instances of a best response strategy in tournaments with and without self interactions. For each trial a set of 2 opponents is randomly generated and the memory-one best responses against them are found. The probabilities q_i of the opponents are randomly generated and Figures 5.12a and 5.12b, show that they are uniformly distributed over the trials. Thus, the full space of possible opponents has been covered.



(a) Distributions of first opponents' probabilities.

(b) Distributions of second opponents' probabilities.

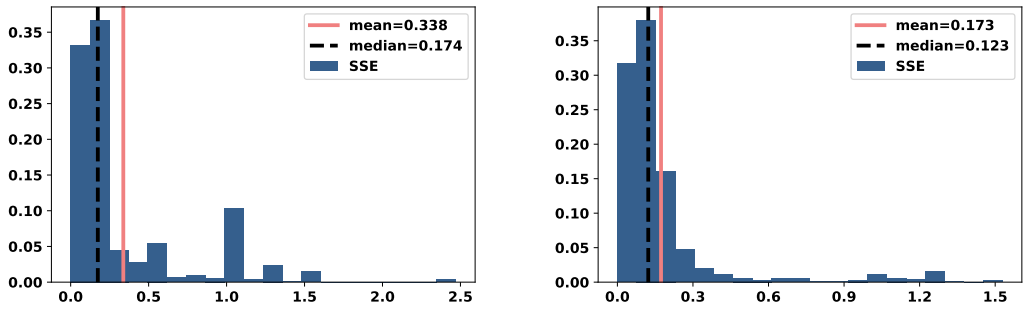
Figure 5.12: Distributions of opponents' probabilities.

The SSE method was applied to the best response memory-one strategies once the data set was generated. The distributions of SSE for the best response in tournaments ($N = 2$) with and without self interactions with ($K = 1$) are given in Figure 5.13. Moreover, a statistical summary of the SSE distributions is given in Table 5.1.

	mean	standard deviation	5%	50%	95%	max	median	skewness	kurtosis
Tournament without self interactions	0.34	0.40	0.028	0.17	1.05	2.47	0.17	1.87	3.60
Tournament with self interactions	0.17	0.23	0.01	0.12	0.67	1.53	0.12	3.42	1.92

Table 5.1: SSE of best response memory-one for $N = 2$.

For the best response in tournaments that do not include self interactions the distribution of SSE is skewed to the left, indicating that the best response does exhibit ZDs behaviour and so could be extortionate, however, the best response is not uniformly a ZDs. A positive measure of skewness and kurtosis, and a mean of 0.34 indicate a heavy tail to the right. Therefore, in several cases the strategy is not trying to extort its opponents. In [129] a similar behaviour is refereed to as the *partner strategy*. The



(a) SEE distribution for best response in tournaments without self interactions.

(b) SEE distribution for best response in tournaments with self interactions.

Figure 5.13: SEE distributions for best response in tournaments without and with self interactions for $N = 2$.

partner strategy aims to share the payoff for mutual cooperation, but it is ready to fight back when being exploited. The partner strategy was designed, but the best responses which are defined by their opponents seem to exhibit the same behaviour.

Similarly, when considering self interactions, the distribution of SSE for the best response strategy has skewness and kurtosis that indicate a heavy tail to the right. This indicates that memory-one strategies that interact with copies of themselves need to even more adaptable than ZDs, and aim for mutual cooperation as well as exploitation which is in line with the results of [129] where their strategy was designed to adapt and was shown to be evolutionary stable. The findings of this work show that an optimal strategy acts in the same way.

The difference between best responses in tournaments without and with self interactions is further explored by Figure 5.14. Though, no statistically significant differences have been found, from Figure 5.14, it seems that the best response that incorporate self interactions has a higher median p_2; which corresponds to the probability of cooperating after receiving a defection. Thus, they are more likely to forgive after being tricked. This is due to the fact that they could be playing against themselves, and they need to be able to forgive so that future cooperation can occur.

5.5.4 Longer memory best responses

This section focuses on the memory size of strategies. The effectiveness of memory in the IPD has been previously explored in the literature, as discussed in section 3.1, however, none of the previous works has compared the performance of longer-memory strategies to memory-one best responses.

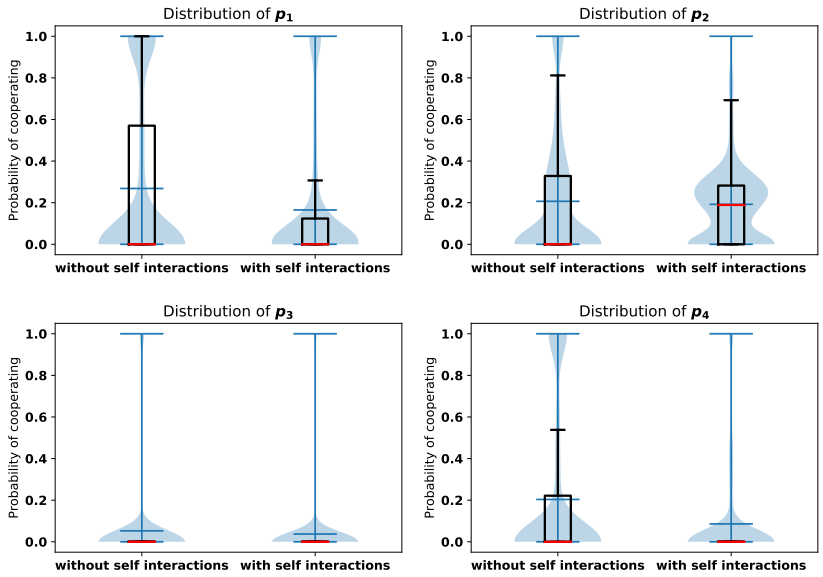


Figure 5.14: Distributions of p^* for best response in tournaments without and with self interactions. The medians, denoted as $\bar{p}^*$, for tournaments are $\bar{p}^* = (0, 0, 0, 0)$, and for evolutionary settings $\bar{p}^* = (0, 0.19, 0, 0)$.

The strategy used in this Chapter is one of the archetypes described in Chapter 2 called Gambler (introduced in [122]). As a reminder, Gambler is a stochastic version of a lookup table. It makes probabilistic decisions based on the opponent's n_1 first moves, the opponent's m_1 last moves and the player's m_2 last moves. In this Chapter Gambler with parameters: $n_1 = 2, m_1 = 1$ and $m_2 = 1$ is used as a longer-memory strategy.

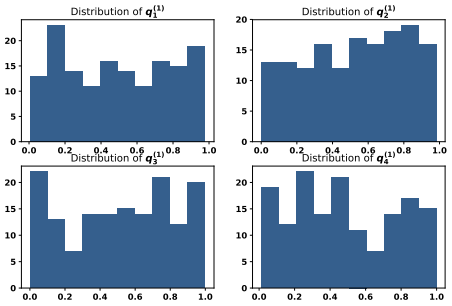
By considering the opponent's first two moves, the opponents last move and the player's last move, there are only 16 $(4 \times 2 \times 2)$ possible outcomes that can occur, furthermore, Gambler also makes a probabilistic decision of cooperating in the opening move. Thus, Gambler is a function $f : \{\text{C, D}\} \to [0, 1]_{\mathbb{R}}$. This can be hard coded as an element of $[0, 1]_{\mathbb{R}}^{16+1}$, one probability for each outcome plus the opening move. Hence, compared to (5.13), finding an optimal Gambler is a 17 dimensional problem given by:

$$\begin{aligned} \max_{p} : & \sum_{i=1}^{N} {U_q}^{(i)}(f) \\ \text{such that} : & f \in \mathbb{R}_{[0,1]}^{17} \end{aligned} \tag{5.32}$$

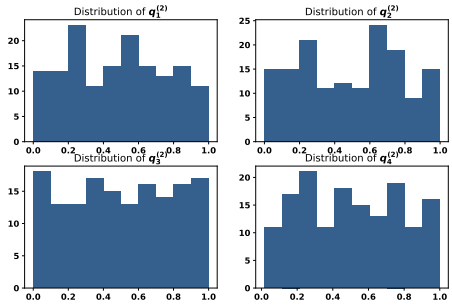
Note that Equation (5.11) can not be used here for the utility of Gambler, and actual

simulated players are used. This is done using a tournament of 500 turns and 200 repetitions, moreover, (5.32) is solved numerically using Bayesian optimisation.

Similarly to the previous section, a large data set has been generated with instances of an optimal Gambler and a memory-one best response, available at [101]. Estimating a best response Gambler (17 dimensions) is computational more expensive compared to a best response memory-one (4 dimensions). As a result, the analysis of this section is based on a total of 130 trials. For each trial two random opponents have been selected. The 130 pair of opponents are a sub set of the opponents used in section 5.5.3. The distributions of their transition probabilities are given in Figures 5.15a and 5.15a.



(a) Distributions of first opponents' probabilities for longer memory experiment.



(b) Distributions of second opponents' probabilities for longer memory experiment.

Figure 5.15: Distributions of opponents' probabilities for longer memory experiment.

The ratio between Gambler's utility and the best response memory-one strategy's utility has been calculated and its distribution in given in Figure 5.16. It is evident from Figure 5.16 that Gambler always performs as well as the best response memory-one strategy and often performs better. There are no points where the ratio value is less than 1, thus Gambler never performed less than the best response memory-one strategy and in places outperforms it. This seems to be at odd with the result of [237] that against a memory-one opponent having a longer memory will not give a strategy any advantage. However, against two memory-one opponents Gambler's performance is better than the optimal memory-one strategy. This is evidence that in the case of two opponents having a shorter memory is limiting.

5.6 Stability of defection

An additional theoretical result that is possible to obtain due to Theorem 2, is a condition for which in an environment of memory-one opponents defection is the stable

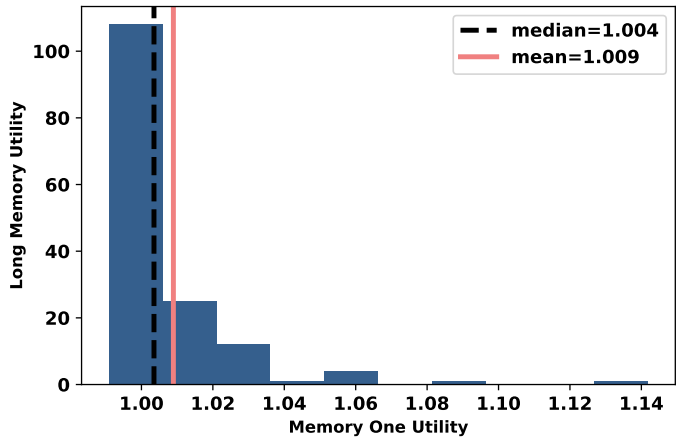


Figure 5.16: Utilities of Gambler and best response memory-one strategies for 130 different pair of opponents.

choice, based only on the coefficients of the opponents.

This result is obtained by evaluating the sign of Equation (5.11)'s derivative at $p = (0, 0, 0, 0)$. If at that point the derivative is negative, then the utility of a player only decreases if they were to change their behaviour, and thus defection at that point is stable.

Lemma 5. *In a tournament of N players $\{q^{(1)}, q^{(2)}, \ldots, q^{(N)}\}$ for $q^{(i)} \in \mathbb{R}^4_{[0,1]}$ defection is stable if the transition probabilities of the opponents satisfy conditions Equation (5.33) and Equation (5.34).*

$$\sum_{i=1}^{N} (c^{(i)T} \bar{a}^{(i)} - \bar{c}^{(i)T} a^{(i)}) \leq 0 \tag{5.33}$$

while,

$$\sum_{i=1}^{N} \bar{a}^{(i)} \neq 0 \tag{5.34}$$

Proof. For defection to be stable the derivative of the utility at the point $p = (0, 0, 0, 0)$ must be negative.

Substituting $p = (0, 0, 0, 0)$ in Equation (5.21) gives:

$$\left.\frac{d\sum_{i=1}^{N} {u_q}^{(i)}(p)}{dp}\right|_{p=(0,0,0,0)} = \sum_{i=1}^{N} \frac{(c^{(i)T}\bar{a}^{(i)} - \bar{c}^{(i)T}a^{(i)})}{(\bar{a}^{(i)})^2} \tag{5.35}$$

The sign of the numerator $\sum_{i=1}^{N}(c^{(i)T}\bar{a}^{(i)} - \bar{c}^{(i)T}a^{(i)})$ can vary based on the transition probabilities of the opponents. The denominator can not be negative, and otherwise is always positive. Thus the sign of the derivative is negative if and only if $\sum_{i=1}^{N}(c^{(i)T}\bar{a}^{(i)} - \bar{c}^{(i)T}a^{(i)}) \leq 0$. □

Consider a population for which defection is known to be stable. In that population all the members will over time adopt the same behaviour; thus in such population cooperation will never take over. This is demonstrated in Figure 5.17.

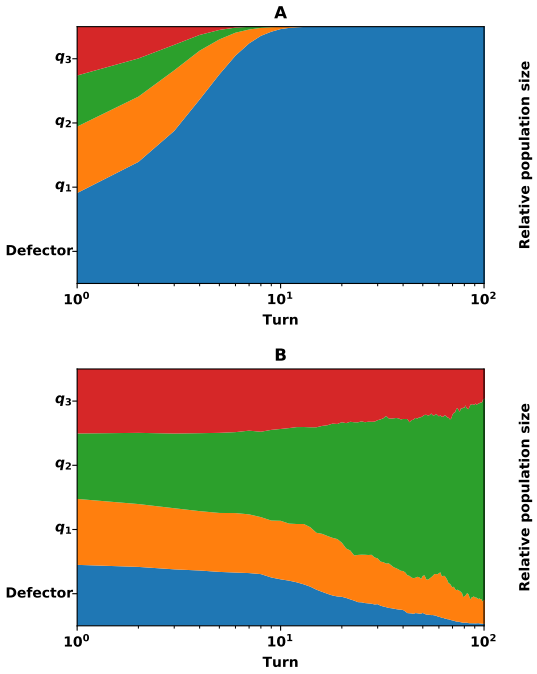


Figure 5.17: A. For $q_1 = (0.2219, 0.8707, 0.2067, 0.9186)$, $q_2 = (0.4884, 0.6117, 0.7659, 0.5184)$ and $q_3 = (0.2968, 0.1877, 0.0807, 0.7384)$, Equation (5.33) and Equation (5.34) hold and **Defector** takes over the population. B. For $q_1 = (0.9670, 0.5472, 0.9726, 0.7148)$, $q_2 = (0.6977, 0.2160, 0.9762, 0.0062)$ and $q_3 = (0.2529, 0.4349, 0.7738, 0.1976)$, Equation (5.33) fails and **Defector** does not take over the population.

5.7 Chapter summary

This Chapter has considered best response strategies in the IPD game, and more specifically, memory-one best responses. It has proven that:

- The utility of a memory-one strategy against a set of memory-one opponents can be written as a sum of ratios of quadratic forms (Theorem 2).
- There is a compact way of identifying a memory-one best response to a group of opponents through a search over a discrete set (Theorem 3).
- There is a compact way of identifying environment of memory-one opponents where defection is the stable choice based only on the coefficients of the opponents (Lemma 5).

Note that Theorem 3 does not only have game theoretic novelty, but also mathematical novelty of solving quadratic ratio optimisation problems where the quadratics are non concave. Additionally, Theorem 3 led to Lemma 4 which defined best response reactive strategies. Using the set of reactive strategies it was possible to demonstrate the usage of resultant theory in the search of best response strategies.

The empirical results of this Chapter were presented in section 5.5. The results relied on a bespoke data set of 1,000 pairs of memory-one opponents. For each pair of opponents two sets of best response memory-one strategies were estimated. Best responses that included and not self interactions. The behaviour of these best responses was investigated. More specifically, it was explored whether it was extortionate acts that made them the most favourable strategies. It was shown that it was not extortion but adaptability that allowed the strategies to gain the most from their interactions. In settings with self interactions there is some evidence that the best response strategy is more likely to forgive after being tricked.

Section 5.5 also explored the limitations of memory. The performance of best response memory-one strategy was compared to that of a Gambler with longer memory. The results concluded that the performance of memory-one strategies is limited by their memory in cases where they interact with multiple opponents. They can never score higher than a longer memory strategy.

By specifically exploring the entire space of memory-one strategies to identify the best strategy for a variety of situations, this Chapter has added to the literature casting

doubt on the effectiveness of ZDs, it has highlighted the importance of adaptability and provides a framework for the continued understanding of these important questions.

Chapter 6

Best response sequences in the Iterated Prisoner's Dilemma

The research reported in this Chapter has been carried out with:

Axelrod-Python library (APL) version: 4.2.0

Associated data set: [103]

Associated codebase: [95]

6.1 Introduction

In this Chapter best response strategies are explored in the form of static sequences of moves, in order to generate a large data set of best response sequences to a collection of opponents.

The data set is generated by considering best response sequences in finite IPD matches of 205 turns against 192 strategies available in the APL. These best response sequences are not obtained explicitly but instead are estimated heuristically using a genetic algorithm devised for this purpose.

The purpose of a large collection of best response sequences is to serve as training data in Chapter 7 which aims to train a recurrent neural network as an IPD strategy. In Chapter 7 the usage of the bespoke data set, which has been archived and made publicly available [103], is discussed in more details. This Chapter is structured as

follow:

- section 6.2 formalises the use of sequences to express a player in a finite IPD match.
- section 6.3 describes the genetic algorithm used to estimate best response sequences.
- section 6.4 details the process of generating best response sequences to a collection of 192 strategies.

6.2 Iterated Prisoner Dilemma Strategies as sequences

In a finite N round IPD match a player that does not react to their opponent can be defined by a sequence,

$$S \in \{C, D\}^n, \text{where } 1 \leq n \leq N. \tag{6.1}$$

Strategies that base their actions on sequences are already established in the literature [43], such as Periodic Player CD, Periodic Player DC, Periodic Player CCD and Periodic Player DDC [178, 208], or as referred to in [164] Cycler CD, Cycler DC, Cycler CCD and Cycler DDC. These are strategies that play a given sequences periodically, however, the strategies concerned with here play a given sequence only once, thus $n = N$.

As an example consider a match of 10 turns between the strategy $S = \{D, D, D, C, C, C, D, D, C, C\}$ and Cooperator. The match between the two strategies is captured by Table 6.1 where $U(s_1, s_2) \in \mathbb{R}^2$ is the average score per turn scored by strategies s_1 and s_2.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	$U(S, \text{Cooperator})$
S	D	D	D	C	C	C	D	D	C	C	4.0
Cooperator	C	C	C	C	C	C	C	C	C	C	1.5

Table 6.1: The interactions of a 10 turns match between $S = \{D, D, D, C, C, C, D, D, C, C\}$ and Cooperator as well as the average score per turn achieved by each strategy.

A sequence strategy S can play against strategies that react to the history, for example

against **Tit For Tat** as demonstrated by Table 6.2,

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	$U(S, \text{Tit For Tat})$
S	D	D	D	C	C	C	D	D	C	C	2.2
Tit For Tat	C	D	D	D	C	C	C	D	D	C	2.2

Table 6.2: The interactions and average score per turn of a 10 turns match between $S = \{D, D, D, C, C, C, D, D, C, C\}$ and **Tit For Tat**.

and against stochastic strategies such as **Random**. **Random** cooperates with a probability of 0.5 at each turn, and thus the actions of the strategy are not deterministic. **Random** is a strategy that does react to the history, however, the actions of stochastic strategies that react to the history also differ between repetitions even when the history of the match is the same. Tables 6.3 and 6.4 both capture a match between S and a **Random** player. For a match of 10 turns **Random** has a total of 2^{10} possible plays. In order to capture several different plays of stochastic strategies computer seeding, further details of this are given in section 6.4.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	$U(S, \text{Random})$
S	D	D	D	C	C	C	D	D	C	C	2.2
Random	D	D	C	C	D	C	D	C	C	D	2.2

Table 6.3: The interactions and average score per turn of a 10 turns match between $S = \{D, D, D, C, C, C, D, D, C, C\}$ and **Random**.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	$U(S, \text{Random})$
S	D	D	D	C	C	C	D	D	C	C	2.4
Random	C	D	D	C	C	C	D	D	C	C	1.9

Table 6.4: The interactions and average score per turn of a 10 turns match between $S = \{D, D, D, C, C, C, D, D, C, C\}$ and **Random**. The actions make by **Random** are different to that of Table 6.3.

As discussed in Chapters 1 and 5, a best response strategy is a strategy that achieves that most favourable outcome. Thus a best response sequence against a given opponent Q corresponds to a sequence S^* for which the average score per turn is maximised, as given in (6.2).

$$\max_{S^*} : U(S^*, Q) \tag{6.2}$$

Identifying best responses to some opponents can be a trivial problem. The optimal sequence against a Cooperator is in an all D sequence. In fact the sequence $\{\underbrace{D, \ldots, D}_{N}\}$ is a best response against any sequence player whose plays are independent of the history. However, for some strategies identifying best responses is a complex problem.

Additionally, there are strategies that have multiple best response sequences. For instance the strategy Adaptive introduced in [178]. Adaptive opens by playing a sequence of $\{\underbrace{C, \ldots, C}_{6}\}$ followed by a sequence of $\{\underbrace{D, \ldots, D}_{5}\}$. The strategy then proceeds to play either C or D depending on which action had a higher total score for the strategy (the total score is recalculated at each turn). A sequence maximises its average score against Adaptive by locking the strategy into unconditional cooperations following its opening 11 turns sequence while the sequence defects. In order for cooperation to be the most favourable action for Adaptive the strategy needs to achieve two mutual cooperations at its opening sequence. That is because the score achieved by cooperating $2 \times 3 + 4 \times 0 = 6$ is greater than the score achieved by defecting $1 \times 5 = 5$. Any sequence which incorporates two cooperations in the first 6 turns and defects thereafter is a best response sequence to Adaptive. Thus, there can be 2^6 best response sequences to the strategy, for example S_1^* and S_2^*,

$$S_1^* = \{C, D, D, D, D, C, D, D, D, D, D, D, D, D, D\}$$

$$S_2^* = \{D, D, C, C, D, D, D, D, D, D, D, D, D, D, D\}$$

where $U(S_1^*, \text{Adaptive}) = U(S_2^*, \text{Adaptive}) = 3.4$.

Due to identifying best response sequences to some opponents being a complex problem, and moreover, multiple best response sequences existing for some opponents the best response sequences are not manually identified. Instead a genetic algorithm is used to estimate them. A background on genetic algorithms as well as the details of the specific genetic algorithm devised for this Chapter are presented in the following section.

6.3 Genetic algorithm

A genetic algorithm (GA) is a heuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms. As stated in [293] GAs encode

a potential solution to a specific problem on a simple chromosome-like data structure, and apply recombination operators to these structures in such a way as to preserve critical information. GAs are often viewed as function optimisers, although the range of problems to which they have been applied is quite broad [137, 152, 298].

An implementation of a GA begins with a *population* P of potential solutions, a number of *generations* $G \in \mathbb{N}$ and a cut-off or *bottleneck* $b < |P|$. At each generation the algorithm scores and potentially removes each member of the population $p_i \in P$. This is done by using a mapping from a member of the population to an ordered set based on an evaluation function f, usually $f(p_i) \to \mathbb{R}$, and by only keeping the top b ranking members (or proportion of members) by score at the end of each generation. The rest of the member are discarded. By keeping the top ranked individuals critical information regarding the successful candidates is preserved and the population rebuilds on it by using a series of *crossovers* and *mutations*.

- During a crossover 2 members of the population are selected, and a new member is created based on combination of their "genes". The 2 selected members are commonly referred to as parents.

- Mutation is a probabilistic change that occurs to an individual, Mutation is commonly associated with a probability, denoted as p_m. p_m is the probability of a mutation happening, either to the individual or to each gene of the individual.

A diagrammatic representation of a generic GA is given in Figure 6.1.

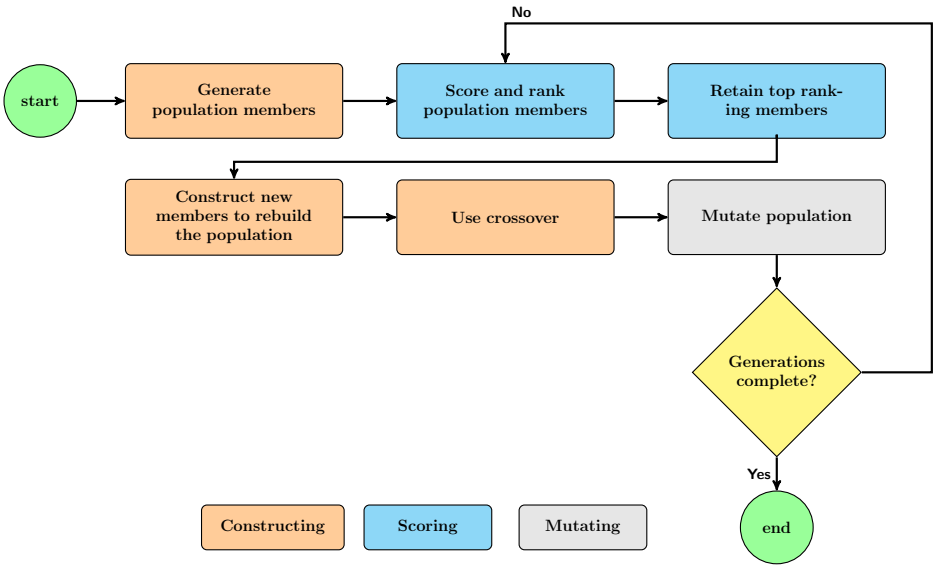


Figure 6.1: Generic flow diagram of a GA.

The purpose of a GA here is to estimate a best response sequence to a given opponent Q. Consequently, the members of the population correspond to sequences,

$$S_i \in P \text{ where } |S_i| = N$$

and the evaluation function corresponds to the average score per turn of a sequence against Q,

$$U(S_i, Q) \to \mathbb{R} \text{ for } S_i \in P.$$

More specifically, the exact GA used in this Chapter is given by Algorithm 6.1.

Algorithm 6.1: GA for estimating best response sequences to a given opponent Q.

Input: Q, N, b, p_m, G, K
Output: The populations at each generation and the members' scores
begin
  create initial population (Algorithm 6.2) of members S, where $|S| = N$ and $|P| = K$
  while $g_i < G$ **do**
    score each member based on $U(S_i, Q)$ for $S_i \in P$
    sort population based on scores
    keep b top members
    while $|P| < K$ **do**
      select 2 random members
      use members to create new member through crossover
      for *gene in new member* **do**
        mutate gene with probability p_m
      end
      add new member to population
    end
  end
end

The initial population is created using Algorithm 6.2.

Using a starting population of random guesses is a generally common approach in the GA literature [137]. However, there is efficiency in using non random starting populations [75, 231]. As discussed in section 6.2 the best response sequence to any strategy that does not react to the history is a Defector. Moreover, the best response sequence to several strategies include sequences mainly dominated by cooperation expect in the last turns. These are sequences that have been incorporated in the initial population.

Algorithm 6.2: Create initial population of individuals S

Input: K, N
Output: A population of size K.
begin
  set of cuts $\leftarrow$ K evenly spaced numbers over $[1, N]$
  for $c \in$ *set of cuts* **do**
    first new member $\leftarrow \{\underbrace{C, \ldots, C}_{c}, \underbrace{D, \ldots, D}_{N-c}\}$
    second new member $\leftarrow \{\underbrace{D, \ldots, D}_{c}, \underbrace{C, \ldots, C}_{N-c}\}$
    add both members to population
  end
end

More specifically, Algorithm 6.2 consider all the possible combinations of:

- $\{\underbrace{C, \ldots, C}_{c}, \underbrace{D, \ldots, D}_{N-c}\}$ for $c \in$ evenly spaced numbers over $[1, N]$ and
- $\{\underbrace{D, \ldots, D}_{c}, \underbrace{C, \ldots, C}_{N-c}\}$ for $c \in$ evenly spaced numbers over $[1, N]$.

The GA of Algorithm 6.1 has been implemented in the programming language Python and it has been organised into a open source package called `sequence_sensei` available at [95]. The properties of creating an initial population, crossover and mutation have been implemented as individual functions. The implementation of Algorithm 6.2 in the package is given by Figure 6.2.

```python
import numpy as np
def get_initial_population(half_size_of_population, sequence_length):
    """
    Generates an initial population of sequences. Note that the length
    of the population which is being generated is 2 * half_size_of_population.
    """
    cuts = np.linspace(1, sequence_length, half_size_of_population, dtype=int)
    sequences = []
    for cut in cuts:
        sequences.append(
            [1 for _ in range(cut)] + [0 for _ in range(sequence_length - cut)]
        )
        sequences.append(
            [0 for _ in range(cut)] + [1 for _ in range(sequence_length - cut)]
        )

    return sequences
```

Figure 6.2: Source code for the function `get_initial_population` implemented in `sequence_sensei` which is used to create an initial population of a given size.

Figure 6.3 gives an example of creating an initial population using the package. Note that the sequences are of 0s and 1s and not of IPD actions. The APL project can map binary number to actions such that $0 \to D$ and $1 \to C$. This is also demonstrated in Figure 6.3.

```
>>> import sequence_sensei as ss
>>> import numpy as np

>>> initial_population = ss.get_initial_population(
...     half_size_of_population=5, sequence_length=8
... )
>>> np.matrix(initial_population)
matrix([[1, 0, 0, 0, 0, 0, 0, 0],
        [0, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 0, 0, 0, 0, 0, 0],
        [0, 0, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 0, 0, 0, 0],
        [0, 0, 0, 0, 1, 1, 1, 1],
        [1, 1, 1, 1, 1, 1, 0, 0],
        [0, 0, 0, 0, 0, 0, 1, 1],
        [1, 1, 1, 1, 1, 1, 1, 1],
        [0, 0, 0, 0, 0, 0, 0, 0]])

>>> import axelrod as axl
>>> np.matrix([[axl.Action(gene) for gene in member] for member in initial_population])
matrix([[C, D, D, D, D, D, D, D],
        [D, C, C, C, C, C, C, C],
        [C, C, D, D, D, D, D, D],
        [D, D, C, C, C, C, C, C],
        [C, C, C, C, D, D, D, D],
        [D, D, D, D, C, C, C, C],
        [C, C, C, C, C, C, D, D],
        [D, D, D, D, D, D, C, C],
        [C, C, C, C, C, C, C, C],
        [D, D, D, D, D, D, D, D]], dtype=object)
```

Figure 6.3: Example of using `get_initial_population` to generate a population of $K = 10$ and $N = 8$.

In the GA, as given by Algorithm 6.1, the crossover occurs by randomly selecting two member of the population, while $|P| < K$, and randomly selecting a crossover point. Note that the crossover point is smaller than N. The new member initially inherits the genes to the left of the crossover point of the first parent, and to the right of the crossover point of the second parent.

For instance, given two member of the population $S_1 = \{C, C, C, C, C, C, C, C, C, C\}$ and $S_2 = \{C, D, C, D, C, D, C, D, C, D\}$ and given that the crossover point is 4, this gives a new member S_3:

$$S_3 = \{\underbrace{C, C, C, C}_{\text{from } S_1}, \underbrace{C, D, C, D, C, D}_{\text{from } S_2}\}.$$

The implementation of crossover in `sequence_sensei` is given by Figure 6.4, and Figure 6.5 demonstrates the usage of the `crossover` function to crossover S_1 and S_2.

```
import random

def crossover(sequence_one, sequence_two):
    sequence_length = len(sequence_one)
    crossover_point = random.randint(0, sequence_length)

    return sequence_one[:crossover_point] + sequence_two[crossover_point:]
```

Figure 6.4: Source code of the `crossover` function.

```
>>> import random
>>> import sequence_sensei as ss

>>> turns = 10
>>> s_one = [1 for _ in range(turns)]
>>> s_two = [i % 2 for i in range(turns)]

>>> random.seed(0)
>>> new_member = ss.crossover(s_one, s_two)
>>> new_member
[1, 1, 1, 1, 1, 1, 0, 1, 0, 1]
```

Figure 6.5: An example of using `crossover` function to crossover S_1 and S_2

Following the crossover between two members, a mutation is applied to new member before it is added to the population. Mutation has been implemented as a given probability p_m that each gene of the new member is flipped. A total of N random numbers between $[0, 1]$ are sampled. If the sampled probability at time i in less than p_m then the i^{th} gene of the individual is flipped, as demonstrated by Figure 6.6.

$$\begin{aligned} S_3 &= \{C, C, C, C, C, C, D, C, D, C\} \\ &\quad\;\; \downarrow 0.134 \;\downarrow 0.847 \;\downarrow 0.764 \;\downarrow 0.255 \;\downarrow 0.495 \;\downarrow 0.449 \;\downarrow 0.652 \;\downarrow 0.789 \;\downarrow 0.094 \;\downarrow 0.028 \\ \text{mutated } S_3 &= \{C, C, C, C, C, C, D, C, D, D\} \end{aligned}$$

Figure 6.6: Mutation example of S_3.

The implementation of mutation in `sequence_sensei` is given by Figure 6.7, and an example of mutating S_3 using the function is given in Figure 6.8.

```
def mutation(gene, mutation_probability):
    if random.random() < mutation_probability:
        return abs(gene - 1)
    return gene
```

Figure 6.7: Source code of the `mutation` function.

```
>>> new_member = [1, 1, 1, 1, 1, 1, 0, 1, 0, 1]

>>> random.seed(1)
>>> [ss.mutation(gene, mutation_probability=0.05) for gene in new_member]
[1, 1, 1, 1, 1, 1, 0, 1, 0, 0]
```

Figure 6.8: An example of using the `mutation` function to mutate S_3.

The main function implemented in `sequence_sensei` for performing a GA is the `evolved` function. The function has several input arguments which correspond to the inputs of Algorithm 6.1. In the following section the `evolved` function is used to run several trials and estimate best response sequences. The parameters values for each run will be presented there. Moreover, the details of the best response sequence collection against the 192 strategies are presented in the following section.

6.4 Data collection

The data set generated in this Chapter was created using the GA of Algorithm 6.1 and the APL project. The GA of Algorithm 6.1 estimates the best response sequence for a given opponent, and in order to generate a collection of best responses a list of opponents is obtained from APL. More specifically 192 strategies are used in this Chapter. These can be found in the Appendix B.2.

The APL project is also used to calculate the average score per turn, $U(S, Q)$, player S can achieve against an opponent Q. The project contains a specific player class that can simulate the play of any given sequence of Cs and Ds. The player class is called `Cycler` and it takes as an input argument a series of actions as a string. An example of creating and using such a player in a match is given by Figure 6.9. The average score per turn is obtained using an in built method of APL once a match has been simulated.

```
>>> import axelrod as axl

>>> players = [axl.Cycler('DDDCCCDDCC'), axl.Cooperator()]
>>> match = axl.Match(players, turns=10)
>>> match.play()
[(D, C), (D, C), (D, C), (C, C), (C, C), (C, C), (D, C), (D, C), (C, C), (C, C)]

>>> match.final_score_per_turn()
(4.0, 1.5)

>>> players = [axl.Cycler('DDDCCCDDCC'), axl.TitForTat()]
>>> match = axl.Match(players, turns=10)
>>> match.play()
[(D, C), (D, D), (D, D), (C, D), (C, C), (C, C), (D, C), (D, D), (C, D), (C, C)]

>>> match.final_score_per_turn()
(2.2, 2.2)
```

Figure 6.9: Simulating a match between **Cycler** and **Cooperator** and **Cycler** and **Tit For Tat**. The class `Cycler` takes a given sequence as an input argument in a string format (`'DDDCCCDDCC'`). Once a match has been simulate with the `play` method the average score per turn is obtained using the `final_score_per_turn` method.

From the 192 strategies, 62 are stochastic and 130 are deterministic. In section 6.2 it was explained that the outcome of a match between two deterministic strategies is always the same. In comparison, the outcome of a match with a stochastic strategy can differ, because the actions of a stochastic opponent are not deterministic. The actions of a stochastic opponent can be repeated by using *computer seeding* for seeding the pseudo random number generator (PRNG) that creates the parameters that define what moves the strategy will take. Seeds are set before generating a random number, and if the same seed is used on initialisation then the random output remains the same. Thus, as long as a match is seeded the behaviour of a stochastic strategy can be reproduced, and different seeds lead to different plays of stochastic strategies.

Consider the match between **Random** and $S = \{D, D, D, C, C, C, D, D, C, C\}$ presented in section 6.2. **Random** has a total of 2^{10} possible plays for the given match. By using different computer seeds a number of these plays can be simulated as shown in Figure 6.10.

```
>>> players = [axl.Cycler('DDDCCCDDCC'), axl.Random()]
>>> for seed in range(5):
...     axl.seed(seed)
...     match = axl.Match(players, turns=10)
...     actions = match.play()
...     print(actions, match.final_score_per_turn())
...     print("================================================================================")
[(D, D), (D, D), (D, C), (C, C), (C, D), (C, C), (D, D), (D, C), (C, C), (C, D)] (2.2, 2.2)
================================================================================
[(D, C), (D, D), (D, D), (C, C), (C, C), (C, C), (D, D), (D, D), (C, C), (C, C)] (2.4, 1.9)
================================================================================
[(D, D), (D, D), (D, C), (C, C), (C, D), (C, D), (D, D), (D, C), (C, D), (C, D)] (1.6, 2.6)
================================================================================
[(D, C), (D, D), (D, C), (C, D), (C, D), (C, C), (D, C), (D, D), (C, C), (C, C)] (2.6, 2.1)
================================================================================
[(D, C), (D, C), (D, C), (C, C), (C, C), (C, C), (D, D), (D, D), (C, D), (C, C)] (2.9, 1.9)
================================================================================
```

Figure 6.10: Example code of using seeding to generate different plays of **Random**. The value of seed changes to {0, 1, 2, 3, 4, 5} and the seed is set with the command `axl.seed(seed)` before simulating the game. This initialises the pseudo random generator that define what moves **Random** will take. The above code snipped will always have the same output each time it is repeated.

A total of 10 different plays are captured for each of the stochastic strategies of this Chapter. Thus, a total of 10 different seeds are used for each stochastic strategy. The data collection process of best response sequences in more details is given by Figure 6.11.

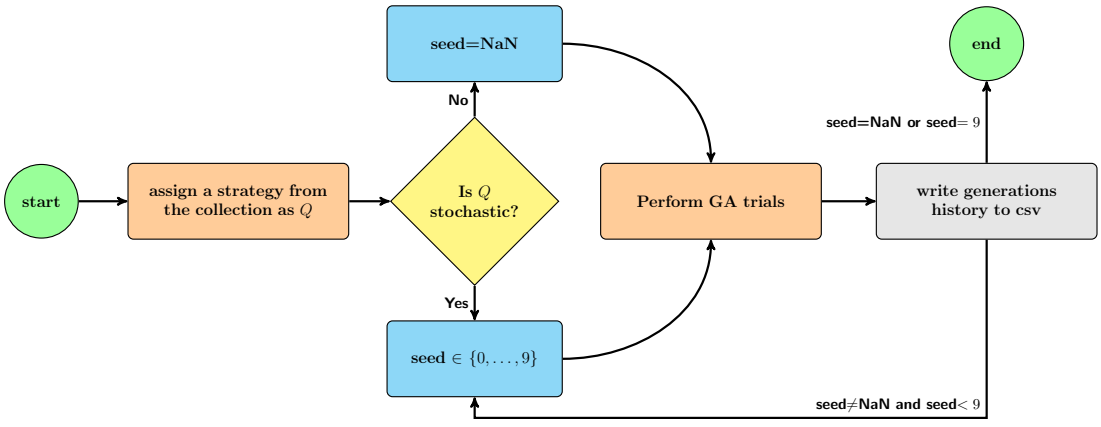


Figure 6.11: Diagrammatic representation of the best response sequences collection process.

From the collection of opponents a strategy is selected at each trial. If the strategy is deterministic a set of GAs with different parameters values are performed for 2,000 generations. The summary of the GA trials which contain information for each generation is exported to a single csv file, then the next opponent is selected. If the opponent

is stochastic the above process is repeated 10 times. Each time the stochastic strategy is accompanied by a different seed value which is used to initialise the pseudo random generator. For each combination of stochastic opponent and seed the GAs summary is exported to a csv file.

For each opponent, or opponent-seed combination a total of 18 GAs are performed. The different values for each parameter are given by Table 6.5.

Parameter	Explanation	Values
N	number of turns	205
G	number of generations	2,000
b	bottleneck	10, 20
K	size of a population	20, 30, 40
p_m	probability of gene mutating	0.01, 0.05, 0.1

Table 6.5: The parameters of the GA. The GA is performed a total of 18 times for each opponent. More specifically, it is performed for each possible combination of the parameters values.

All the best response sequences that are generated in this Chapter are best response sequences of 205 turns. Moreover, they are best responses not to 192 strategies, but to a total of $(130 + 62 \times 10)$ 750 opponents. Thus, a total of 750 trials of Figure 6.11 have been performed.

An example of an exported summary for the deterministic opponent Alternator is given by Table 6.6.

	opponent	seed	b	p_m	$K/2$	g_i	index	score	gene 1	gene 2	...	gene 203	gene 204	gene 205
0	Alternator	NaN	10	0.01	10	0	19	3.009756	0	0	...	0	0	0
1	Alternator	NaN	10	0.01	10	0	0	3.000000	1	0	...	0	0	0
2	Alternator	NaN	10	0.01	10	0	2	2.839024	1	1	...	0	0	0
3	Alternator	NaN	10	0.01	10	0	17	2.839024	0	0	...	1	1	1
4	Alternator	NaN	10	0.01	10	0	4	2.673171	1	1	...	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1080535	Alternator	NaN	20	0.10	20	2000	29	2.775610	1	0	...	0	0	0
1080536	Alternator	NaN	20	0.10	20	2000	24	2.770732	0	0	...	0	0	0
1080537	Alternator	NaN	20	0.10	20	2000	31	2.770732	1	0	...	0	0	0
1080538	Alternator	NaN	20	0.10	20	2000	32	2.770732	0	0	...	0	0	1
1080539	Alternator	NaN	20	0.10	20	2000	34	2.756098	0	0	...	0	0	0

Table 6.6: An exampled of an exported summary. The specific output is for the opponent Alternator. Alternator is a deterministic strategy, consequently, the value of seed in NaN. The values of the different GA parameters are recorded in the summary, as well as the details of each member of each generation. The sequences' genes were recorded in 0 and 1, where $0 \to D$ and $1 \to C$. The best responses sequences are the individuals that have the maximum score at $g_i = 2{,}000$.

For a stochastic strategy there are a total of 9 exported summaries. An example of an exported summary for Champion with seed=9 is given by Table 6.7. The best response sequences are collected from the last generation of the exported summaries.

	opponent	seed	b	p_m	$K/2$	g_i	index	score	gene 1	gene 2	...	gene 203	gene 204	gene 205
0	Champion	9	10	0.01	10	0	10	3.712195	1	1	...	0	0	0
1	Champion	9	10	0.01	10	0	12	3.663415	1	1	...	0	0	0
2	Champion	9	10	0.01	10	0	8	3.585366	1	1	...	0	0	0
3	Champion	9	10	0.01	10	0	14	3.448780	1	1	...	0	0	0
4	Champion	9	10	0.01	10	0	6	3.312195	1	1	...	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1080535	Champion	9	20	0.10	20	2000	25	3.634146	0	0	...	1	1	0
1080536	Champion	9	20	0.10	20	2000	34	3.629268	1	1	...	0	0	1
1080537	Champion	9	20	0.10	20	2000	31	3.604878	1	1	...	1	0	1
1080538	Champion	9	20	0.10	20	2000	21	3.443902	0	0	...	1	0	0
1080539	Champion	9	20	0.10	20	2000	20	3.351220	1	0	...	0	1	0

Table 6.7: An exampled of an exported summary for a stochastic strategy. The column seed does not have a value of NaN anymore but has captured the seed that was used to generate the specific play of the stochastic opponent. The members' genes are also recorded for each generation. Note that $0 \to D$ and $1 \to C$.

In section 6.2 it was stated that the best response to a strategy that does not react to the history is to always defect. Such strategies include Cooperator, Defector, Alternator and the family of the Cycler strategies. Algorithm 6.1 successfully identified their best responses, as shown in Table 6.8.

opponent	score	gene 1	gene 2	gene 3	gene 4	gene 5	gene 6	...	gene 201	gene 202	gene 203	gene 204	gene 205
Cooperator	5.000	0	0	0	0	0	0	...	0	0	0	0	0
Alternator	3.010	0	0	0	0	0	0	...	0	0	0	0	0
Cycler CCCCCD	4.337	0	0	0	0	0	0	...	0	0	0	0	0
Cycler CCCD	4.005	0	0	0	0	0	0	...	0	0	0	0	0
Cycler CCCDCD	3.673	0	0	0	0	0	0	...	0	0	0	0	0
Cycler CCD	3.673	0	0	0	0	0	0	...	0	0	0	0	0
Cycler DC	2.990	0	0	0	0	0	0	...	0	0	0	0	0
Cycler DDC	2.327	0	0	0	0	0	0	...	0	0	0	0	0
Defector	1.000	0	0	0	0	0	0	...	0	0	0	0	0

Table 6.8: Best response sequences to a number of opponents that do not react to the history of the match. The best response to such strategies are to always defect. This was successfully captured by the best sequence collection process of this Chapter.

The best response sequence to Tit For Tat and Grudger is the sequence that cooperates until before the last turn. That way the sequence receives the payoff for mutual cooperation for $N-1$ turns, and then betrays its opponent in the last turn to maximise its score, receiving a payoff of T. It only defects on the last turn because then it can not be punished. The sequence is also the best response to the strategy Hard

Tit For Tat. The strategy is a variant of Tit For Tat that uses that uses a longer history for retaliation. The best response sequences were captured for all three strategies, Table 6.9.

opponent	score	gene 1	gene 2	gene 3	gene 4	gene 5	gene 6	...	gene 201	gene 202	gene 203	gene 204	gene 205
Tit For Tat	3.01	1	1	1	1	1	1	...	1	1	1	1	0
Grudger	3.01	1	1	1	1	1	1	...	1	1	1	1	0
Hard Tit For Tat	3.01	1	1	1	1	1	1	...	1	1	1	1	0

Table 6.9: Best response sequences to strategies Tit For Tat, Grudger and Hard Tit For Tat.

There are more sophisticated yet still established best response sequences. These include the best response to TF1 [163]. In [163] the strategy TF1 was trained using a 16 state finite state machine in a Moran process setting. The TF1 strategy developed a hand shake mechanism that allowed it to identify strategies that play like itself. Once two copies of TF1 identify each other they go into mutually cooperations until the end of their interactions. The best response to the strategy is the sequence that performs the handshake, and goes into mutual cooperations until before the final turn. This best response sequence was also captured by the data collection process. The handshake is performed in the opening three turns, and it is the sequence $\{C, C, D\}$ as shown in Table 6.10.

opponent	score	gene 1	gene 2	gene 3	gene 4	gene 5	gene 6	...	gene 201	gene 202	gene 203	gene 204	gene 205
TF1	3.0	1	1	0	1	1	1	...	1	1	1	1	0

Table 6.10: Best response sequence to TF1 introduced in [163]. The strategy performs a handshake in the first three moves. The hand shake is the sequence CDC. If the opponent plays that same then the strategies go into mutual cooperation.

6.4.1 Parallelisation and stochastic results

The data collection process of this Chapter was carried out using parallel processing. In parallel processing many calculations or executions of tasks are carried out simultaneously. In the case of the data collection here the tasks were scoring members of the population. More specifically, at most 10 members of the population were being scored at the same time.

Parallel programming was executed using *multi threading* [265]. Threads are "light-weight" processes, a unit of execution within a process. Threads are designed to have shared memory and can manipulate global variables of main thread. Scoring a member

of the population corresponded to a single task which was executed on a single thread. For the stochastic opponents the task included seeding/setting the PRNG state before simulating the match. Seeding at the time of generating the data collection was not implemented in a safe thread way.

The data collection was implemented in a way that each thread was setting the global PRNG state. That was then shared across all the the threads without synchronisation. Since the threads are running in parallel, at the same time, and their access to this global PRNG is not synchronised between them, errors occur. In the case of the stochastic opponents this means that there are given instances that are not reproducible and for those instance the simulated behaviour does not reflect the opponent's behaviour for its given seed.

There are several ways that this error can be corrected. There is a thread safe way of implementing seeding which involves giving each thread its own local PRNG. Then there is no longer any state that's shared by multiple threads without synchronisation.

For the stochastic opponents 34% of the collected sequences are not reproducible. Recalling that the aim of this Chapter is to generate a collection of well performing sequences in the IPD, so that they can be used as the training data for the purposes of Chapter 7. A 34% percentage of non reproducible sequences is a reasonable ratio which also ensures more variability to the data for training.

6.4.2 The collection of best response sequences

The collections process was performed for 750 trials, and a total of 18 GAs were performed for each trial. The best response sequences are the sequences with the highest average score per turn in the final generation regardless of the GA.

In order to understand whether the algorithms reached convergence over the 2,000 generations, the highest score in a population over the generations for four different strategies is given in Figure 6.12.

There are trials for which the algorithms never converged to the best response sequences. There are two reason that as to why this happened:

- The bottleneck was equal to the population size. A value $K/2 = 10$ while $b = 20$ results to no new members being added to the population. These trials would

have reached convergence only if the best response was in the initial generation.

- The mutation probability is too high. The earliest converged GA trials are the trials for which $p_m = 0.01$. As p_m is the probability that each gene of the new member is being flipped, higher values could potentially add to much variation to the new members. This could lead to the new members losing critical information they inherited from their parents.

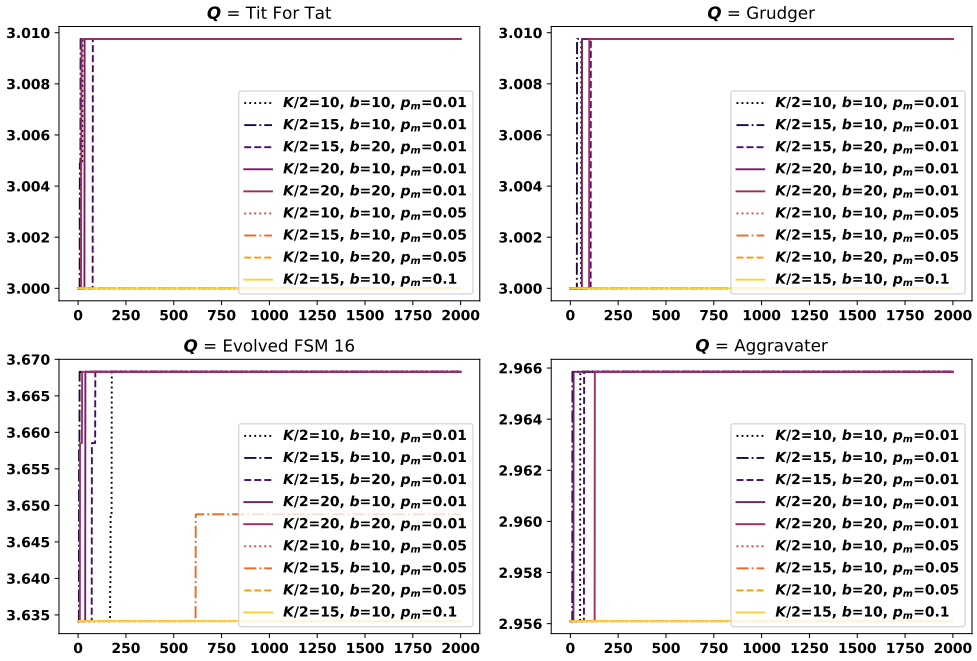


Figure 6.12: The highest score in a population over the generations for **Tit For Tat**, **Grudger**, **FSM 16** and **Aggravater**. The selected trials capture the results of all the 18 trials for the given set of opponents.

Nevertheless, a sequence that has not converged is still useful. Even though it is not the highest scoring member, it is a sequence that was not arbitrarily generated but has some critical information regarding playing against given opponents.

There are several trials that have managed to converged, and they did so in less than 200 generations. This is potentially the effect of a non random initial population. Figure 6.13 shows the highest score in a population over the generations for the four strategies of Figure 6.12 but only up to $g_i = 500$. All trials with a $p_m = 0.01$ converged within the first 200 generations. The trial which reaches convergence first (in the four demonstrated cases) is the trial with the parameter values of $K/2 = 15, b = 10$ and $p_m = 0.1$.

There are a total of 130 deterministic strategies in the collection of opponents and best response sequences were estimated for each. Several of the known best response

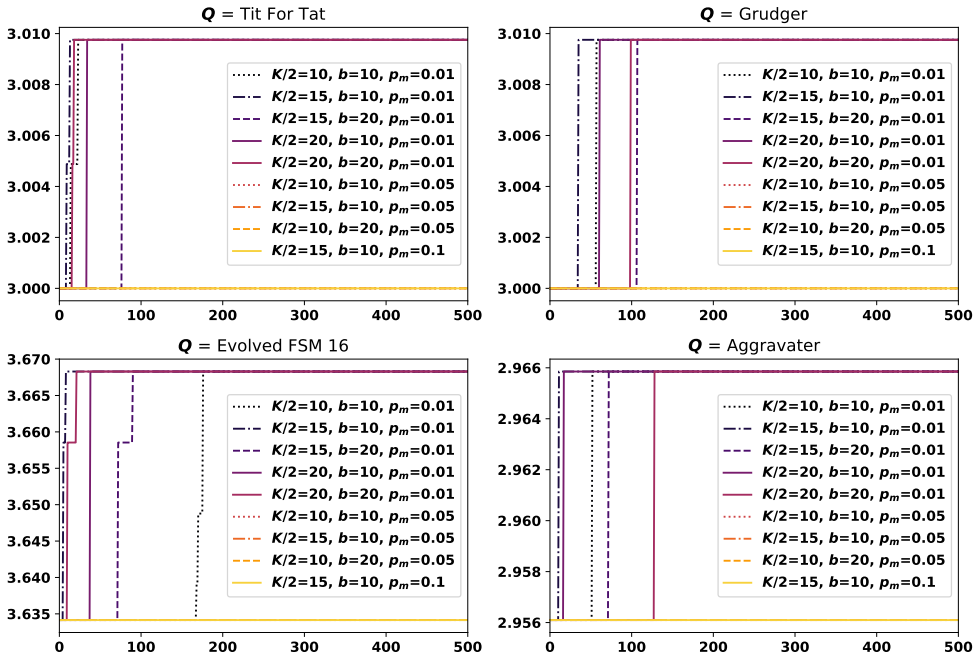


Figure 6.13: The highest score in a population over the generations for **Tit For Tat**, **Grudger**, **FSM 16** and **Aggravater** up to $g_i = 500$.

sequences have been manually checked and they have been successfully estimated by the algorithm. These include best response sequences to **Tit For Tat**, **Grudger**, **Alternator**, **Pavlov** and the **Cycler** strategies.

In section 6.2 is was explained that strategies can have more than a single best response sequence. In the case of the strategy **Adaptive** any strategy that cooperated twice in the opening 6 moves and defected thereafter is a best response. The data collection has managed to successfully identify multiple best responses to **Adaptive**, these are given by Table 6.11. **Adaptive** is not the only deterministic strategy with multiple best response sequences. More specifically, for the 130 deterministic opponents a total of 2,949 best response sequences were collected.

index	gene 1	gene 2	gene 3	gene 4	gene 5	gene 6	gene 7	...	gene 203	gene 204	gene 205
0	1	0	0	0	0	1	0	...	0	0	0
1	0	0	1	1	0	0	0	...	0	0	0
2	1	1	0	0	0	0	0	...	0	0	0

Table 6.11: Best responses sequences estimated by the data collection process. Note that 0 corresponds to defection and 1 to cooperation.

An interesting questions that arises is: how diverse are the set of best response sequences? Out of the 2,949 sequences 2,836 are unique. A graphical representation of

these sequences is given by Figure 6.16a. The two distinct colours represent genes of C and D. Overall, it can be seen that there is diversity in the best response sequences, and they are not just long sequences of either C or D. A common trend appears to be a series of defections at the last turns. In a finite IPD this is to expected. As it was mentioned in Chapter 4, as the likelihood of a match ending in the following turn increases the effectiveness of defecting.

A total of 2,309 sequences have been estimated for the stochastic opponents. From these sequences, 66% did indeed play against the correct seeded opponent and achieved their respective scores at the final generation.

For instance for the strategy - seed combination of **Champion** - seed= 9 the highest score achieved by a member of the population over the generations is given by Figure 6.14. There is variation in the highest score occurring over the generations with several increasing and decreasing peeks. However, for most of the generations the highest score appears to be between $3.80 - 3.82$. The best response sequence retrieved by the data collection scored 3.82 against **Champion**, and it reflected the score of the sequence against **Champion** - seed= 9.

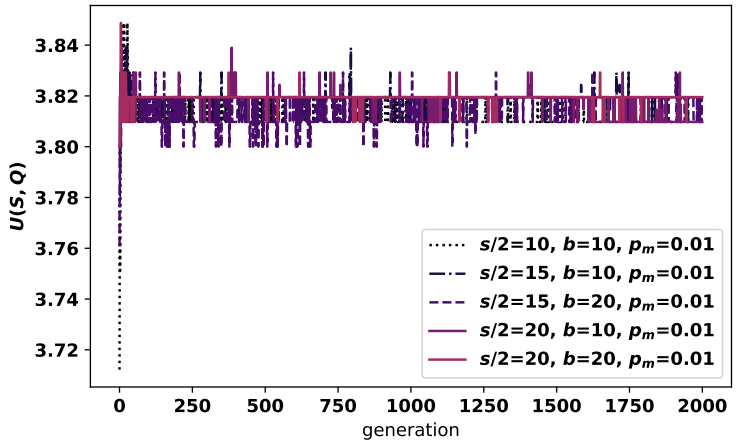


Figure 6.14: The maximum score a sequence achieved against **Champion** with seed 9 over the generations. Each line represents a different GA trial. Only the GAs with $p_m = 0.01$ have been included.

There are stochastic opponents for which more variation occurred over the generations. An example of that is **Random**. The highest score of the population in a single GA trial, for opponent - seed combination **Random** - seed= 1, is given by Figure 6.15. In the case of **Random** - seed= 1 the score of the best response sequence that was collected was not the actual score the sequence scores against **Random** - seed= 1.

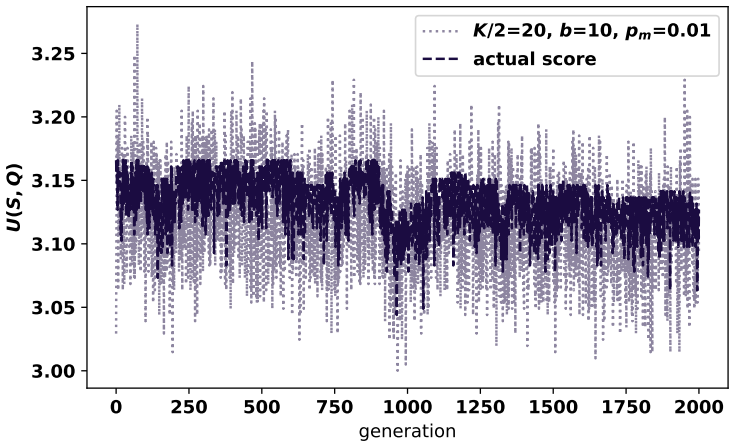


Figure 6.15: The maximum score a sequence achieved against Random - seed= 1 over the generations.

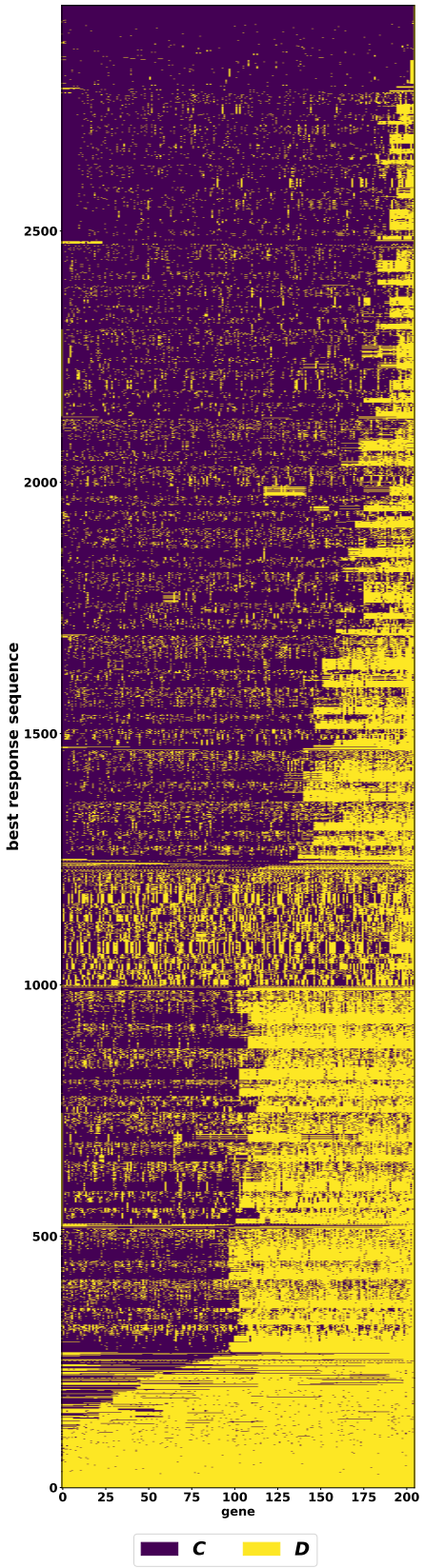
From the 2,309 sequences against stochastic strategies 2,130 are unique. A graphical representation of these sequences are given by Figure 6.16b. Similar to the results of the best response sequences against deterministic opponents the sequences are diverse. However, compared to the best response sequences to deterministic opponents the responses to stochastic opponents appear to be less diverse. The responses to stochastic opponents appear to belong to two families of strategies. Strategies that cooperate for approximately the first 150 turns and defect thereafter, and strategies that play a series of alternating sets of cooperations and defections. The best response sequences to deterministic opponents can not be classified in just two families of strategies, and thus are more diverse.

In summary, from the list of 192 strategies examined in this Chapter, 750 different opponent instances were simulated and a total of 5,258 best response sequences of length 205 were retrieved. The choice of 205 turns will be explained in the following Chapter. The best response sequences have been archived and made available at [103].

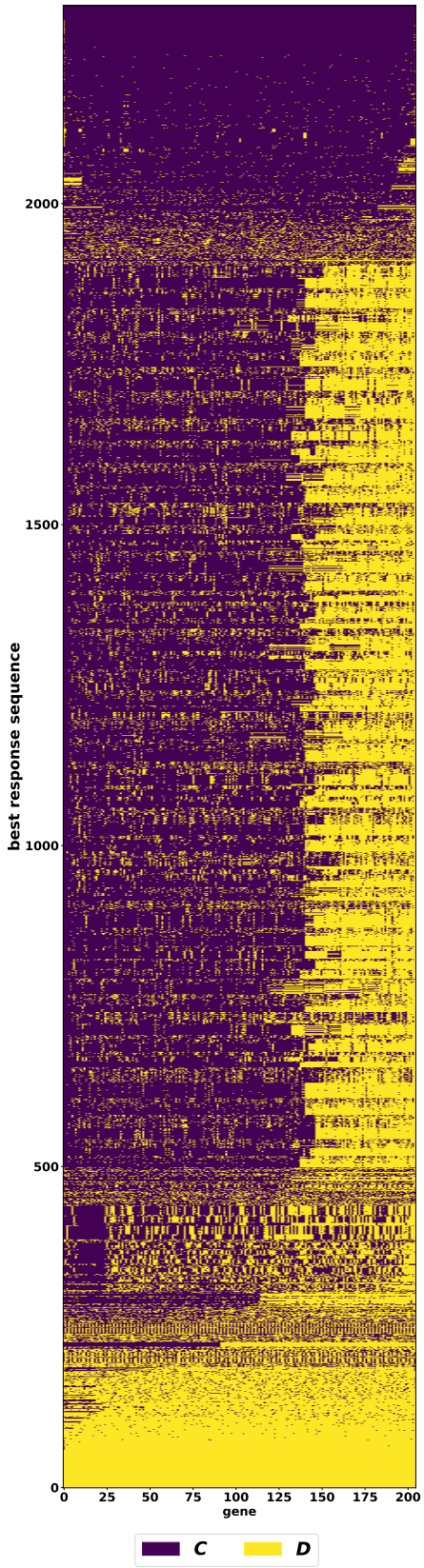
6.5 Chapter summary

This Chapter has explored the concept of best responses in the IPD game in the form of static sequences of moves. It introduced an evolutionary algorithm, Algorithm 6.1, which can successfully identify best response sequences.

The algorithm was executed to estimate best response sequences to the majority of opponents listed in the APL. More specifically, a total of 192 opponents from APL were used. Several of the strategies in the project are stochastic and computer seeded



(a) A graphical representation of best 2,949 response sequence. These have been estimated using deterministic opponents.



(b) A graphical representation of 2,309 response sequence. These have been estimated using stochastic opponents.

Figure 6.16: A graphical representation of best response sequences. The best response sequences have been sorted based on their total number of cooperations. Thus the top rows of the plots are dominated by best response sequences that mainly defect and the bottom rows by sequences that mainly cooperate.

versions of these strategies were used to explore their different behaviours. From the list of 192 opponents a total of 750 different behaviours were simulated.

For the 130 deterministic strategies a total of 2,949 sequences, from which 2,836 were unique, were estimated. These sequences were not just a set of trivial sequences of either C or D. A common trait in the best response sequences appeared to be a series of defection closer to the final turns. For the seeded versions of the 62 stochastic opponents the best response sequences are not guaranteed to have been captured due to issues related to PRNGs and multi threading. Nevertheless, a total of 2,309 sequences from which 2,130 are unique were collected. Similar, these sequences are more diverse than just a series of a single action.

A total of 5,258 sequences were collected. These have been archived and are available at [103]. The main purpose of this Chapter has been to generate the bespoke data set, which contains a large number of unique and diverse sequences, so it can be used as training data in Chapter 7.

Chapter 7

Training long short-term memory networks produces successful Prisoner's Dilemma strategies

The research reported in this Chapter has been carried out with:

Axelrod-Python library (APL) version: 4.2.0

Associated data sets: [105, 106]

Associated codebase: [95]

This work was performed using the computational facilities of the Advanced Research Computing @ Cardiff (ARCCA) Division, Cardiff University.

7.1 Introduction

In Chapter 4 it was mentioned that conceptualising and introducing new strategies has been an important aspect of research to the field. The aim of this Chapter is to introduce new IPD strategies based on an archetype that has not received much attention in the literature.

In Chapter 4 it was concluded that one of the properties successful strategies in a IPD competition need to have is cleverness/complexity. Complexity can confer to adaptability, and adaptability is important in performing well in diverse sets of environments.

This was established not only in Chapter 4 but also from the results of Chapter 5. The set of complex strategies that ranked highly across distinct tournaments in Chapter 4 included strategies based on archetypes such as finite state automata, hidden Markov models and *artificial neural networks* (ANNs).

ANNs have successfully been trained to play games other than the IPD such as checkers [59], chess [83] and Go [267]. *Feed forward networks* were firstly used to represent IPD strategies in 1996 [123]. Feed forward networks have been used in the literature ever since [23, 26, 70, 84], and possibly the most successful ANN strategies in the game are the ones introduced in [122]. The three ANN based strategies in [122] ranked 7^{th}, 9^{th}, and 11^{th} in a tournament of 223 strategies.

A type of ANNs that have not received much attention in the literature are the *recurrent neural networks* (RNNs). RNNs are a type of neural networks that include a feedback connection and are designed to work with inputs in the form of sequences. RNNs were firstly considered as an archetype in 1996. In [257] a RNN which considered a single previous step as an input was trained via a Q learning algorithm to play against a single opponent. The opponent was either the strategy Tit For Tat or another Q learning strategy. The results of [257] although somewhat promising, the RNN player learned to optimally play against Tit For Tat, they were limited.

The limitations of [257] could potentially have been due to the limitations of the RNNs themselves. As it will be discussed later in section 7.2, RNNs quickly became unable to learn due to the vanishing gradient problem. To improve on the standard recurrent networks a new model called the *long short-term memory network* (LSTM) was introduced in [134]. Remembering information for long periods of time is is the default behaviour of LSTMs, not something they struggle to learn.

LSTMs are a set of networks that have been proven to be successfully trained and today are being used in a number of innovative applications such as time series analysis [188], speech recognition [255] and prediction in medical care pathways [290]. However, they have not received attention in the IPD literature. The aim of this Chapter is to train and introduce a number of strategies based on LSTMs. The training has been possible due to the collection of best response sequences generated in Chapter 6.

A total of 24 LSTMs based strategies are introduced in this Chapter, and their performance is evaluated and compared in a meta tournament analysis of 300 standard tournaments. The results demonstrate that LSTM networks can be trained to successfully

compete in IPD tournaments. The rest of the Chapter is structured as follows:

- section 7.2 presents an introduction to artificial, recurrent and long short-term memory neural networks.
- section 7.3 covers the architectural details of the LSTMs considered in this Chapter. The networks are trained to predict best response sequences. This is done not only on the entire data set generated by the collection of best response sequences, but also for three distinct subsets of the collection.
- section 7.4 evaluates and compares the performance of 24 LSTMs strategies in 300 standard IPD computer tournaments.

7.2 Artificial, recurrent and long short-term memory neural networks

ANNs are computing systems vaguely inspired by the biological neural networks that constitute brains. As stated in [147] the research on ANNs has experienced three periods of extensive activity. The first peak was in the 1940s. The work of [197] opened the subject by creating a computational model for neural networks. The second peak occurred in 1960s with the introduction of the perceptron [251]. The perceptron is a linear binary classifier and in the 1960s in was shown that it could be used to learn to classify simple shapes with 20×20 pixels input. However, it was impossible for the perceptron to be extended to classification tasks with many categories. The limitations of the model were presented in [207] which halted the enthusiasm of most researchers in ANNs, resulting in no activity in the field for almost 20 years. The third peak occurred in the early 1980s with the introduction of the back propagation algorithm for multilayered networks [292]. Though the algorithm was initially proposed by [292] it has been reinvented several times and it was popularised by [196]. Following the introduction of the algorithm and the ability to now train more complex models, ANNs have received considerable interest and are being used today in a number of applications, good review articles include [12, 182, 209, 264].

ANNs based on the connection pattern can be grouped into two categories:

- Feed forward networks.
- Recurrent or feedback networks.

ANNs can be viewed as weighted directed graphs in which artificial neurons are nodes and directed edges, with weights, are connections between the neuron outputs and the neuron inputs [147]. This is demonstrated by Figure 7.1. In feed forward networks the graphs have no loops, whereas in recurrent networks loops occur because of feedback connections.

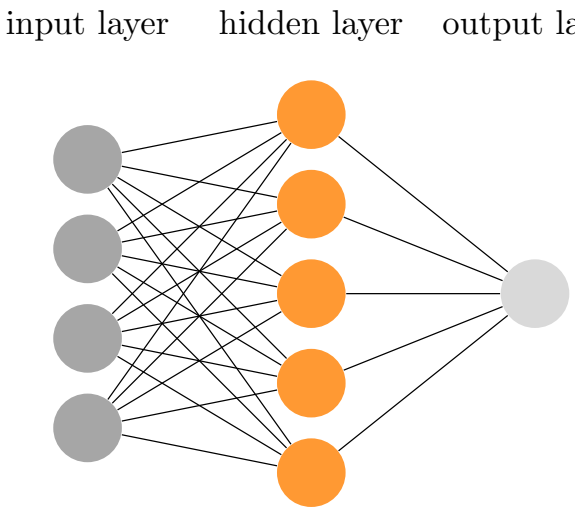


Figure 7.1: A generic representation of an ANN.

A graphical representation of a feed forward network with a single hidden layer is given by Figure 7.2.

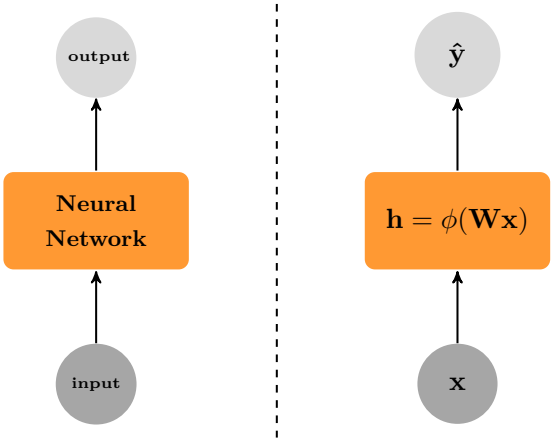


Figure 7.2: Graphical representation of a feed forward network.

A feed forward network is composed by an input layer, hidden layers, and an output layer [147, 171]. The input is a vector x. The dimensionality, or number of nodes of the input layer is dependent on the dimensionality of x. Each element of the input vector is connected to the hidden layer via a set of learned weights. The hidden layer also consists of nodes. The number of nodes of the hidden layer is an architectural decision. The j^{th} hidden node outputs,

$$h_j = \phi(\sum_i w_{ij} x_i), \text{where } \phi \text{ is an activation function.}$$

Frequently used activation functions [157, 171] include:

- The sigmoid function $\sigma(x)$ which squashes numbers into the range $[0, 1]$.
- The hyperbolic tangent, $tanh(x)$ which squashes numbers into the range $[-1, 1]$.
- The rectified linear unit, $ReLU(x) = max(0, x)$.

The hidden layer in turn is fully connected to an output layer, where the j^{th} output node outputs,

$$y_j = \sum_i v_{ij} h_i.$$

Feed forward networks make predictions using forward propagation which in matrix notation is described by,

$$h = \phi(Wx) \tag{7.1}$$

$$\hat{y} = \text{softmax}(Vh) \tag{7.2}$$

where W is a weight matrix connecting the input and hidden layers, V is a weight matrix connecting the hidden and output layers and the output layer transforms the raw scores to a probability via a *softmax function.*

The softmax function, referred to also as softargmax [113] or normalised exponential function [51], is a function that normalises a vector of M real numbers to a probability distribution consisting of M probabilities proportional to the exponentials of the input numbers. The normalisation ensures that the sum of the components of the output vector is 1. The softmax function, denoted as $s(x_i)$, is defined by Equation (7.3).

$$s(x_i) = \frac{e^{x_i}}{\sum_{m=1}^{M} e^{x_m}} \text{ for } i \in 1, \ldots, M. \tag{7.3}$$

W and V are the learning parameters of the network. Their dimensionality depends

on the dimensionality of network's layers. For example if x and $\hat{y}$ were 2 dimensional and the hidden layer had 500 nodes then $W \in \mathbb{R}^{2\times500}$ and $V \in \mathbb{R}^{500\times2}$. Increasing the dimensionality of the hidden layer corresponds to larger number of learning parameters.

To train an ANN a set of values for the learning parameters need to be found so that the error on the training data is minimised [118]. The function that measures error is called the *loss function*. An example of a loss functions is the *cross entropy*, denoted as $L(y, \hat{y})$ where $\hat{y}$ is the prediction and y the true output. For M training examples and K classes the cross entropy error is given by Equation (7.4).

$$L(y, \hat{y}) = -\frac{1}{M} \sum_{m\in M} \sum_{k\in K} y_{m,k} \log(\hat{y}_{m,k}) \tag{7.4}$$

The minimum of the loss function is calculated using the gradient descent algorithm [253]. The gradient descent algorithm relies on the gradients which are the vector of derivatives of the loss function with respect to the learning parameters. Thus $\frac{\partial L}{\partial W}$ and $\frac{\partial L}{\partial V}$. These gradients are calculated with the back propagation algorithm [297] which is a way of efficiently calculating the gradients starting from the output.

According [253] to there are variants of the gradient descent which differ in how much data is used to compute the gradient of the objective function. Using the gradient descent algorithm, the weights are updated incrementally after each pass over the training data set. In contrast, the stochastic gradient descent performs a parameter update for each training example. This work uses a variant of the stochastic gradient descent called the Adaptive Moment Estimation (Adam) algorithm [162].

An extension of feed forward networks are recurrent networks. RNNs are capable of processing variable length sequences of inputs. Sequences are data samples over a number of time steps. RNNs can receive sequence inputs and output sequences of the same length, using their internal state/knowledge to make a prediction on the input at time t based on the decisions made at the previous time steps.

A graphical representation of a RNN is given by Figure 7.3. The loop represents that information can be passed down from one step of the network to the next. RNNs can be thought of as multiple copies of the same network each passing a message to a successor, which is also demonstrated by Figure 7.3.

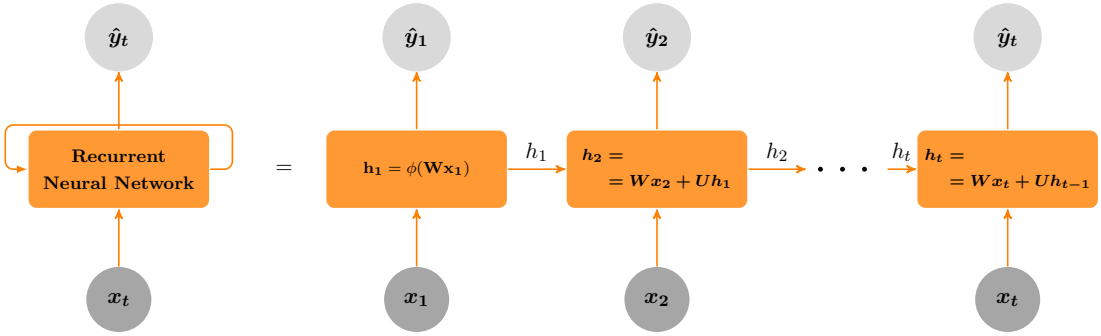


Figure 7.3: Graphical representation of a RNN.

In RNNs information is passed down by using,

$$h_t = \phi(Wx_t + Uh_{t-1})$$

where h_{t-1} is the hidden state computed at time $t-1$ multiplied by some weight vector U. In matrix notation RNNs are described by,

$$h_t = \phi(Wx_t + Uh_{t-1}) \tag{7.5}$$

$$\hat{y}_t = \text{softmax}(Vh_t). \tag{7.6}$$

Unfortunately in practice RNNs quickly become unable to learn to connect the information. The longer the sequences, the higher the chance that the back propagation gradients either accumulate and explode or vanish down to zero. The fundamental difficulties of RNNs were explored in depth by [47]. As stated in [134], although RNNs are theoretically fascinating there was no clear practical advantage over feed forward networks.

A network specifically designed to avoid the long-term dependency problem was introduced by [134], called the long short-term memory network. The core idea to LSTMs is the *cell state* also referred to as the long term memory, denoted as C_t. The cell state is a vector designed to pass down information with only a few carefully regulated changes being applied to it by structures called *gates*. Gates are composed of a sigmoid neural net layer and a point wise multiplication operator. In order to explain the cell gate and subsequently how LSTMs make predictions consider a LSTM's hidden layer at time step t given by Figure 7.4.

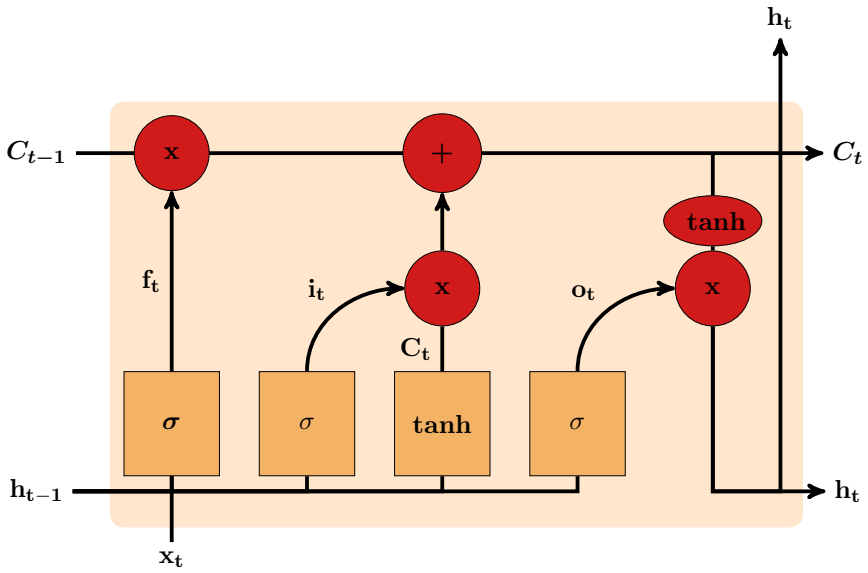


Figure 7.4: An LSTM hidden layer at time step t.

The cell state and hidden state from the previous time step are fed back into the network. Initially, the network decides what information from the previous cell state is to be discarded. This decision is made by the *forget state*. The forget state considers the hidden state at time $t-1$ and the input at time t,

$$f_t = \sigma(W_f x_t + U_f h_{t-1}). \tag{7.7}$$

The forget gate outputs a number between 0 and 1 or each number in the cell state C_{t-1}. A 1 represents "keep this" while a 0 represents "forget this".

Secondly, the network decides what information is going to be stored at the cell gate. There are two parts to this. Firstly, the input gate decides which values are going to be updated,

$$i_t = \sigma(W_i x_t + U_i h_{t-1}). \tag{7.8}$$

and secondly, a tanh layer creates a vector of candidate values denoted as $\tilde{C}_t$ that could be added to the cell state. $\tilde{C}_t$ is given by Equation (7.9).

$$\tilde{C}_t = tanh(W_c x_t + U_c h_{t-1}). \tag{7.9}$$

The cell state C_{t-1} is multiplied by f_t, forgetting the values which have been decided

to be discarded. The new candidate values are scaled by how much information has been decided to keep from the input. Thus,

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t. \tag{7.10}$$

The LSTM outputs a hidden state at each time step which is based on the current cell state. Initially, the cell state goes through a tanh function and then it is multiplied by a sigmoid gate that decides which parts to output from the cell state,

$$o_t = \sigma(W_o x_t + U_o h_{t-1}), \tag{7.11}$$

$$h_t = o_t tanh(C_t). \tag{7.12}$$

This process is being carried out for each time step of the input sequence. At each time step both the cell state and hidden state are feed back into the network. The hidden state can also be used to make a prediction at each time step as demonstrated by Figure 7.5.

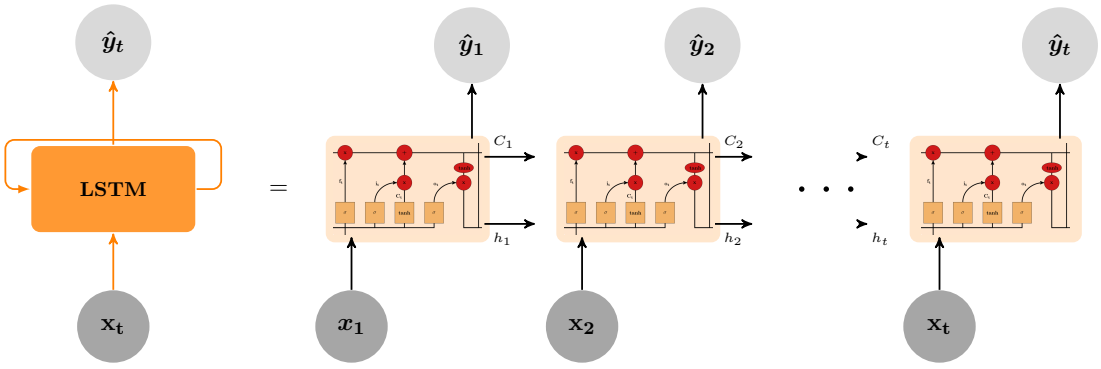


Figure 7.5: Graphical representation of an LSTM network.

LSTMs unique architecture allows them to learn longer-term dependencies and to be trained using the back propagation algorithm. Hence, they are an exceptional model for sequential data and this is why they were chosen here. As it was described in Chapter 6, the actions of IPD strategies can be defined as sequences, and therefore a collection of best response sequences was generated using a heuristic method. In section 7.3 a variety of LSTM networks are trained to predict these best response sequences.

7.3 Training LSTM networks to predict best response sequences

LSTM are trained in a supervised fashion on a set of training sequences. The purpose of training a network in this Chapter is so it can learn to play optimally against IPD strategies. For that reason the networks are going to be trained on the collection of best response sequences generated in Chapter 6.

The training inputs are the actions of a given strategy for N turns. The expected outputs are the responses to those N actions by the opponent's best response sequence. Each best response sequence, from Chapter's 6 collection, corresponds to 204 expected outputs. This is done so that each sample captures a match between a strategy and its best response for N turns in a match where $N \in \{1, 204\}$.

Consider the actions of the strategy Adaptive and its best response, as presented in section 6.4.2, given by Table 7.1.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	...	**204**	**205**
Adaptive	1	1	1	1	1	1	1	0	0	0	0	1	...	1	1
S^*	1	1	0	0	0	0	0	0	0	0	0	0	...	0	0

Table 7.1: The actions of the strategy Adaptive against one of the best response sequences to the strategy. Note that $0 \to D$ and $1 \to C$.

Initially the highest dimensionality a training sample based on Table 7.1 can have is 204. This is because the expected output to Adaptive's action in turn 1 is the action of the best response sequence at turn 2. Moreover, the expected output to the Adaptive's 204^{th} action is the best response's 205^{th} action. This is demonstrated by Figure 7.6.

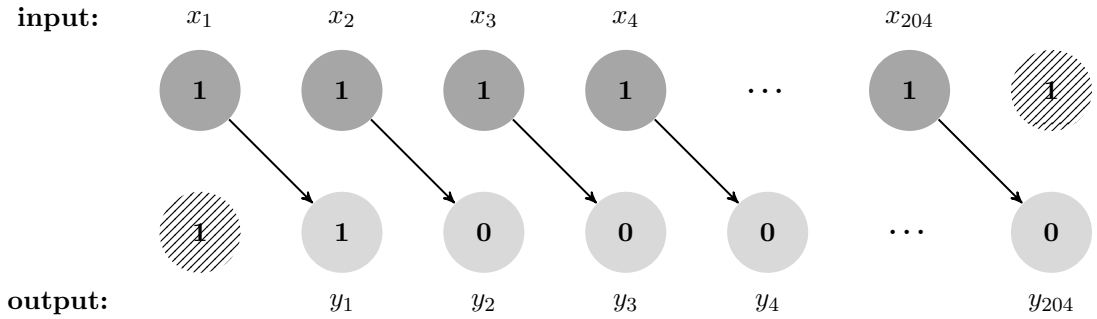


Figure 7.6: An example of a networks input and output of $t = 204$. The last action of **Adaptive** as well as the first action of the best response sequence are discarded.

Secondly, in order to train the networks on different input lengths the training sample of Figure 7.6 is transformed to 204 samples. This is done by considering all the possible IPD matches between the pair where the number of turns $N \in [1, 204]$. For example Table 7.1 corresponds to the training samples given by Equation (7.13).

$$
\begin{aligned}
x &= \begin{bmatrix} 1 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 1 \end{bmatrix} \\
x &= \begin{bmatrix} 1 & 1 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 1 & 1 \end{bmatrix} \\
x &= \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \\
x &= \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \\
x &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \end{bmatrix} \\
x &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\
x &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
&\vdots && && \vdots
\end{aligned}
\tag{7.13}
$$

Subsequently, the training data set retrieved from the collection of best responses has a total of $5,258 \times 204 = 1,122,612$ training samples.

Two types of LSTMs have been trained in this Chapter. These are referred to as:

- The sequence to sequence (StoS) network.

- The sequence to probability (StoP) network.

Both networks take as input a sequence of actions. The StoS network outputs a response to each time step of the input sequence, thus attempting to recover the entire best response sequence. The StoP network only outputs a response to the sequence at the last time step, thus only attempting to figure out what comes next. Both networks indicate the predicted action with a probability of cooperating. A graphical representation of the two networks are given by Figures 7.7 and 7.8.

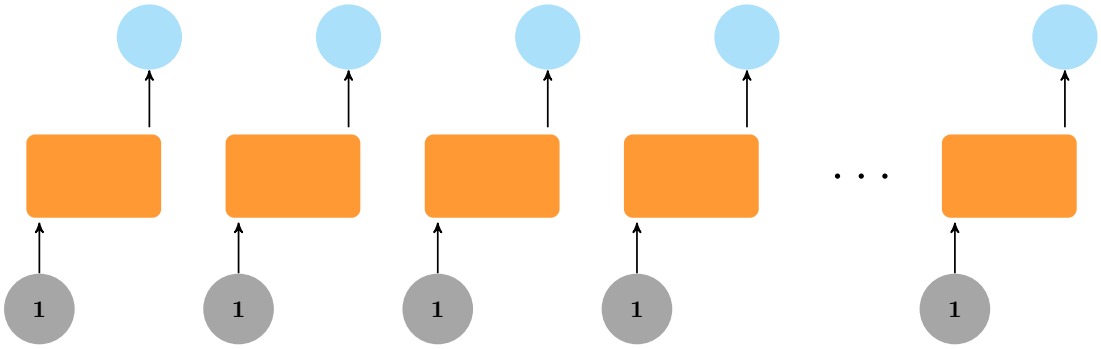


Figure 7.7: A graphical representation of the StoS LSTM network.

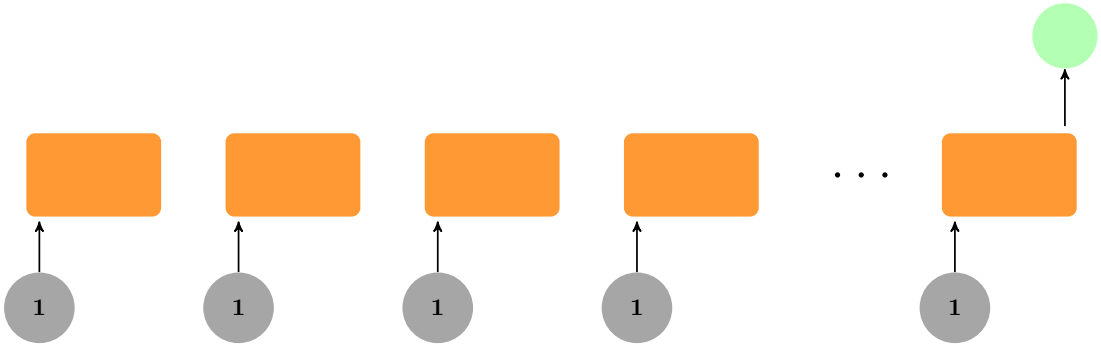


Figure 7.8: A graphical representation of the StoP LSTM network.

The StoS network is trained on samples in the form of Equation (7.13), whereas the StoP network is trained on sample in the form of Equation (7.14).

$$
\begin{aligned}
x &= \begin{bmatrix} 1 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 1 \end{bmatrix} \\
x &= \begin{bmatrix} 1 & 1 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 1 \end{bmatrix} \\
x &= \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 0 \end{bmatrix} \\
x &= \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 0 \end{bmatrix} \\
x &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 0 \end{bmatrix} \\
x &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 0 \end{bmatrix} \\
x &= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} && \rightarrow && y = \begin{bmatrix} 0 \end{bmatrix} \\
&\vdots && && &\vdots
\end{aligned}
\tag{7.14}
$$

7.3.1 Building the networks with Keras

There are many open source libraries that allow the creation of neural nets in Python without having to explicitly write the code from scratch. As stated in [287] the three most popular are TensorFlow [9], Keras [63], and PyTorch [234]. Keras is a high level neural net Python library that runs on top of TensorFlow. Though Keras' performance is comparatively slower than TensorFlow and PyTorch, Keras has a simple architecture and it is more readable and concise. Keras has been used in several academic works such as [42, 201, 272], and is also used here to construct and train the networks.

The Python code for implementing the StoS model is given by Figure 7.9. In line 12 the model is defined to be of the `Sequential` class. This mean that the model will be constructed layer by layer. The StoS network has a single LSTM layer with 100 nodes. The input to the LSTM network is not of a fixed length and there a single time step between the elements of each input sequence. This is defined by the argument `input_shape=(None, 1)`. The LSTM layer outputs a hidden state at each time step. Initially, the hidden states go through a dropout layer. The dropout layer is a simple and yet efficient method to reduce overfitting by randomly dropping out nodes of the hidden states [39]. Finally, the hidden states are transformed into probabilities via a

sigmoid layer. There are a total of 41,301 learning parameters to the StoS model.

```
>>> from keras.models import Sequential

>>> from keras.layers import (
...     Dense,
...     Dropout,
...     CuDNNLSTM,
... )

>>> num_hidden_cells = 100
>>> drop_out_rate = 0.2

>>> model = Sequential()

>>> model.add(
...     CuDNNLSTM(
...         num_hidden_cells, return_sequences=True, input_shape=(None, 1))
... )

>>> model.add(Dropout(rate=drop_out_rate))

>>> model.add(Dense(1, activation="sigmoid"))
>>> model.summary()
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cu_dnnlstm_1 (CuDNNLSTM)     (None, None, 100)         41200
_________________________________________________________________
dropout_1 (Dropout)          (None, None, 100)         0
_________________________________________________________________
dense_1 (Dense)              (None, None, 1)           101
=================================================================
Total params: 41,301
Trainable params: 41,301
Non-trainable params: 0
_________________________________________________________________
```

Figure 7.9: Python code for implementing the StoS LSTM with Keras.

Regarding the dimensionality of the hidden layer there is no direct answer as to what is the optimal number of nodes. Several methods for determining the dimensionality include experimentation, intuition and building on the work of others. A common practice is that the dimensionality of the hidden layer is smaller than the dimensionality of the input layer. The dimensionality of the input layer changes from 1 to 204, and thus a number of 100 nodes was chosen so that the dimensionality of the hidden layer is smaller 50% of the times.

The Python code for implementing the StoP network with Keras is given by Figure 7.10. The implementations of the networks are similar, however, the StoP model contains 2

LSTM layers. The first LSTM layer outputs the hidden states at each time step. In turn these are connected to the second layer which only outputs the hidden state at the final time step. The StoP network has a higher number of learning parameters due to the two LSTM layers. More specifically, there are 122,101 learning parameters to the network.

```
>>> from keras.models import Sequential
>>> from keras.layers import (
...     Dense,
...     Dropout,
...     CuDNNLSTM,
... )

>>> num_hidden_cells = 100
>>> drop_out_rate = 0.2

>>> model = Sequential()

>>> model.add(
...     CuDNNLSTM(num_hidden_cells, return_sequences=True, input_shape=(None, 1))
... )

>>> model.add(CuDNNLSTM(num_hidden_cells))
>>> model.add(Dropout(rate=drop_out_rate))

>>> model.add((Dense(1, activation="sigmoid")))
>>> model.summary()
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
cu_dnnlstm_2 (CuDNNLSTM)     (None, None, 100)         41200
_________________________________________________________________
cu_dnnlstm_3 (CuDNNLSTM)     (None, 100)               80800
_________________________________________________________________
dropout_2 (Dropout)          (None, 100)               0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 101
=================================================================
Total params: 122,101
Trainable params: 122,101
Non-trainable params: 0
_________________________________________________________________
```

Figure 7.10: Python code for implementing the StoP LSTM with Keras.

Keras includes two implementations for an LSTM layer. The class `LSTM` and the class `CuDNNLSTM` which is the class used here. `CuDNNLSTM` provides a faster implementation of `LSTM` with the NVIDIA CUDA Deep Neural Network library. The use of the `CuDNNLSTM` class means that the networks can be trained on a graphics processing unit (GPU).

7.3.2 High performance training

Conventionally the execution of computer code happens on the central processing unit (CPU). The CPU, also called main processor, is essentially the brain of any computing device [49]. Architecturally the CPU is composed of just a few cores designed to support an extremely broad variety of tasks.

A graphical processing unit (GPU), on the other hand, is composed of hundred of cores designed to process a set of simpler and more identical computations in parallel. GPUs were initially designed as dedicated graphical rendering workhorses for computer games. However as stated in [57], graphics processors transitioned from their initial role to general purpose engines for high throughput floating-point computation.

A CPU core is more powerful than a GPU core. A CPU core is designed to carry out a variety of tasks one of which include computations, whereas GPUs are designed exclusively for data computations. The vast majority of a CPUs power goes unused by machine learning applications. Machine learning applications which perform large numbers of computations on a vast amount of data can see huge performance improvements when running on a GPU versus a CPU.

There are several manufacturers of GPUs. These include NVIDIA, AMD, Asus and Intel. NVIDIA created a parallel computing architecture and platform for its GPUs called CUDA [124]. CUDA programming model gave developers access and the ability to express simple processing operations in parallel through code. The vast majority of deep learning projects work exclusively with NVIDIA GPUs because of the better software support NVIDIA provides. In 2005, [273] presented the usage of GPUs in training a generic 2 layer fully connected neural network. The first important work was later in 2008 [241]. However, the usage of GPUs in machine learning was popularised in 2012 by [170].

Due to the time advantage, the training process of the networks was carried out on a GPU. The training was performed using the computational facilities of the ARCCA division, Cardiff University.

7.3.3 Training data sets

Section 7.3 covered the training data set used to train the LSTM networks. There are a total of 1,122,612 training inputs and expected outputs in the data set. In order to understand the effect of the training samples on the LSTM strategies' performance the

networks are trained on the entire data set and on three unique subsets.

The subsets are based on three collection of opponents. These are a collection of best performing strategies, 15 representative strategies with. equally distributed ranks across a standard tournament and a collection of basic strategies. The details of the subsets are given by Table 7.2. A list of the strategies' names used in each subset is given in the Appendix B.3.

Data set	# of opponents	Explanation	# of best response sequences
all strategies	192	The data set as presented in section 7.3.	5,258
top performing strategies	18	A data set constructed in the same way as the training data set but only with the best response sequences to 18 strategies. These are the top performing strategies in a standard tournament of 218 opponents.	714
representative strategies	15	A data set generated only with the best response sequences to 15 strategies whose ranks are across the 218 ranks of the standard tournament. They are referred to as representative strategies because they represent behaviours of all types of performance (successful and not).	212
basic strategies	11	A data set generated with the best response sequences to 11 strategies which are classified as a set of basic strategies in the APL.	84

Table 7.2: Training data sets used to train the LSTM networks. The IPD standard tournament with the 218 opponent has been carried out using APL version 3.10.0. The results are available at [104].

The training data sets have been archive and are available at [106].

7.3.4 Training and validation

The two different LSTM networks, StoS and StoP, are trained on four different training sets. Thus, a total of $4 \times 2 = 8$ LSTM networks have been trained in this Chapter.

The networks were trained using the back propagation algorithm and the Adam algorithm presented in section 7.2. Due to the different size of the training data the networks have been trained for a different number of epochs. The number of epochs is the number of times the training algorithm has work through an entire training data set. The number of epochs for each of the 8 network is given by Table 7.3.

	all strategies	top strategies	representative strategies	basic strategies
sequence to sequence	19999	83999	39999	129999
sequence to probability	7999	74999	39999	84999

Table 7.3: Number of epochs for each of the LSTM networks.

The StoP networks have a larger number of learning parameters. Subsequently, their

training corresponds to more computer time. It can been seen from Table 7.3 that the StoP networks have been trained for less epochs compared to the StoS networks.

At each training process the training data set is split into 80% training samples and 20% test samples. At each epoch the loss function is calculated for both the training and test samples. The loss function used to trained the networks is the *binary cross entropy*. In essence, the task of predicting IPD actions is a binary classification problem as there can only be two classes: cooperation or defection.

As discussed in section 7.2, the loss function is used to optimise the learning algorithm. The networks that will be used as IPD strategies in the next section are the networks that achieved the lowest loss value over the epochs. The weights of the best performing networks have been archive and are available at [105].

Another measure which is being reported at each epoch is the accuracy. The accuracy is calculated after the learning parameters have been determined and it is in the form of a percentage. Accuracy is a measure of how accurate the predictions are based on the expected output. Both the measures over the number of epochs are given by Figures 7.11 and 7.12.

In Figure 7.11 the loss and accuracy are given for the StoS networks. Overall the StoS networks, regarding the training data set, have maintained a high value of accuracy over the epochs. Whilst training on the basic strategies there was a minor decrease, however, the accuracy still remained over 0.85 (85%). The loss function values have also remained low over the epochs, for all the networks, with only a few spikes occurring.

For the StoP networks there appears to be more variation in the test loss and accuracy, Figure 7.12. This is more evident for the networks that were trained on the top strategies and the representative strategies. This could indicate that the networks are overfitting. The StoP network trained on the entire training data set has managed to maintained a low loss value (at 0.5) and a training accuracy of 0.75. The most successfully trained StoP network appears to be the network trained on the basic strategies.

This section has presented the 8 trained LSTM networks of this thesis. These networks are based on two different architectures and have been trained on four different training data sets. The networks' validation based on the training data sets was presented in this section. In the next section the networks are evaluated as IPD strategies.

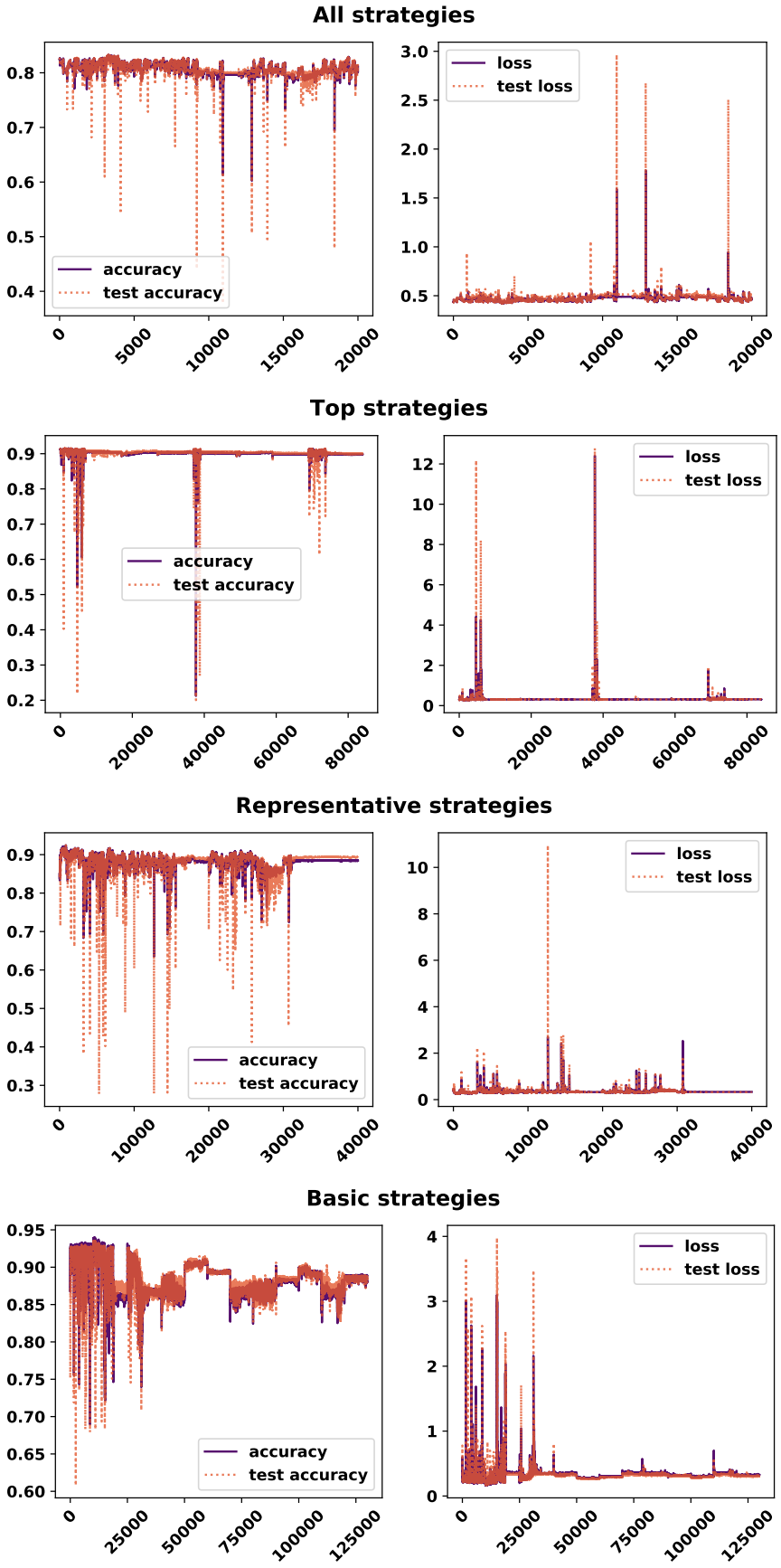


Figure 7.11: Loss function and accuracy of the networks based on the StoS network, over the number of epochs.

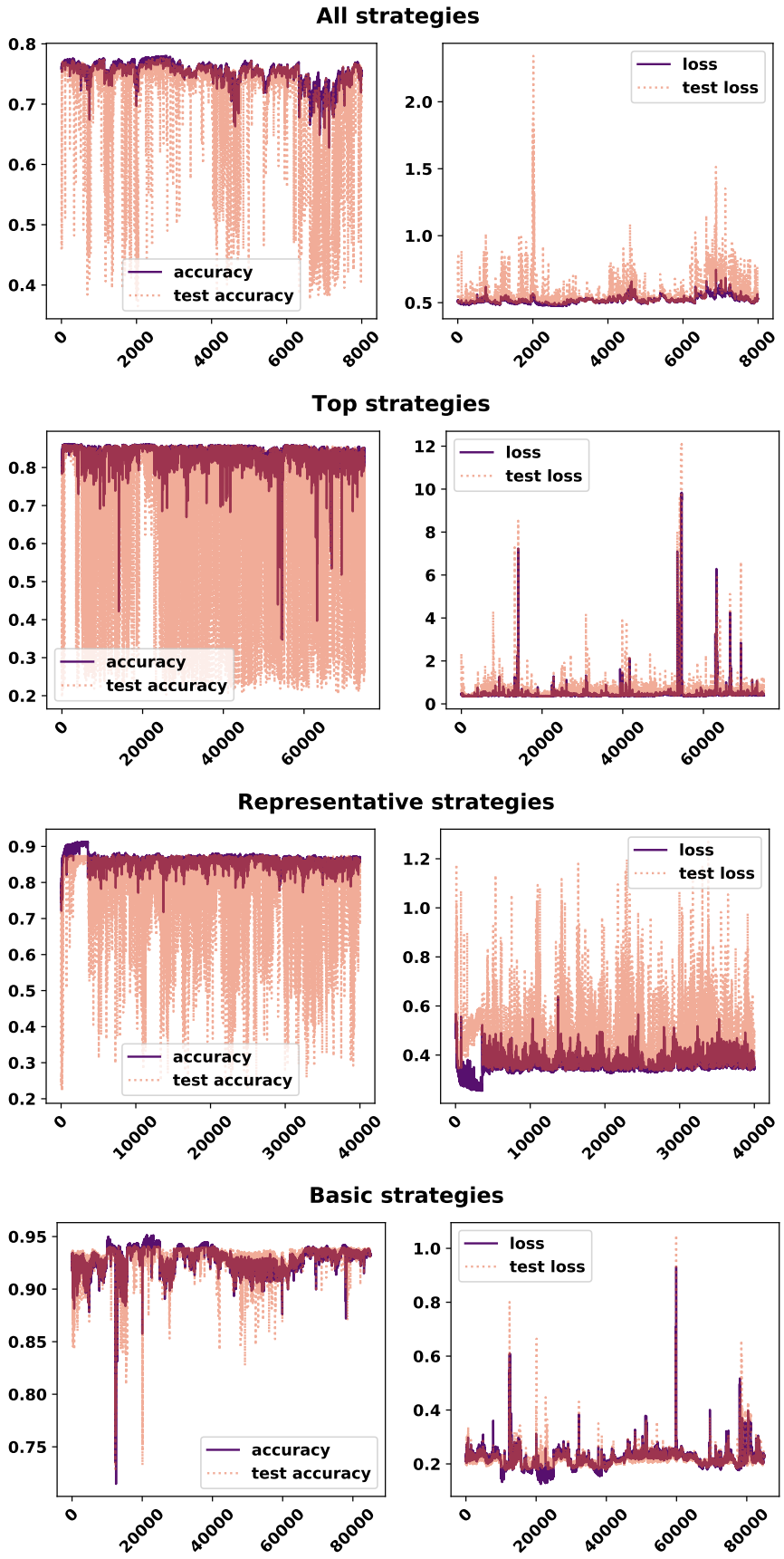


Figure 7.12: Loss function and accuracy of the networks based on the StoP network, over the number of epochs.

7.4 Validation of LSTM based strategies using a meta tournament analysis

This section evaluates the trained LSTM networks as IPD strategies. A strategy class called the `LSTMPlayer` was implemented in order for the networks, which are Keras models, to interact in an IPD match simulated with APL. The source code for the `LSTMPlayer` is given by Figure 7.13.

```
1   import numpy as np
2
3   import axelrod as axl
4   from axelrod.random_ import random_choice
5   from keras.layers import LSTM, Dense, Dropout
6   from keras.models import Sequential
7
8   C, D = axl.Action.C, axl.Action.D
9
10  class LSTMPlayer(axl.Player):
11      name = "The LSTM player"
12      classifier = {
13          "memory_depth": float("inf"),
14          "stochastic": True,
15          "inspects_source": False,
16          "manipulates_source": False,
17          "manipulates_state": False,
18      }
19
20      def __init__(self, model, reshape_history_funct, opening_probability=0.78):
21          self.model = model
22          self.opening_probability = opening_probability
23          self.reshape_history_function = reshape_history_funct
24          super().__init__()
25          if opening_probability in [0, 1]:
26              self.classifier["stochastic"] = False
27
28      def strategy(self, opponent):
29          if len(self.history) == 0:
30              return random_choice(self.opening_probability)
31
32          history = [action.value for action in opponent.history]
33          prediction = float(
34              self.model.predict(self.reshape_history_function(history))[0][-1]
35          )
36
37          return axl.Action(round(prediction))
38
39      def __repr__(self):
40          return self.name
```

Figure 7.13: Implementation of the `LSTMPlayer` class.

The class has three input arguments:

- A Keras model. The input models are the 8 trained LSTM networks presented in section 7.3.
- A reshape history function. A function that reshapes the opponent's history to the appropriate LSTM input.
- The probability that the strategy opens with cooperation on the first turn denoted as p_o. The LSTM networks can be used to predict the strategy's next action following the opponent's opening turn. Thus, the strategy's opening action must be manually defined.

Following the opening turn the LSTM strategy makes a prediction based on the opponent's history. The strategy has an infinite memory because it needs to remember all the actions made by the opponent. The prediction of the networks correspond to the probability of cooperating. The LSTM strategy makes a deterministic decision based on the predicted probability. It cooperates if the prediction on the last time step is greater than 0.5, otherwise it defects. In section 7.4.2 the performance of a strategy that plays stochastically will be briefly discussed.

The 8 trained LSTM networks are used to introduce 24 new IPD strategies. Each network corresponds to three distinct players with a different opening move. More specifically, three different values of p_o are used here. These are $p_o = 0$, $p_o = 1$ and $p_o = 0.78$. The probability 0.78 is the probability that the best response sequences of Chapter 6 opens with a cooperation. Thus, a total of $8 \times 3 = 24$ IPD strategies are evaluated in this section.

The performance of the LSTM strategies are evaluated and compared in 300 standard tournaments similarly to Chapter 4. The process of collecting the tournaments results for each strategy is given by Algorithm 7.1.

For each trial a random size $s \in [5, 10]$ is selected, and from the 192 strategies of Appendix B.2, a random list of s strategies is chosen. The LSTM player is then added to the list of players, increasing the size to $s + 1$. For the given list of strategies a standard tournament of 200 turns is performed and repeated 50 times. The number of turns is fixed at 200. In Chapter 6 the sequences were fixed to 205 turns which resulted in many best response sequences to defect on the last turn as the match was coming to an end. To avoid a series of unconditional defections by the LSTM strategies their performance is evaluated in 200 turns, which is a common number of turns used in the

Algorithm 7.1: Standard tournament result summary collection algorithm

foreach *seed* $\in [0, 300]$ **do**

$s \leftarrow$ randomly select integer $\in [s_{min}, s_{max}]$;

players $\leftarrow$ randomly select s players;

players $\leftarrow$ players + LSTM strategy;

$s \leftarrow s + 1$;

$k \leftarrow 50$;

$n \leftarrow 200$;

result standard $\leftarrow$ Axelrod.tournament(players, n, k);

return *result standard*;

IPD literature [32, 33, 122, 164].

A total of 300 trials of Algorithm 7.1 have been run. For each trial a result summary (in the format of Table 4.2) is exported. Similarly to Chapter 4, the performance of the strategies is evaluated on the normalised rank r, and more specifically on the median normalised rank $\bar{r}$. As a reminder r is calculated as a strategy's rank divided by $s - 1$.

The $\bar{r}$ of each of the 24 strategies over the 300 standard tournaments is given by Table 7.4.

	sequence to sequence			**sequence to probability**		
	$p_o = 0$	$p_o = 1$	$p_o = 0.78$	$p_o = 0$	$p_o = 1$	$p_o = 0.78$
All strategies	0.667	0.222	0.333	0.778	0.333	0.500
Top strategies	0.714	0.444	0.500	0.500	0.429	0.429
Representative strategies	0.750	0.667	0.683	0.500	0.250	0.300
Basic strategies	0.800	0.600	0.625	0.800	0.300	0.429

Table 7.4: The median normalised ranks of the 24 LSTM strategies over the standard tournaments. A $\bar{r}$ closer to 0 indicates a more successful performance.

The strategy with the lowest $\bar{r}$ over the 300 tournaments is the LSTM strategy based on the StoS network trained over the entire training set with $p_o = 1$. The strategy achieved a $\bar{r}$ of 0.222. The second most successful performance is by the StoP based strategy trained against the representative strategies with $p_o = 1$. In section 7.3 it was indicated that the specific network was overfitting, however, as an IPD strategy the network outperforms any other StoP strategies.

A few strategies have achieved a $\bar{r}$ close to 0.3. These include the StoS strategy trained on the entire data set with $p_o = 0.78$, and the StoP strategies trained against all strategies, the representative strategies, and against the basic strategies with $p_o = 1$, $p_o = 0.78$ and $p_o = 1$ equivalently.

The LSTM strategies that open with cooperation outperform any other strategy based on the same LSTM networks. Overall, the best performing strategies open with cooperation. The strategies that open with a probabilistic cooperation perform better than the strategies that open with defection. Interestedly, from Table 7.4 it is indicated that the strategies trained on the subsets perform better when they are based on the StoP model. There is no intuition as to why that is. The StoP networks have more learning parameters and yet they perform better when trained on the smaller training sets than the StoS network. The StoP network, trained against the entire data set, has trained for the smallest number of epochs. A topic of future work would be to train the specific network for more epochs.

Figure 7.14 gives the r distributions for the strategies based on the StoS network. All the distributions for $p_o = 0$ have a median higher than 0.66, indicating that those strategies on average perform in the bottom half of a tournament. The most successful strategy is the strategy trained against all strategies with $p_o = 1$. The strategy's distribution shows that the strategy ranked highly in most of the tournament it participated with only a few exceptions. Even if when p_o is lowered to 0.78 the strategy still performs adequately. The rest of the distributions appear to have peaks either at 0.5 or closer to 1. A statistical summary of the distributions are given by Table 7.5.

		count	mean	std	min	10%	25%	50%	75%	95%	max	skew	kurt
All strategies	$p_o = 0$	300.0	0.620	0.278	0.0	0.200	0.429	0.667	0.839	1.000	1.0	-0.523	-0.587
	$p_o = 1$	300.0	0.295	0.252	0.0	0.000	0.111	0.222	0.458	0.779	1.0	0.702	-0.251
	$p_o = 0.78$	300.0	0.368	0.265	0.0	0.000	0.143	0.333	0.560	0.833	1.0	0.433	-0.647
Top strategies	$p_o = 0$	300.0	0.655	0.273	0.0	0.250	0.500	0.714	0.875	1.000	1.0	-0.629	-0.436
	$p_o = 1$	300.0	0.461	0.274	0.0	0.090	0.222	0.444	0.667	0.875	1.0	-0.029	-0.940
	$p_o = 0.78$	300.0	0.509	0.255	0.0	0.167	0.333	0.500	0.704	0.876	1.0	-0.158	-0.710
Representative strategies	$p_o = 0$	300.0	0.643	0.306	0.0	0.141	0.486	0.750	0.875	1.000	1.0	-0.768	-0.523
	$p_o = 1$	300.0	0.636	0.262	0.0	0.250	0.500	0.667	0.833	1.000	1.0	-0.680	-0.198
	$p_o = 0.78$	300.0	0.645	0.255	0.0	0.250	0.500	0.683	0.833	1.000	1.0	-0.754	-0.034
Basic strategies	$p_o = 0$	300.0	0.728	0.263	0.0	0.333	0.600	0.800	1.000	1.000	1.0	-0.949	0.229
	$p_o = 1$	300.0	0.556	0.283	0.0	0.143	0.375	0.600	0.778	1.000	1.0	-0.365	-0.803
	$p_o = 0.78$	300.0	0.579	0.280	0.0	0.167	0.375	0.625	0.800	1.000	1.0	-0.435	-0.772

Table 7.5: Statistics summary of the r distributions for the strategies based on the StoS network.

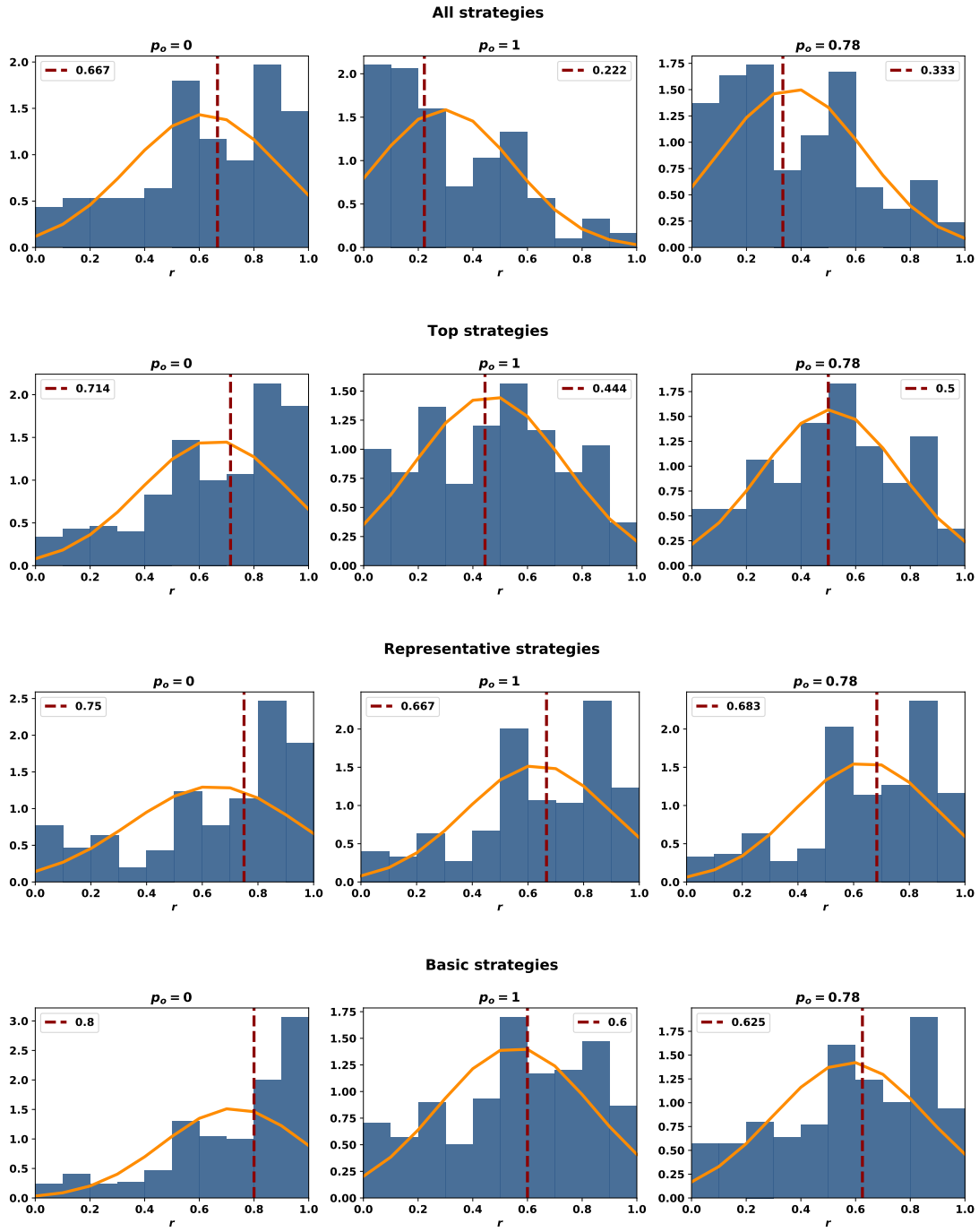


Figure 7.14: Normalised rank distributions for the strategies which are based on the StoS LSTM.

Figure 7.15 gives the r distributions for the strategies based on the StoP network. For $p_o = 0$ the performance of the strategies remains poorly. The strategies that have been trained against all strategies, representative and basic strategies with $p_o = 1$ appear to have won several of the tournaments they participated in. The statistics summary of the distributions is given by Table 7.6.

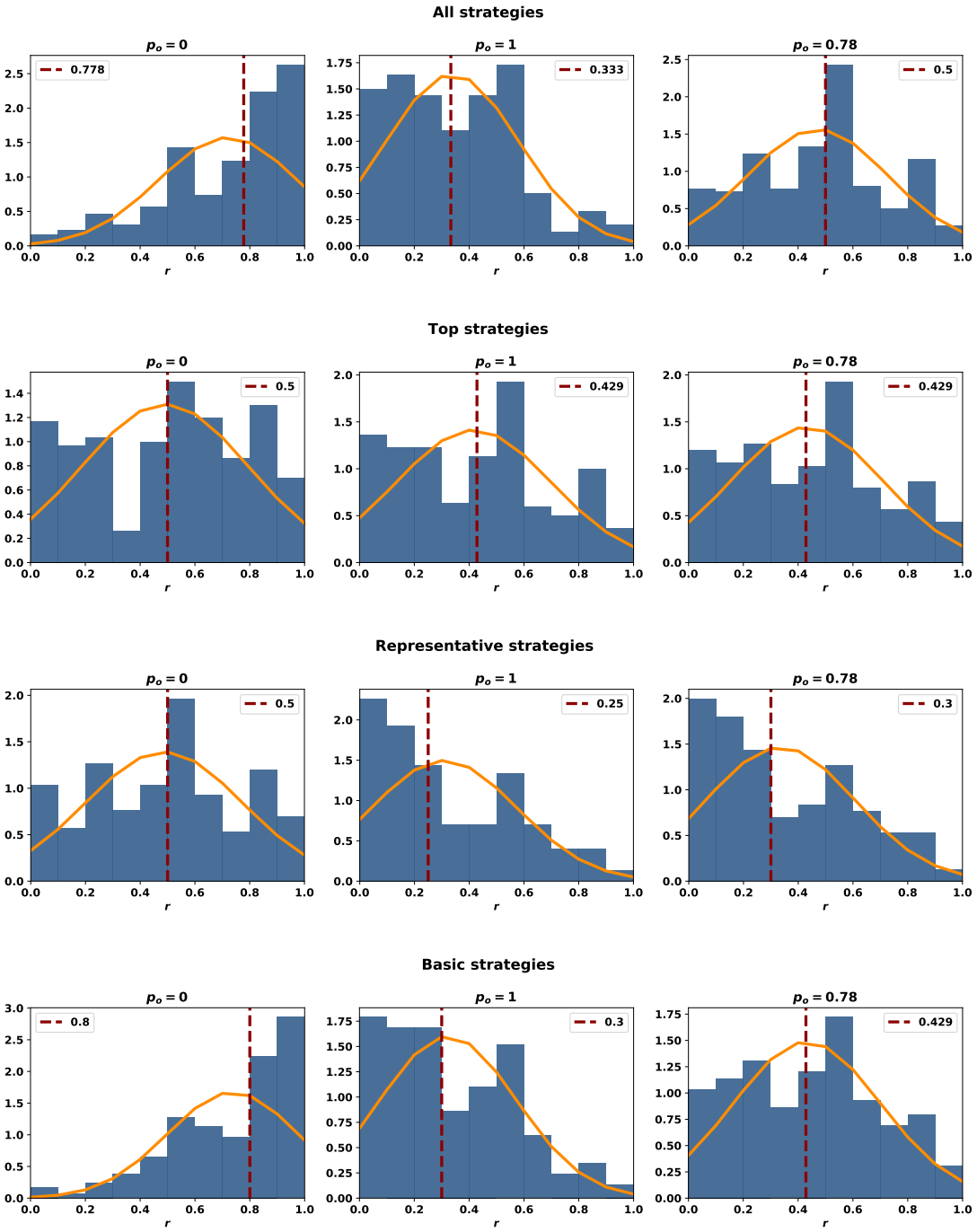


Figure 7.15: Normalised rank distributions for the strategies which are based on the StoP LSTM.

		count	mean	std	min	10%	25%	50%	75%	95%	max	skew	kurt
All strategies	$p_o = 0$	300.0	0.720	0.254	0.0	0.333	0.571	0.778	0.900	1.000	1.0	-0.860	0.056
	$p_o = 1$	300.0	0.339	0.244	0.0	0.000	0.125	0.333	0.500	0.800	1.0	0.458	-0.280
	$p_o = 0.78$	300.0	0.471	0.255	0.0	0.125	0.286	0.500	0.625	0.875	1.0	-0.064	-0.696
Top strategies	$p_o = 0$	300.0	0.491	0.305	0.0	0.000	0.200	0.500	0.714	1.000	1.0	-0.131	-1.140
	$p_o = 1$	300.0	0.417	0.283	0.0	0.000	0.167	0.429	0.600	0.875	1.0	0.157	-0.974
	$p_o = 0.78$	300.0	0.432	0.277	0.0	0.000	0.200	0.429	0.625	0.876	1.0	0.109	-0.891
Representative strategies	$p_o = 0$	300.0	0.487	0.287	0.0	0.000	0.286	0.500	0.700	1.000	1.0	-0.037	-0.899
	$p_o = 1$	300.0	0.308	0.267	0.0	0.000	0.111	0.250	0.500	0.800	1.0	0.586	-0.695
	$p_o = 0.78$	300.0	0.335	0.272	0.0	0.000	0.125	0.300	0.556	0.800	1.0	0.465	-0.867
Basic strategies	$p_o = 0$	290.0	0.738	0.239	0.0	0.400	0.600	0.800	0.975	1.000	1.0	-0.881	0.315
	$p_o = 1$	290.0	0.323	0.249	0.0	0.000	0.125	0.300	0.500	0.778	1.0	0.491	-0.535
	$p_o = 0.78$	290.0	0.432	0.269	0.0	0.000	0.200	0.429	0.625	0.875	1.0	0.096	-0.916

Table 7.6: Statistics summary of the r distributions for the strategies based on the StoP network.

An interesting question that arises is: what is the probability that a LSTM strategy ranks in the top half of a standard tournament?

This is answered by calculating the cumulative distribution function (CFD) of r. CFD is the cumulative probability for a given value. It can be used to determine the probability that a random observation taken from the population will be less than or equal to a certain value. The CFD distributions are shown by Figures 7.16-7.17. The demonstrate that the strategies with a $\bar{r}$ less than 0.333 have a 0.70-0.80 probability of being on the top ranks on a standard IPD tournament.

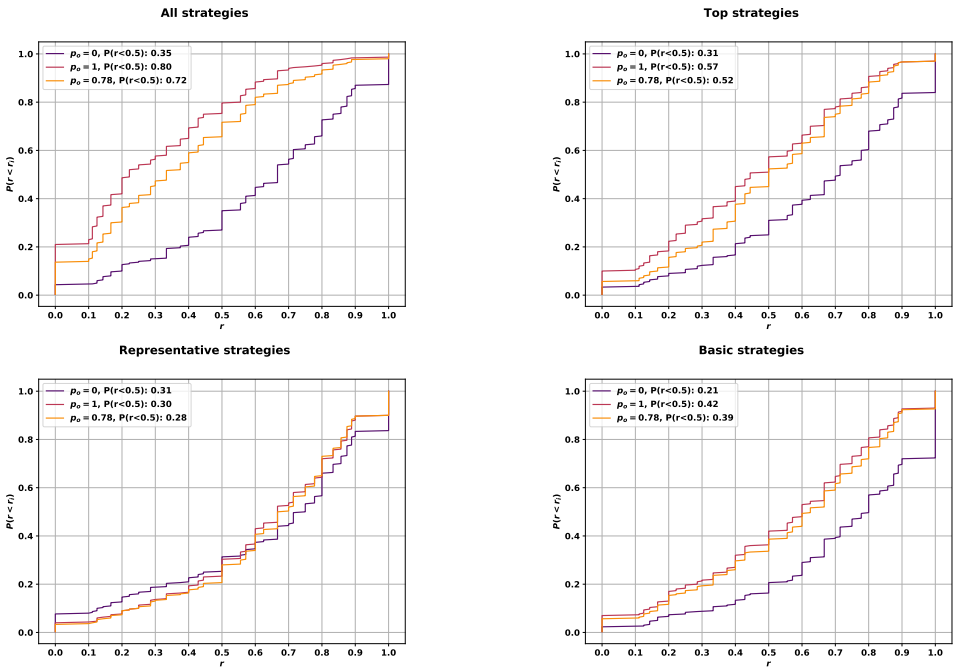


Figure 7.16: The cumulative distribution function (CFD) for the r distributions for the LSTM strategies based on the StoS network.

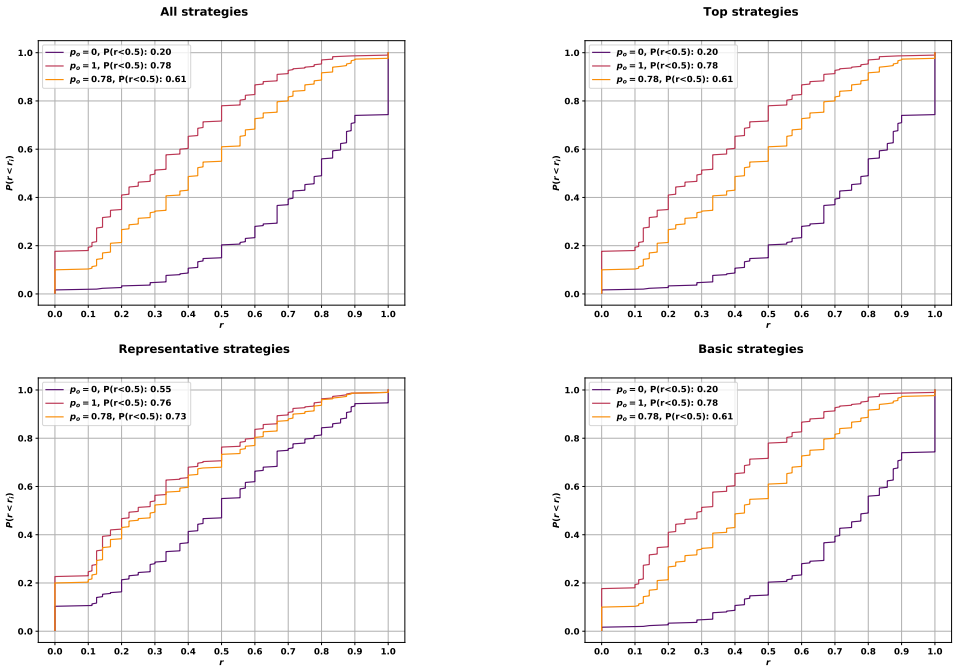


Figure 7.17: The cumulative distribution function (CFD) for the r distributions for the LSTM strategies based on the StoP network.

This section has evaluated the performance of 24 newly introduced IPD strategies based on LSTM networks. The performance of the strategies was evaluated based on their normalised ranks in 300 standard computer tournaments.

A total of 6 strategies have achieved a $\bar{r}$ lower than 0.35. Thus, these strategies have on average ranked on the top 30% of a standard tournament. Furthermore, these strategies have a 0.70-0.80 probability of ranking in top half of a standard tournament. Overall the most successful strategies of the analysis have been strategies that open with a cooperation.

The LSTM strategies that have been trained against the top ranked strategies performed poorly. The top ranked strategies consisted of many trained strategies from [122]. In [122] it was shown that these strategies managed to exploit weak opponents whilst achieving mutual cooperation with strong opponents. The best response sequences of these strategies could have potentially only captured a single behaviour of these strategies, thus not providing enough diverse training samples. This could have in turn made the LSTM strategies less adaptable to diverse environments.

On the whole, the analysis of this section has shown that the LSTM strategies which were trained on the entire data set of best responses were successful strategies regardless of the LSTM architecture. Both the StoS and StoP networks have produced strategies

that can win IPD tournaments, and on average rank on the top 30% of any given standard tournament. Moreover, the LSTM strategies trained on subsets of the training data set, perform better when trained with the StoP network than the StoS network. The StoP networks for the subsets have been trained for a longer number of epochs compared to the StoP network over the entire data set. An interesting question is whether the StoP strategy, trained on the entire data set, would perform even better than its equivalent StoS strategy if it was trained for longer.

Having successfully trained high performing strategies using LSTM networks, the next section will attempt to qualify their behaviour.

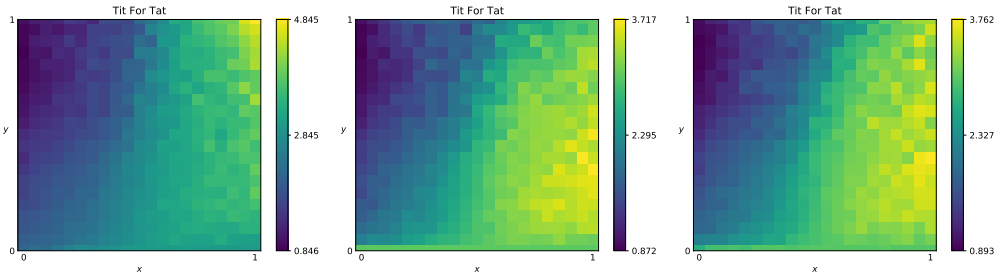
7.4.1 Fingerprinting the LSTM based strategies

The 24 strategies that have been introduced in this Chapter are based on an LSTM archetype. These strategies are based on a complex structure and interpreting their behaviour is not trivial. The difference between the strategies is not straightforward either. In Chapter 2 a method that produces a functional signature of a strategy called fingerprinting was presented. More specifically, two types of fingerprints were discussed which were the Ashlock fingerprints and the transitive fingerprints.

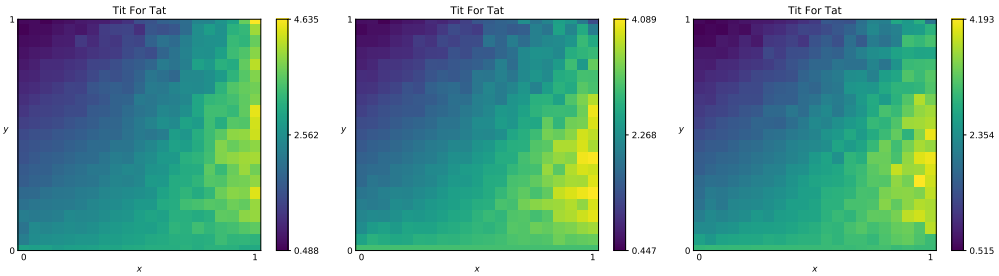
Ashlock's fingerprints [22, 23, 24, 25, 26] compute the score of a strategy against a spectrum of opponents. The basic method is to play the strategy against a probe strategy with varying noise parameters. The fingerprints for the 24 strategies based on Ashlock's approach have been generated for the probe strategies Tit For Tat and Pavlov. These are given by Figures 7.18 - 7.21. Note that the strategies that appear on the same row are of the same network type and have been trained on the same training data set.

Tit For Tat was used as the probe strategy for Figures 7.18 and 7.19. From the fingerprints it is demonstrated that the strategies of the same row behave in a similar manner even though their opening moves differ. The strategies based on different networks and the strategies trained on different training data sets behave differently. The only set of strategies based on different networks that exhibit similarities, is the strategies trained against the top performing strategies.

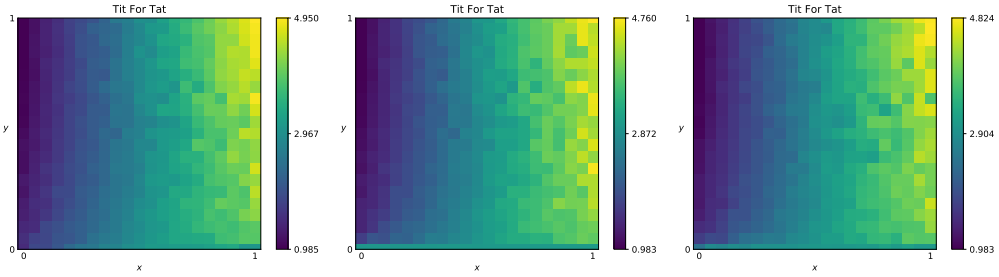
Figures 7.20 and 7.21 give the Ashlock fingerprints whilst Pavlov is used as a probe. These fingerprints, across the network types, training data set and opening moves appear to be more similar.



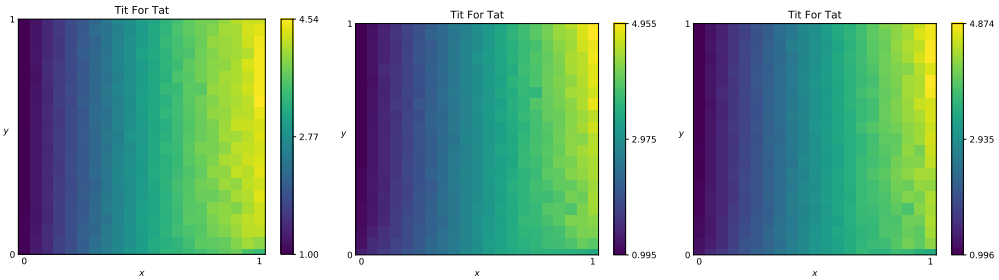
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.

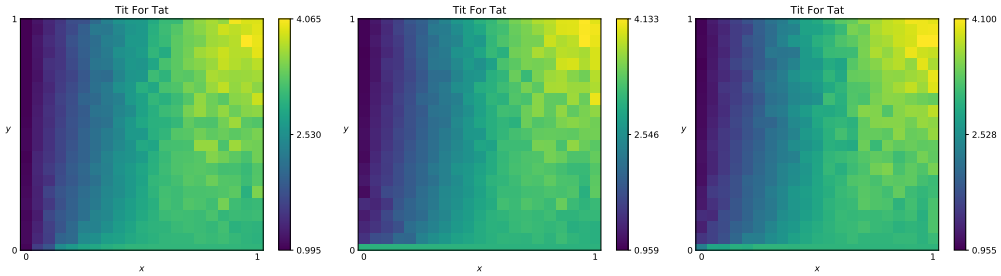


(c) Strategies trained against the representative strategies.

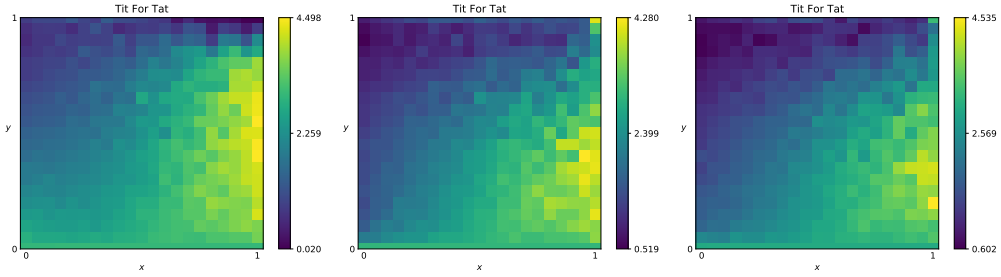


(d) Strategies trained against basic strategies.

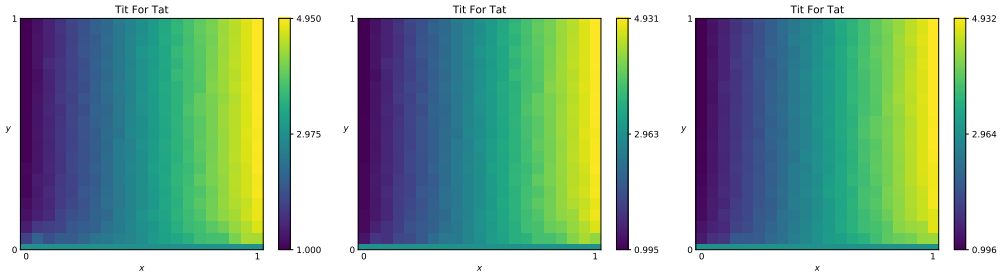
Figure 7.18: Ashlock's fingerprints for the LSTM strategies based on the StoS network when **Tit For Tat** is the probe strategy.



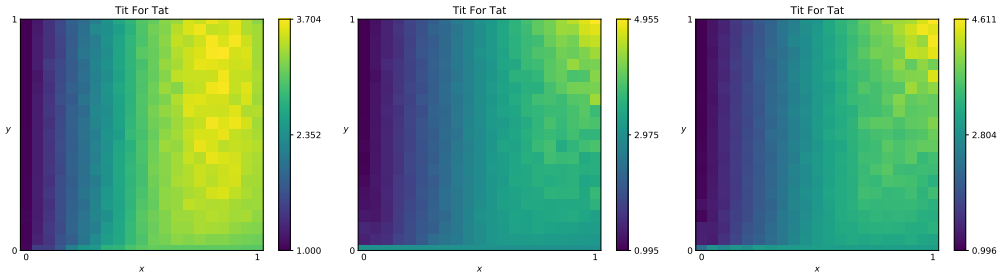
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.

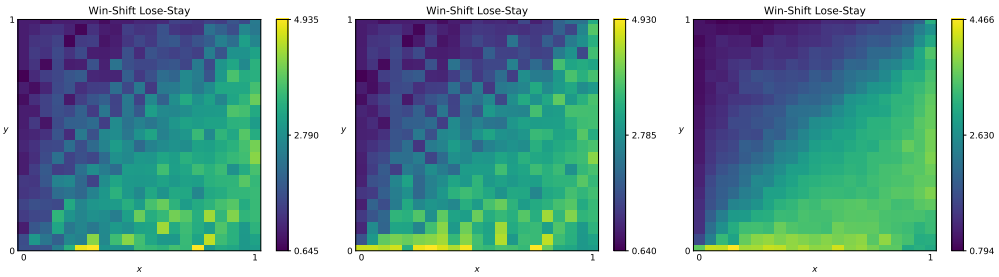


(c) Strategies trained against the representative strategies.

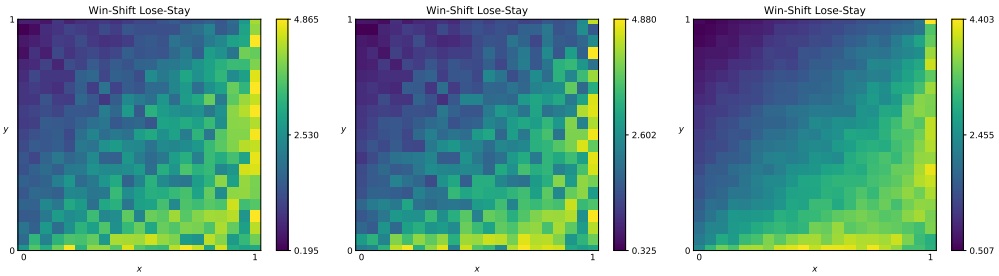


(d) Strategies trained against basic strategies.

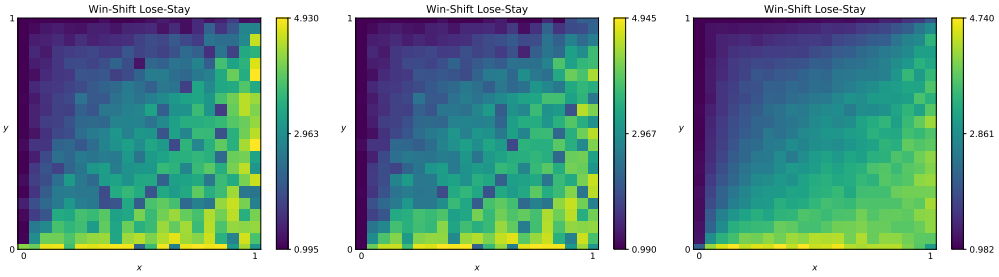
Figure 7.19: Ashlock's fingerprints for the LSTM strategies based on the StoP network when **Tit For Tat** is the probe strategy.



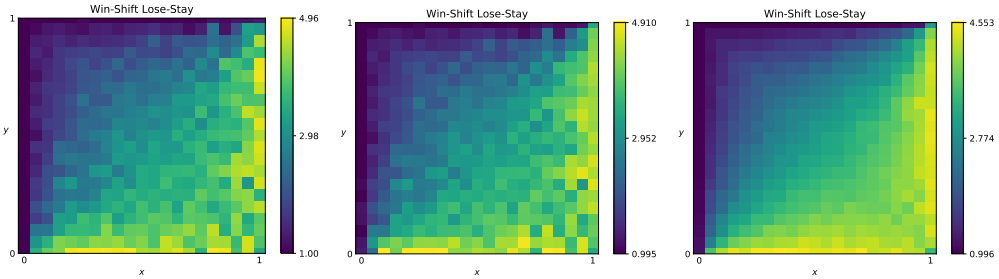
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.

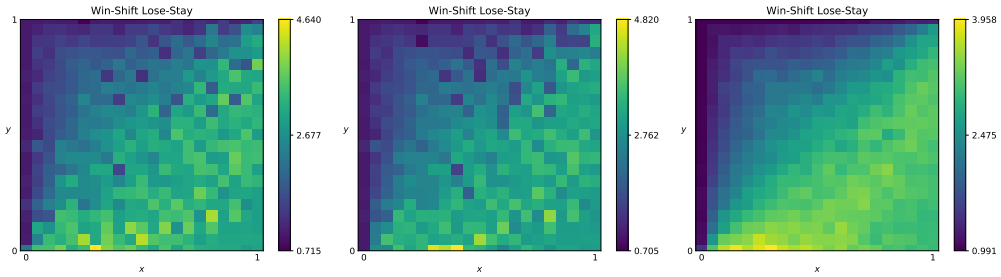


(c) Strategies trained against the representative strategies.

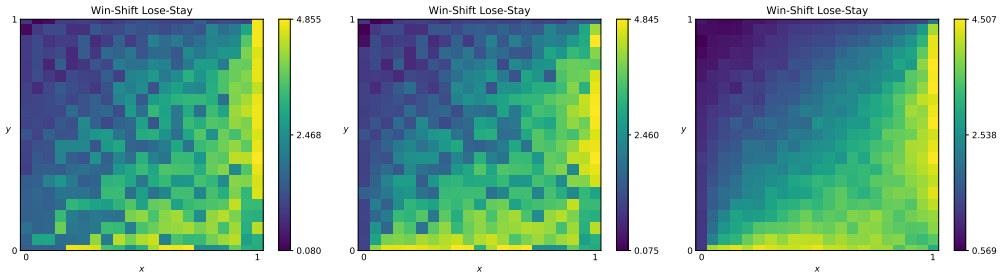


(d) Strategies trained against basic strategies.

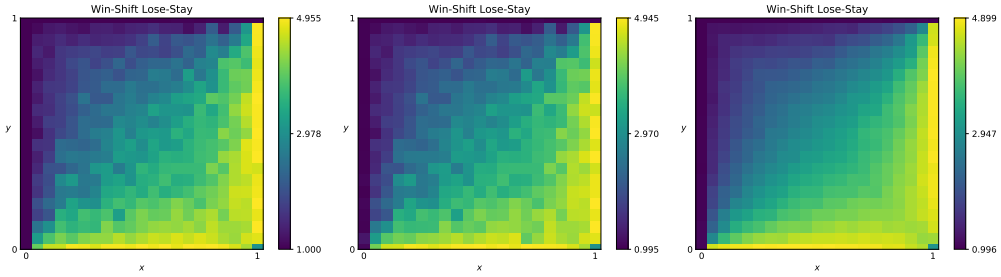
Figure 7.20: Ashlock's fingerprints for the LSTM strategies based on the StoS network when **Pavlov** is the probe strategy.



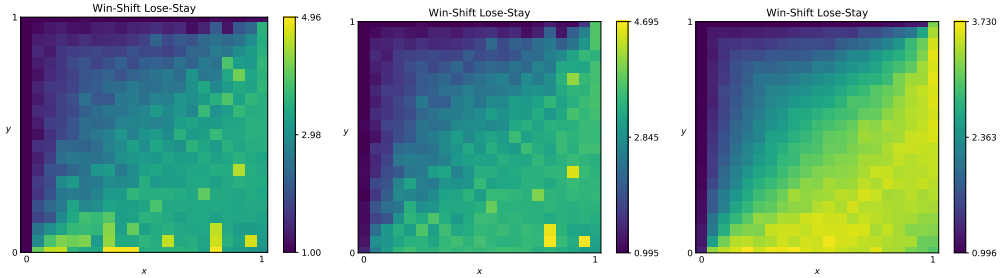
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.



(c) Strategies trained against the representative strategies.



(d) Strategies trained against basic strategies.

Figure 7.21: Ashlock's fingerprints for the LSTM strategies based on the StoP network when **Pavlov** is the probe strategy.

Ashlock fingerprints do not give an immediate qualifiable understanding of behaviour and so to further explore the similarities of the strategies a set of more interpretable fingerprints, the transitive fingerprints implemented in APL, have also been generated. The transitive fingerprints represent the cooperation rate of a strategy against a set of opponents over a number of turns. There are three set of opponents used here to generate the transitive fingerprints. These are the collection of strategies from [274] and from [43], and a spectrum of Random opponents with varying cooperating probability p. The transitive fingerprints are given by Figures 7.22 -7.27.

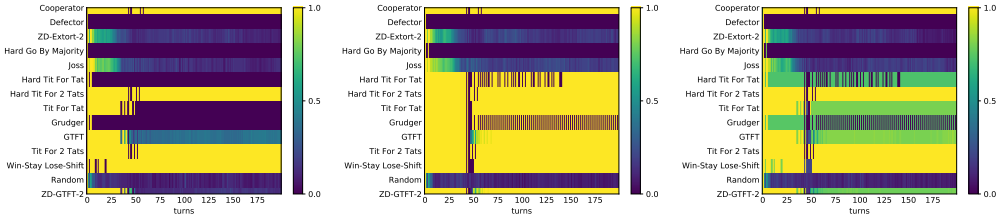
The differences between the strategies are more distinct using the transitive fingerprinting method. The transitive fingerprints demonstrate that the LSTM strategies do behave differently against the same list of opponents and that the strategies that open with cooperation achieve a higher cooperation rate compared to the strategies (on the same row) that do not.

Furthermore, the LSTM strategies appear to be using the opening moves to make up their minds regarding their opponents. Following the opening moves the strategies decide on a play. This is demonstrated by the fact that there is some variation in the opening moves of each fingerprint, but following the opening turns the patterns became more stable. An exception to this can be seen from the transitive fingerprints against the spectrum of Random opponents (Figures 7.26 and 7.27). It can be seen that this is not true for the strategies trained against the top performing strategies. This set of LSTM strategies do not make their mind regarding their opponent. In fact, following the opening moves the strategies just play a series of alternating sets of cooperations and defections.

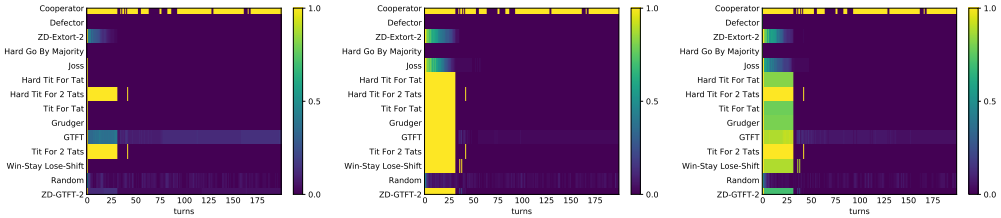
This could potentially reinforce the discussion that the training set against the top performing strategies is not diverse. Against strong opponents the top performing strategies simply cooperate. The trained LSTM strategies, on that training data set, have not been trained to react against random defections.

(a) Strategies trained on the entire training data set.

(b) Strategies trained against the top performing strategies.

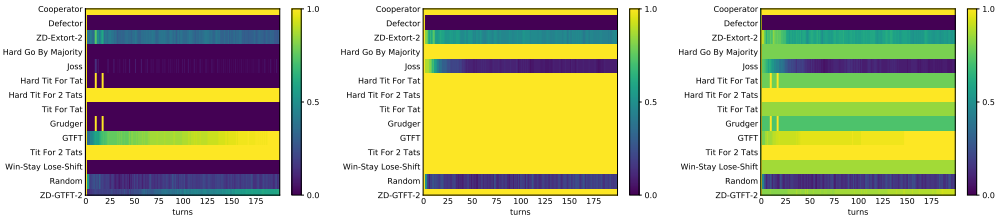


(c) Strategies trained against the representative strategies.

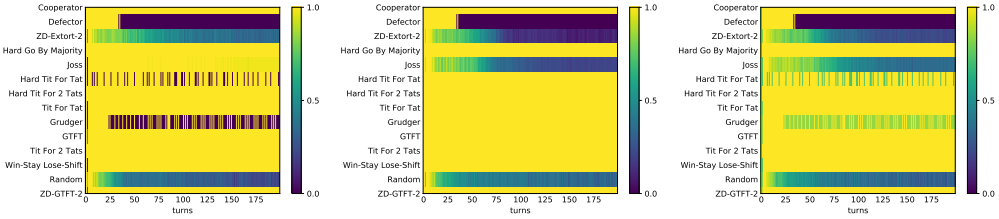


(d) Strategies trained against basic strategies.

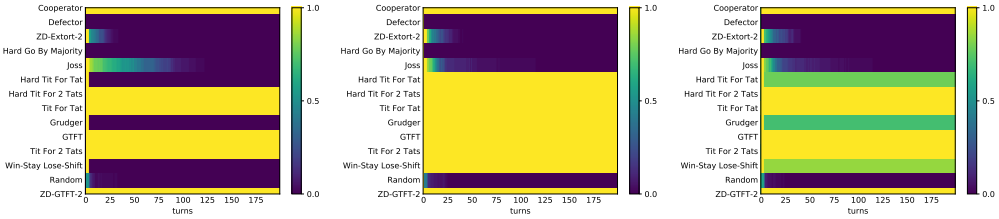
Figure 7.22: Transitive fingerprints for the LSTM strategies based on the StoS network against the list of opponents from [274].



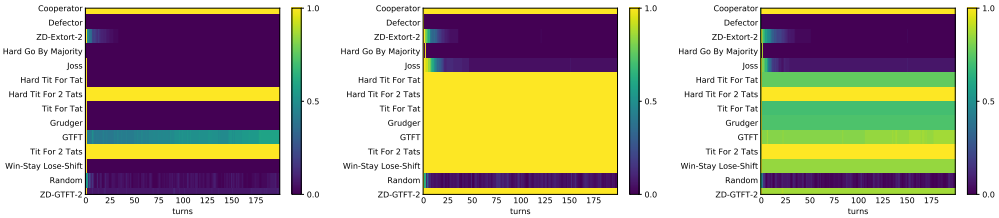
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.

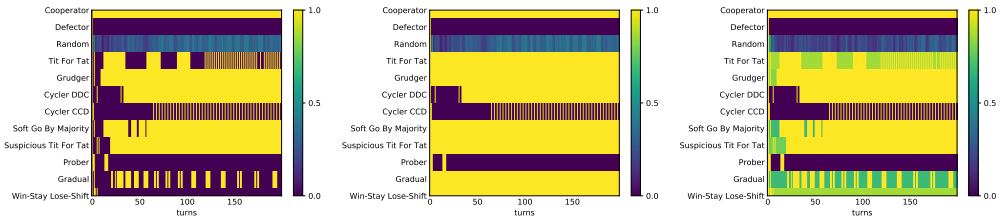


(c) Strategies trained against the representative strategies.

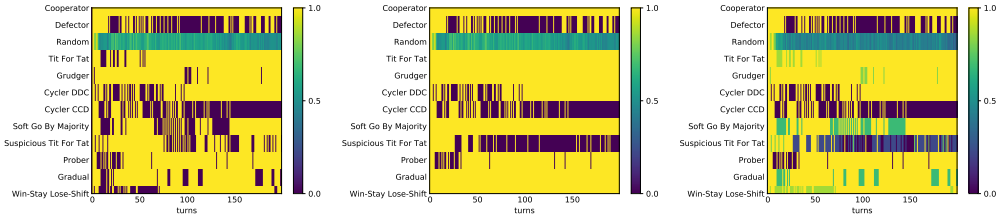


(d) Strategies trained against basic strategies.

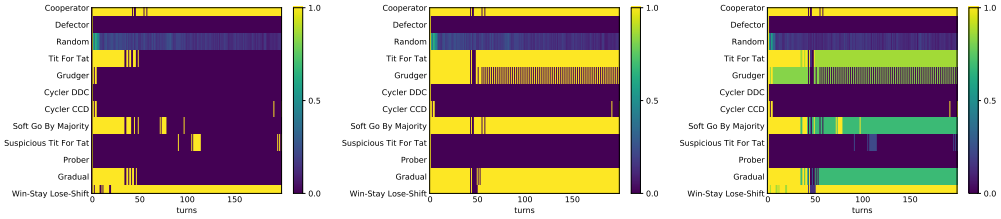
Figure 7.23: Transitive fingerprints for the LSTM strategies based on the StoP network against the list of opponents from [274].



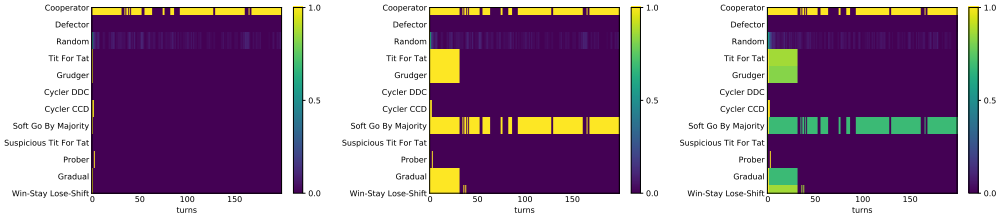
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.

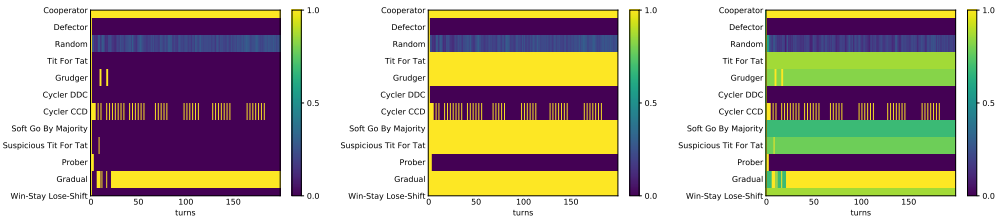


(c) Strategies trained against the representative strategies.

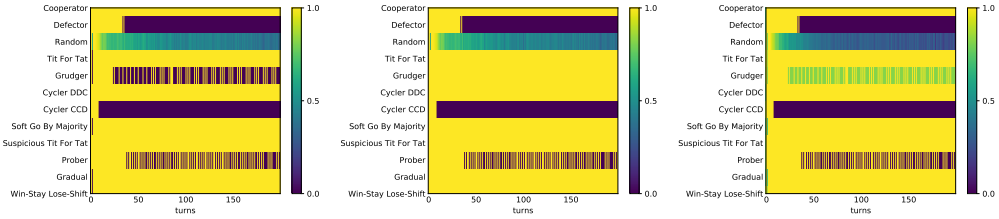


(d) Strategies trained against basic strategies.

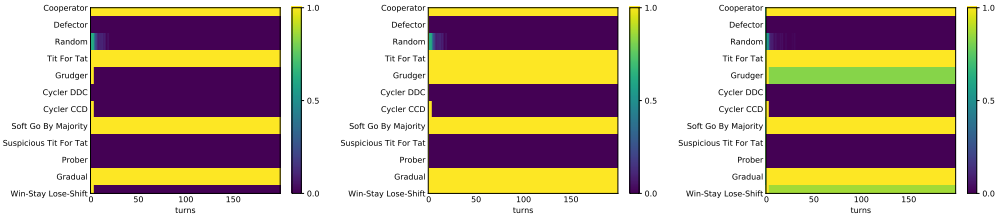
Figure 7.24: Transitive fingerprints for the LSTM strategies based on the StoS network against the list of opponents from [43].



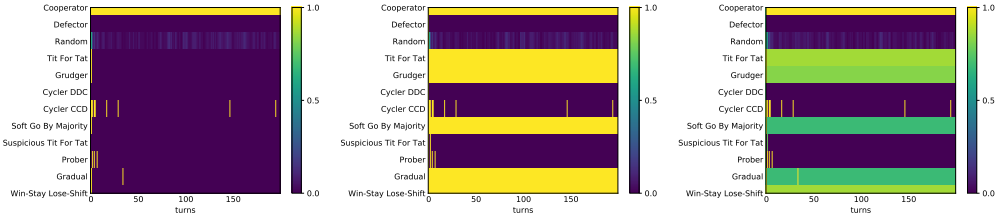
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.

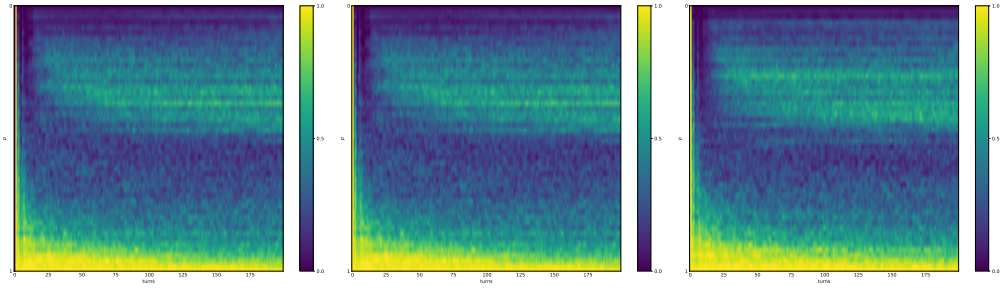


(c) Strategies trained against the representative strategies.

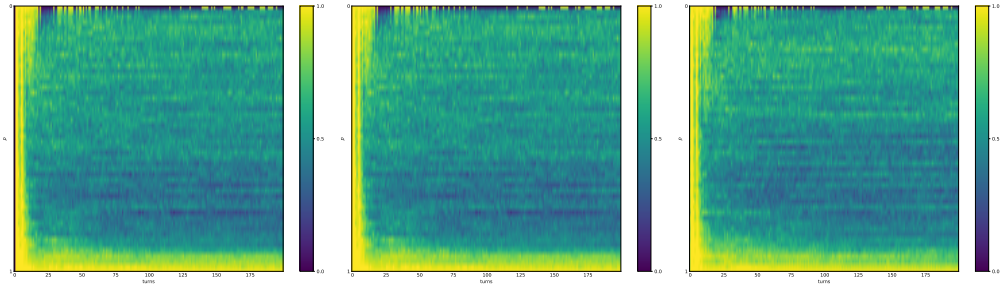


(d) Strategies trained against basic strategies.

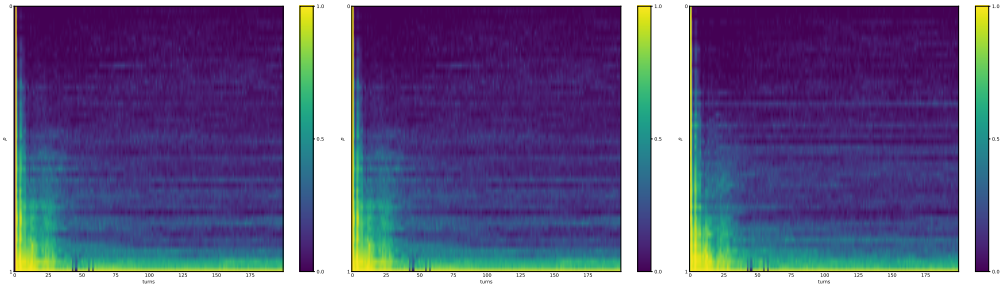
Figure 7.25: Transitive fingerprints for the LSTM strategies based on the StoP network against the list of opponents from [43].



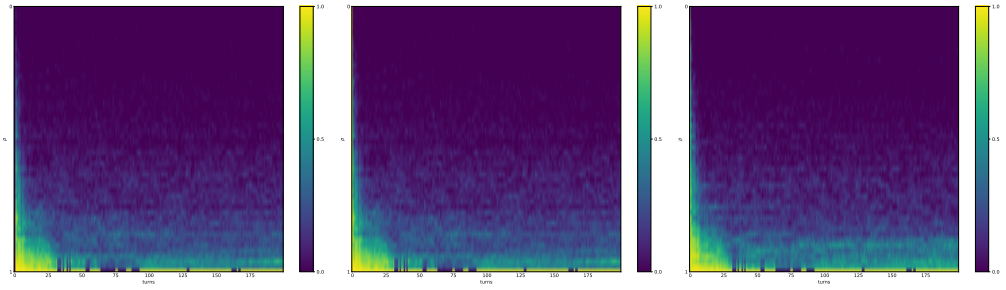
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.

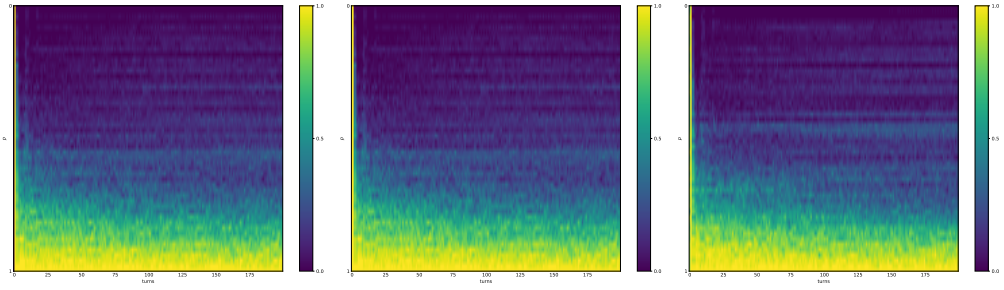


(c) Strategies trained against the representative strategies.

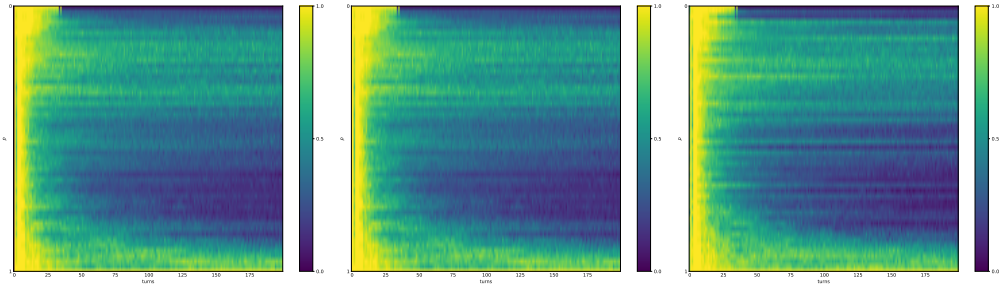


(d) Strategies trained against basic strategies.

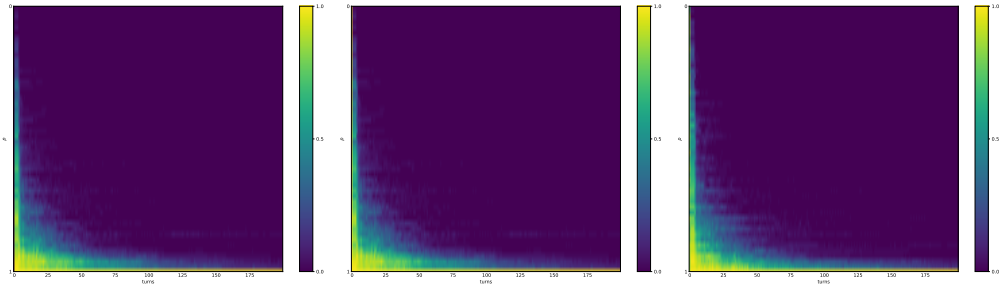
Figure 7.26: Transitive fingerprints for the LSTM strategies based on the StoS network against a list of **Random** opponents.



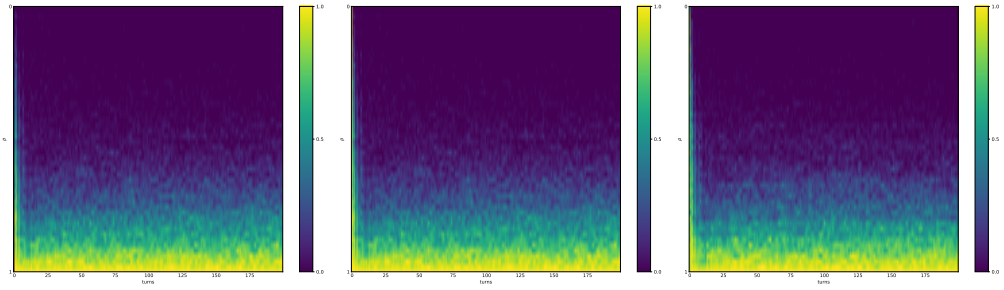
(a) Strategies trained on the entire training data set.



(b) Strategies trained against the top performing strategies.



(c) Strategies trained against the representative strategies.



(d) Strategies trained against basic strategies.

Figure 7.27: Transitive fingerprints for the LSTM strategies based on the StoP network against a list of **Random** opponents.

In order to gain a further understanding of the behaviour of the LSTM strategies produced by the training, the top performing LSTM strategies are put against a list of manually selected opponents. These are:

1. Tit For Tat. A strategy that retaliates a defection but also forgives if the opponent apologises. The strategy was selected to explore whether the LSTM strategies try to exploit the strategy, and whether they apologise after being in DD.

2. Gradual. Plays in a similar fashion as Tit For Tat but retaliates with a growing number of defection. Gradual was selected for the same reason as Tit For Tat.

3. Cooperator. A strategy that can been taken advantage of. The strategy was selected to explore whether the LSTM strategies do exploit the strategy.

4. Alternator. Another strategy that does not react to the history and can be taken advantage of.

5. Defector. A strategy that just defects. It was selected to inspect whether the LSTM strategies defend themselves from unconditional defections.

6. ZDExtort2. A strategy that exploits its opponents. The strategy was chosen to see if the LSTM players protect themselves from being exploited.

7. TF1. The strategy was presented in section 6.4. The strategy includes a handshake. The strategies are matched against TF1 to investigate whether they have developed the handshake.

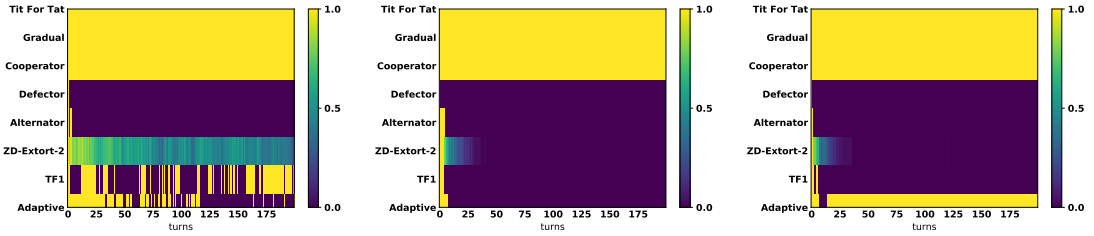
8. Adaptive. A strategy discussed in Chapter 6. The strategy has a unique set of best responses. It can be exploited to unconditionally cooperate while the opponent defects.

Three LSTM players are matched against these strategies in a tournament of 200 turns and 50 repetitions:

- The StoS based strategy, trained against all strategies with $p_o = 1$.
- The StoP based strategy, trained against the representative strategies with $p_o = 1$.
- The StoP based strategy, trained against the basic strategies with $p_o = 1$.

The median scores of each tournament are given by Tables 7.7- 7.9 and the transitive fingerprints of the three LSTM strategies against the selected opponents are given by

Figure 7.28.



(a) StoS network trained against all strategies with $p_o = 1$.

(b) StoP network trained against the representative strategies with $p_o = 1$.

(c) StoP network trained against the basic strategies with $p_o = 1$.

Figure 7.28: Transitive fingerprints for the top performing LSTM strategies against a list of manually selected strategies.

	LSTM strategy	Tit For Tat	Gradual	Cooperator	Defector	Alternator	ZD-Extort-2	TF1	Adaptive
LSTM strategy	3.0000	3.0000	3.000	3.0000	0.9900	2.9800	1.8221	0.8950	0.670
Tit For Tat	3.0000	3.0000	3.000	3.0000	0.9950	2.4900	1.0893	1.0400	2.955
Gradual	3.0000	3.0000	3.000	3.0000	0.8250	2.6850	1.5835	0.8800	2.945
Cooperator	3.0000	3.0000	3.000	3.0000	0.0000	1.5000	2.2146	1.5150	0.090
Defector	1.0400	1.0200	1.700	5.0000	1.0000	3.0000	1.0408	1.0400	1.120
Alternator	0.5550	2.5150	1.310	4.0000	0.5000	2.0000	1.7567	1.1750	0.605
ZD-Extort-2	2.3981	1.1143	2.087	3.5236	0.9898	2.5437	1.0824	1.0586	2.953
TF1	2.5950	1.0650	1.705	3.9900	0.9900	2.8250	1.0976	2.9900	1.080
Adaptive	2.6200	2.9550	2.995	4.9400	0.9700	2.9550	2.0210	1.0550	2.950

Table 7.7: Median scores of a standard tournament of 200 turns that was repeated 50 times. The LSTM strategy corresponds to the strategy based on the StoS network trained against all strategies with $p_o = 1$.

	LSTM strategy	Tit For Tat	Gradual	Cooperator	Defector	Alternator	ZD-Extort-2	TF1	Adaptive
LSTM strategy	3.0000	3.0000	3.0000	3.0000	0.9950	2.9600	1.1189	1.0250	4.8300
Tit For Tat	3.0000	3.0000	3.0000	3.0000	0.9950	2.4900	1.0954	1.0400	2.9550
Gradual	3.0000	3.0000	3.0000	3.0000	0.8250	2.6850	1.5264	0.8800	2.9450
Cooperator	3.0000	3.0000	3.0000	3.0000	0.0000	1.5000	2.2989	1.5150	0.0900
Defector	1.0200	1.0200	1.7000	5.0000	1.0000	3.0000	1.0384	1.0400	1.1200
Alternator	0.5850	2.5150	1.3100	4.0000	0.5000	2.0000	1.7972	1.1750	0.6050
ZD-Extort-2	1.1484	1.1204	2.0449	3.4674	0.9904	2.5247	1.0793	1.0604	2.8963
TF1	1.0500	1.0650	1.7050	3.9900	0.9900	2.8250	1.0659	2.9900	1.0800
Adaptive	0.1550	2.9550	2.9950	4.9400	0.9700	2.9550	1.9138	1.0550	2.9500

Table 7.8: Median scores of a standard tournament of 200 turns that was repeated 50 times. The LSTM strategy corresponds to the strategy based on the StoP network trained against the representative strategies with $p_o = 1$.

It is demonstrated that all three LSTM strategies achieve mutual cooperation for all 200 turns when matched against **Tit For Tat**, **Gradual** and **Cooperator**. The LSTM strategies open with cooperation and are never the first ones to defect. However, they quickly defend themselves against unconditional defections made by **Defector**, and

	LSTM strategy	Tit For Tat	Gradual	Cooperator	Defector	Alternator	ZD-Extort-2	TF1	Adaptive
LSTM strategy	3.0000	3.0000	3.0000	3.00	0.995	2.9850	1.0919	1.0400	2.9750
Tit For Tat	3.0000	3.0000	3.0000	3.00	0.995	2.4900	1.1134	1.0400	2.9550
Gradual	3.0000	3.0000	3.0000	3.00	0.825	2.6850	1.5399	0.8800	2.9450
Cooperator	3.0000	3.0000	3.0000	3.00	0.000	1.5000	2.2650	1.5150	0.0900
Defector	1.0200	1.0200	1.7000	5.00	1.000	3.0000	1.0400	1.0400	1.1200
Alternator	0.5350	2.5150	1.3100	4.00	0.500	2.0000	1.8194	1.1750	0.6050
ZD-Extort-2	1.1109	1.1384	2.0329	3.49	0.990	2.5149	1.0909	1.0598	2.9747
TF1	1.0650	1.0650	1.7050	3.99	0.990	2.8250	1.0823	2.9900	1.0800
Adaptive	2.9250	2.9550	2.9950	4.94	0.970	2.9550	1.9897	1.0550	2.9500

Table 7.9: Median scores of a standard tournament of 200 turns that was repeated 50 times. The LSTM strategy corresponds to the strategy based on the StoP network trained against the basic strategies with $p_o = 1$.

quickly learn to exploit **Alternator**. Following the opening 2 to 4 moves the LSTM strategies decide on unconditional defections against **Alternator**.

The StoP based strategy trained against the basic strategies with $p_o = 1$, is the strategy that reacts quickest to **Alternator**'s alternate defections. In fact, against **Alternator** the strategy demonstrates a **Grudger** like behaviour. However, this is not true for every opponent the strategy is matched against. Figure 7.28 shows that strategy manages mutual cooperation with **Adaptive** following a series of mutual defections. Thus, the LSTM strategy exhibits more adaptable behaviour than **Grudger**.

The biggest difference between the three LSTM strategies behaviours are when matched with **ZDExtort2**, **TF1** and **Adaptive**.

Initially, against **TF1** none of the three strategies carry out **TF1**'s specific handshake. The StoP strategies have a **Grudger** behaviour against the strategy, and go into mutual defections following the opening moves. The StoS strategy demonstrates a more varying behaviour against the strategy, which includes a series of cooperations and defections. Amongst the three strategies, the StoS strategy achieves the highest score per turn against **TF1**. The two StoP strategies also demonstrate a more aggressive behaviour against **ZDExtort2**. In comparison, the StoS sequence manages to achieve a higher cooperation rate against **ZDExtort2** which subsequently results in a better score.

Amongst the three strategy the most aggressive appears to be the StoP based strategy trained against the representative strategies. Even against **Adaptive** the strategy goes into mutual defection. The strategy does exhibit a more **Grudger** behaviour than the rest of the LSTM strategies. However, as it can be seen against **ZDExtort2** the strategy is not as provocable as **Grudger**. The StoP based strategy trained against the

basic strategies also exhibited a more adaptable behaviour, however, it is still a quite aggressive strategy. The LSTM strategy based on the StoS network trained against the entire data set is less aggressive, appears to achieve series of both mutual cooperation and defection and it is the best performing strategy amongst the LSTMs.

The three LSTM strategies when matched against each other achieve mutual cooperation. The median scores of a standard tournament of 200 turns and 50 repetitions with the three best performing LSTM strategies are given by Table 7.10. Being able to achieve mutual cooperation when competing against each other is a strong property of the LSTM strategies. As was discussed in Chapter 5, self interactions are important in evolutionary dynamic settings.

	StoS strategy trained against all	StoP strategy trained against the representative strategies	StoP strategy trained against basic strategies
StoS strategy trained against all	3.0	3.0	3.0
StoP strategy trained against the representative strategies	3.0	3.0	3.0
StoP strategy trained against basic strategies	3.0	3.0	3.0

Table 7.10: Median scores of a standard tournament with the three best performing LSTM strategies. The tournament is of 200 turns and of 50 repetitions.

On the whole, the high performing LSTM strategies appear to have the following properties:

1. Never defect first.
2. Are complex by design.
3. Use the opening moves to make up their mind about their opponents and decide on a play.
4. Can achieve mutual cooperation following mutual defections.

7.4.2 Stochastic LSTM strategies

The LSTM strategies that have been considered so far make a deterministic decision based on the networks predictions. Another variation of the LSTM strategies that has been considered are strategies that make a probabilistic choice based on the prediction. A strategy class called the `StochasticLSTMPlayer` has been implemented to simulate the behaviour of these strategies. The source code for the `StochasticLSTMPlayer` is given by Figure 7.29.

```python
import numpy as np

import axelrod as axl
from axelrod.random_ import random_choice
from keras.layers import LSTM, Dense, Dropout
from keras.models import Sequential

C, D = axl.Action.C, axl.Action.D


class StochasticLSTMPlayer(axl.Player):
name = "Stochastic LSTM Player"
classifier = {
    "memory_depth": float("inf"),
    "stochastic": True,
    "inspects_source": False,
    "manipulates_source": False,
    "manipulates_state": False,
}

def __init__(self, model, reshape_history_funct, opening_probability=0.78):
    self.model = model
    self.opening_probability = opening_probability
    self.reshape_history_function = reshape_history_funct
    super().__init__()

def strategy(self, opponent):
    if len(self.history) == 0:
        return random_choice(self.opening_probability)

    history = [action.value for action in opponent.history]
    prediction = float(
        self.model.predict(self.reshape_history_function(history))[0][-1]
    )

    return random_choice(prediction)

def __repr__(self):
    return self.name
```

Figure 7.29: Implementation of the `StochasticLSTMPlayer` class.

The 24 strategies that make a probabilistic decision at each turn are evaluated on the same 300 tournaments as the strategies in section 7.4. The $\bar{r}$ for the strategies are given by Table 7.11.

	sequence to sequence			**sequence to probability**		
	$p_o = 0$	$p_o = 1$	$p_o = 0.78$	$p_o = 0$	$p_o = 1$	$p_o = 0.78$
All strategies	0.833	0.800	0.800	0.800	0.750	0.778
Top strategies	0.800	0.800	0.789	0.778	0.714	0.714
Representative strategies	0.750	0.700	0.707	0.667	0.600	0.600
Basic strategies	0.833	0.700	0.714	0.857	0.700	0.714

Table 7.11: The median normalised ranks of the 24 LSTM strategies that make stochastic decisions. A $\bar{r}$ closer to 0 indicates a more successful performance.

The results of Table 7.11 demonstrate that the strategies that make a probabilistic decision instead of a deterministic one (following the opening turn) performed very poorly. The smallest $\bar{r}$ has a value of 0.6. Thus, the most successful strategy on average performs on the bottom half of the tournaments it participated in.

7.5 Chapter summary

This Chapter has introduced a total of 24 new IPD strategies based on LSTM networks. The advantage of using LSTMs, contrary to feed forward networks, is that LSTMs incorporate a mechanism of memory. This allows the networks to learn to using the history of an opponent in order to decide their next move. The collection of best response sequences of Chapter 6 was purposely generated so that LSTMs could be trained to play optimally against a list of known IPD strategies.

Two types of LSTM networks have been trained in this Chapter. Presented in section 7.3, there were referred to as the sequence to sequence (StoS) network and the sequence to probability (StoP) network. The two networks were trained on a training data set generated by the collection of best response strategies, but also on three subsets of that training data set. This was done in order to understand the effect of the training samples on the network's performance in a IPD tournament. The subsets included the best response sequences to top performing strategies, to representative strategies with ranks across a standard tournament and a set of basic strategies. The networks were developed and trained using the open source package Keras, and the training process was carried out on a GPU.

A total of 8 LSTM networks were trained, and those corresponded to 24 strategies when the opening move was taken into account. The opening move of an LSTM strategy had to be manually defined, and three different values were chosen to carry out the evaluation analysis. These were $p_o = 0$, $p_o = 1$ and $p_o = 0.78$.

The performance of the 24 LSTM strategies was evaluated in 300 standard tournaments. The results of the meta tournament analysis demonstrated that the strategies trained on the entire data set performed well in the 300 standard tournaments regardless the LSTM network type. Moreover, the strategies trained on the subsets performed well only when they were trained using the StoP network. Finally, it was shown that the strategies that performed well in the analysis of this Chapter have a high probability of ranking in the top half of any standard tournament. This demonstrates that LSTM strategies trained on a collection of sequences can lead to successful behaviours in the IPD. This unsupervised approach is the first contribution of this type at the intersection of deep learning and game theory.

An interesting result demonstrated by the analysis was the effect of opening with a cooperation. The most successful strategies of this Chapter have been the strategies that cooperated on the first turn. The transitive fingerprints demonstrated that it was because these strategies achieve a higher cooperation rate compared to the rest of the trained strategies. This result reinforces the discussion started by Axelrod: opening with a cooperation is a property that successful strategies in a IPD competition need to have.

Chapter 8

Conclusions

This Chapter serves to summarise the work and contributions of this thesis. Each chapter contains a detailed chapter summary section, and so the summary here will be brief.

8.1 Research summary

The fundamental research question of this thesis has been the same question that has troubled the scientific community since the formulation of the IPD in 1950. Namely, what is the optimal behaviour an Iterated Prisoner's Dilemma (IPD) strategy should adapt as a response to different environments.

Chapter 1 introduced the IPD, carried out an initial literature review and outlined the research tasks of this thesis. A more detailed literature review was presented in Chapter 2. The literature reviewed in Chapter 2 was divided into different research topics. These included evolutionary dynamics, intelligently designed strategies, structured strategies and training, and software that has been developed specifically for the game.

In Chapter 3 a bespoke research software tool called Arcas was developed and used to collect a data set of articles' metadata on the IPD. A topic modelling technique, called Latent Dirichlet Allocation, was applied to the abstracts of these articles and allocated them into five different research topics. These were human subject research, biological studies, strategies, evolutionary dynamics on networks and modelling problems as a Prisoner's Dilemma (PD).

The bespoke data set was further analysed to explore whether the academic field of the PD is cooperative and whether there is influence between the authors. It was shown that the field of the IPD is a collaborative field, yet it is not necessarily more collaborative than other fields. Many authors tend to collaborate with authors from one community and are not involved in multiple communities. The collaborativeness was also explored over time, and it was shown that since the first publications authors tended to write only with a single community and that it is not an effect of a specific time period. Exploring the influence of authors in the field based on the specific publications showed that authors do not gain much influence, and the only ones with influence are the ones connected to a "main" group.

Chapter 4 examined the performance of a collection of 195 strategies in the largest collection of computer tournaments in the field. The results across the 45,600 tournaments of various tournaments types deduced that there was not a single strategy that performs well in diverse IPD scenarios. The later parts of the Chapter analysed and extracted the salient features of the best performing strategies across the various tournament types and established that there are several properties that heavily influence the best performing strategies. There were: be nice, be provocable and forgiving, be a little envious, be clever, and adapt to the environment.

Chapter 5 investigated best response memory-one strategies with a theory of mind. It presented several theoretical and numerical results. More specifically, it proved that:

- The utility of a memory-one strategy against a set of memory-one opponents can be written as a sum of ratios of quadratic forms.
- There is a compact way of identifying a memory-one best response to a group of opponents through a search over a discrete set.
- There is a condition for which in an environment of memory-one opponents defection is the stable choice, based only on the coefficients of the opponents.

Additionally, the numerical results of Chapter 5 reinforced established result of the literature. Namely, they showed that extortionate play is not always optimal by showing that optimal play is often not extortionate, and that memory-one strategies suffer from their limited memory in multi agent interactions and can be out performed by optimised strategies with longer memory.

Chapter 6 also investigated best responses but in the form of sequences. It defined an IPD strategy in a finite match as a sequence, and it defined the best response sequence against a given opponent. It introduced an evolutionary algorithm which demonstrated that it can successfully identify best response sequences, and which was used to estimate best response sequences to 192 strategies. A total of 5,258 best responses sequences were obtained as result of Chapter 6.

The purpose of this collection of best response sequences was to be used in Chapter 7 to train a series of long short-term memory (LSTM) networks. These networks were trained on the collection, and on three subsets of the collection, to predict best response sequences. A total of 8 LSTMs were trained, and these were used to introduce a total of 24 distinct IPD strategies. The results of 300 standard tournaments demonstrated that a set of these LSTM strategies can successfully be on the top ranks of any given standard tournament. The top performing LSTM strategies exhibited distinct behaviours against the same opponents, demonstrated that are adaptable and that opening with a cooperation is crucial to a successful performance.

8.2 Contributions

This thesis has made novel contributions across various themes. Numerous research software packages have been implemented as part of this thesis. These packages have been written following the highest standards of software development, and have been made available so that other researchers can contribute to and use them. The packages include Arcas a tool designed for scraping academic articles from various APIs and `sequences-sensei` a project for performing genetic algorithms. Additionally, software contributions were made to well established Python libraries such as SymPy [203] and Axelrod-Python Library [7].

A total of six accompanying data sets have been generated as a result of this thesis, which include one of the largest collection of IPD tournaments known to the field:

1. Articles' meta data on the Prisoner's Dilemma [100].
2. Articles' meta data on the Price of Anarchy [99].
3. Articles' meta data on Auction Games [98].
4. Raw data for: "Stability of defection, optimisation of strategies and the limits of memory in the Prisoner's Dilemma" [101].

5. A data set of 45686 Iterated Prisoner's Dilemma tournaments' results [97, 102]
6. Best response sequences in the Prisoner's Dilemma [103].

These have been archived and made available via Zenodo, and likewise, are available to other researchers. They can be used to conduct further analysis and provide new insights to the field.

A total of four scientific manuscripts presenting the methodology, analysis and results of this thesis have been prepared and three of them are currently under submission to respective academic journals. The title of these manuscripts are:

1. A bibliometric study of research topics, collaboration and influence in the field of the Iterated Prisoner's Dilemma [111].
2. Properties of winning Iterated Prisoner's Dilemma strategies [218].
3. Recognising and evaluating the effectiveness of extortion in the Iterated Prisoner's Dilemma [166].
4. A theory of mind: Best responses to memory-one strategies. The limitations of extortion and restricted memory [112].

These manuscripts have been uploaded on the pre print server arXiv and are currently available and accessible to the scientific community.

Designing new strategies is an important type of research for the field. This thesis has introduced an abundant number of properties of successful strategies which can be of interest to researchers designing a new strategy for new environments, or just to understand the reasons behind some strategies being better than others. Complementing this, a new mathematical framework has been developed for the better understanding of memory-one strategies and an an initial understanding of using recurrent neural networks to train IPD strategies has been presented.

This thesis has contributed to the continuous understanding of the emergence of cooperation by providing a condition for which cooperation can not occur in memory-one environments. It has also has proven that constrained quadratic ratio optimisation problems that are non concave can be solved explicitly by using resultant theory.

Finally, compared to conventional works where a strategy is trained against a specific set of opponents and its performance is then validated against that same set, this thesis has trained LSTM networks on data sets of best response sequences and then validated

their performance as IPD strategies is 300 different tournaments. This has shown the potential for an unsupervised learning approach to training a recurrent neural network to compete in IPD competitions.

8.3 Complementary research

The results of this thesis are not the only scientific results to which I contributed during this doctoral research. The publications that will be discussed in this section are publications to which I am an author.

Two other projects which focused on the IPD have been [121, 122]. The works of [121, 122] focused on the usage of reinforcement learning algorithms (genetic algorithms and particle swarm optimisation algorithms) in training a series of strategies based on different structures such as finite state machines, hidden Markov models and neural networks. These strategies were trained in two settings:

- A Moran process which is an evolutionary model of invasion and resistance across time during which high performing individuals are more likely to be replicated.
- A standard tournament.

The results of [121] were confirmed in Chapter 4. The trained strategies performed at the top of the standard tournament surpassing well established strategies such as **Tit For Tat**, **Pavlov**, **Gradual** and zero-determinant strategies. In [122] it was observed that the trained strategies (with no manual input) evolved the ability to have a handshake, to recognise themselves. This was particularly important in a Moran process of resisting invasion where a single individual of another type is introduced and the strategies need to resist the invasion.

Another undertaken project included exploring rhino poaching behaviour using evolutionary game theory [109]. Rhino populations are at critical level today and in protected areas devaluation approaches are used to secure the life of the animals. The effectiveness of these approaches, however, relies on poachers behaviour as they can be selective and not kill devalued rhinos or indiscriminate. Populations of differently behaving poachers were modelled using evolutionary game theory. The results demonstrated that full devaluation of all rhinos is likely to lead to indiscriminate poaching and that devaluating of rhinos can only be effective when implemented along with a strong disincentive framework. The paper aimed to contribute to the necessary re-

search required for an informed discussion about the lively debate on legalising rhino horn trade.

Finally, delivering science outreach workshops is a great way to gain a deeper understanding of science and its applications, and enhancing students interest in science. With that in mind I created an open source educational tutorial, called Game Theory and Python [110], aimed at introducing participants to game theory and more specifically to repeated games. The tutorial is aimed at two groups of individuals: individuals familiar with Python (programmers) who want to start to learn game theory and mathematicians with little or no programming knowledge as a pathway to programming through the interesting subject. The tutorial has gained much interest and is currently under submission at the Journal of Open Source Education.

A full list of the publications produced during the research presented in this section is:

1. Reinforcement Learning Produces Dominant Strategies for the Iterated Prisoner's Dilemma [121].
2. Evolution Reinforces Cooperation with the Emergence of Self-Recognition Mechanisms: an empirical study of the Moran process for the iterated Prisoner's dilemma [122].
3. An Evolutionary Game Theoretic Model of Rhino Horn Devaluation [109].
4. Game Theory and Python [110].

8.4 Future research directions

Each part of this thesis has given rise to further interesting questions and research directions that, although not in the scope of the current work, would improve or compliment it.

Future research - Meta tournament Analysis

In Chapter 4 during the data collection the probability of noise was allowed to vary between values of 0 and 1. However, it was established that large values of noise (> 0.1) caused an impactful variation to the environment. From the collection of 195 strategies considered in the Chapter there was not a single strategy that performed well in that spectrum of noise.

Strategies that have been trained specifically for noisy environments such as DBS, Evolved FSM 16 Noise 05, Evolved ANN 5 Noise 05, PSO Gambler 2 2 2 Noise 05 and Omega Tit For Tat, performed adequately only in tournaments with restricted noise. This indicates that possibly there is not a strategy in the literature trained to be effective for a broad spectrum of noise values. Training such a strategy would be an interesting avenue of further research. The analysis of the top performances would then be reproduced whilst including the new trained strategy.

Future research - Memory-one strategies

In Chapter 5 the empirical results supported that extortionate play is not always optimal and that memory-one strategies suffer from their limited memory in multi agent interactions. All the empirical results presented have been for the case of two opponents ($N = 2$). A future research direction would be to validate the empirical results of the Chapter for larger values of N.

Another restricted set of strategies on memory that have been studied in the literature are memory-two strategies. These are strategies that take into account the past two turns of the match. A compelling research question that arises is whether the current formulation of Chapter 5 can be expanded to include memory-two strategies, and whether the results still hold.

Future research - Training an LSTM strategy

An interesting question that was raised in Chapter 7 was whether the sequence to probability based strategy trained on the entire data set would perform even better if it was trained for longer. An interesting avenue of further research would be to train the specific strategy for more epochs, and to evaluate its performance again in the meta tournament analysis presented in Chapter 7. Finally, another avenue of further research would be to explore the effect of the dimensionality of the hidden layers in the performance of the LSTM networks as IPD strategies.

Bibliography

[1] Complexity of cooperation web site. `http://www-personal.umich.edu/~axe/research/Software/CC/CC2.html`. Accessed: 2017-10-23.

[2] The evolution of trust. `http://ncase.me/trust/`. Accessed: 2017-10-23.

[3] The iterated prisoner's dilemma game. `http://selborne.nl/ipd/`. Accessed: 2017-10-23.

[4] Lifl (1998) prison. `http://www.lifl.fr/IPD/ipd.frame.html`. Accessed: 2017-10-23.

[5] PLOS public library of science. https://www.plos.org/.

[6] The prisoner's dilemma. http://www.prisoners-dilemma.com/, 2017.

[7] The Axelrod project developers . Axelrod: 4.4.0, April 2016.

[8] E. A. Coopsim v0.9.9 beta 6. `https://github.com/jecki/CoopSim/`, 2015.

[9] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[10] K. M. Abadir and J. R. Magnus. *Matrix algebra*, volume 1. Cambridge University Press, 2005.

[11] M. Aberdour. Achieving quality in open-source software. *IEEE software*, 24(1):58–64, 2007.

[12] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, A. M. Umar, O. U. Linus, H. Arshad, A. A. Kazaure, U. Gana, and M. U. Kiru. Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access*, 7:158820–158846, 2019.

[13] C. Adami and A. Hintze. Evolutionary instability of zero-determinant strategies demonstrates that winning is not everything. *Nature communications*, 4:2193, 2013.

[14] A. Agarwal and B. Triggs. Multilevel image coding with hyperfeatures. *International Journal of Computer Vision*, 78(1):15–27, 2008.

[15] A. Akritas, G. Malaschonok, and P. Vigklas. Sturm sequences and modified subresultant polynomial remainder sequences. *Serdica Journal of Computing*, 8(1):29–46, 2014.

[16] N. Anastassacos and M. Musolesi. Learning through probing: a decentralized reinforcement learning architecture for social dilemmas. *arXiv preprint arXiv:1809.10007*, 2018.

[17] Apache. Subversion. `https://subversion.apache.org/`.

[18] M. Archetti. Evolutionary game theory of growth factor production: implications for tumour heterogeneity and resistance to therapies. *British journal of cancer*, 109(4):1056, 2013.

[19] M. Archetti and K. J. Pienta. Cooperation among cancer cells: applying game theory to cancer. *Nature Reviews Cancer*, page 1, 2018.

[20] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[21] D. Ashlock, J. A. Brown, and P. Hingston. Multiple opponent optimization of prisoner's dilemma playing agents. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(1):53–65, 2015.

[22] D. Ashlock and E. Y. Kim. Techniques for analysis of evolved prisoner's dilemma strategies with fingerprints. 3:2613–2620 Vol. 3, Sept 2005.

[23] D. Ashlock and E. Y. Kim. Fingerprinting: Visualization and automatic analysis of prisoner's dilemma strategies. *IEEE Transactions on Evolutionary Computation*, 12(5):647–659, Oct 2008.

[24] D. Ashlock, E. Y. Kim, and W. Ashlock. Fingerprint analysis of the noisy prisoner's dilemma using a finite-state representation. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(2):154–167, June 2009.

[25] D. Ashlock, E. Y. Kim, and W. Ashlock. A fingerprint comparison of different prisoner's dilemma payoff matrices. pages 219–226, Aug 2010.

[26] D. Ashlock, E. Y. Kim, and N. Leahy. Understanding representational sensitivity in the iterated prisoner's dilemma with fingerprints. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(4):464–475, July 2006.

[27] W. Ashlock and D. Ashlock. Changes in prisoner's dilemma strategies over evolutionary time with different population sizes. In *2006 IEEE International Conference on Evolutionary Computation*, pages 297–304. IEEE, 2006.

[28] W. Ashlock and D. Ashlock. Changes in prisoner's dilemma strategies over evolutionary time with different population sizes. pages 297–304, 2006.

[29] W. Ashlock, J. Tsang, and D. Ashlock. The evolution of exploitation. In *2014 IEEE Symposium on Foundations of Computational Intelligence (FOCI)*, pages 135–142. IEEE, 2014.

[30] Atlassian. Bitbucket. `https://bitbucket.org/`.

[31] T. C. Au and D. Nau. Accident or intention: that is the question (in the noisy iterated prisoner's dilemma). In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 561–568. ACM, 2006.

[32] R. Axelrod. Effective choice in the prisoner's dilemma. *The Journal of Conflict Resolution*, 24(1):3–25, 1980.

[33] R. Axelrod. More effective choice in the prisoner's dilemma. *The Journal of Conflict Resolution*, 24(3):379–403, 1980.

[34] R. Axelrod. The emergence of cooperation among egoists. *American political science review*, 75(2):306–318, 1981.

[35] R. Axelrod. The evolution of strategies in the iterated prisoner's dilemma. *Genetic Algorithms and Simulated Annealing*, pages 32–41, 1987.

[36] R. Axelrod. Launching "the evolution of cooperation". *Journal of Theoretical Biology*, 299(Supplement C):21 – 24, 2012. Evolution of Cooperation.

[37] R. Axelrod and D. Dion. The further evolution of cooperation. *Science*, 242(4884):1385–1390, 1988.

[38] R. Axelrod and W. D. Hamilton. *The Evolution of Cooperation*. 1984.

[39] P. Baldi and P. J. Sadowski. Understanding dropout. In *Advances in neural information processing systems*, pages 2814–2822, 2013.

[40] D. Banerjee and S. Sen. Reaching pareto-optimality in prisoner's dilemma using conditional joint action learning. *Autonomous Agents and Multi-Agent Systems*, 15(1):91–108, 2007.

[41] J. S. Banks and R. K. Sundaram. Repeated games, finite automata, and complexity. *Games and Economic Behavior*, 2(2):97–117, 1990.

[42] G. Barbastathis, A. Ozcan, and G. Situ. On the use of deep learning for computational imaging. *Optica*, 6(8):921–943, 2019.

[43] B. Beaufils, J. P. Delahaye, and P. Mathieu. Our meeting with gradual: A good strategy for the iterated prisoner's dilemma. 1997.

[44] B. Beaufils, J. P. Delahaye, and P. Mathieu. Complete Classes of Strategies for the Classical Iterated Prisoner's Dilemma. 1447:33–41, 1998.

[45] R. Bell, L. Mieth, and A. Buchner. Separating conditional and unconditional cooperation in a sequential prisoner's dilemma game. *PloS one*, 12(11):e0187952, 2017.

[46] J. Bendor, R. M. Kramer, and S. Stout. When in doubt... cooperation in a noisy prisoner's dilemma. *The Journal of Conflict Resolution*, 35(4):691–719, 1991.

[47] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[48] F. Benureau and N. P. Rougier. Re-run, repeat, reproduce, reuse, replicate: transforming code into scientific contributions. *Frontiers in neuroinformatics*, 11:69, 2018.

[49] A. S. Berger. *Hardware and computer organization*. Newnes, 2005.

[50] T. Bergmann and R. Dale. A scientometric analysis of evolang: Intersections and authorships. In *The evolution of language: Proceedings of the 11th international conference (EVOLANGX11). http://evolang. org/neworleans/papers/182. html. Retrieved*, volume 22, 2018.

[51] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[52] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[53] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[54] R. Boyd and J. P. Lorberbaum. No pure strategy is evolutionarily stable in the repeated prisoner's dilemma game. *Nature*, 327:58–59, 1987.

[55] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[56] A. Carvalho, H. P. Rocha, F. T. Amaral, and F. G. Guimaraes. Iterated prisoner's dilemma-an extended analysis. 2013.

[57] B. Catanzaro, N. Sundaram, and K. Keutzer. Fast support vector machine training and classification on graphics processors. In *Proceedings of the 25th international conference on Machine learning*, pages 104–111, 2008.

[58] CERN. Zenodo. `https://zenodo.org`.

[59] K. Chellapilla and D. B. Fogel. Evolving neural networks to play checkers without relying on expert knowledge. *IEEE transactions on neural networks*, 10(6):1382–1391, 1999.

[60] X. Chen, F. Fu, and L. Wang. Influence of initial distributions on robust cooperation in evolutionary prisoner's dilemma. *arXiv preprint physics/0701318*, 2007.

[61] Y.-T. Chen, A. McAvoy, and M. A. Nowak. Fixation probabilities for any configuration of two strategies on regular graphs. *Scientific reports*, 6:39181, 2016.

[62] J. Choi, J. Yang, and H. Jo. The co-evolution of cooperation and trait distinction. In *Proceedings of the First Complexity Conference*, 2006.

[63] F. Chollet et al. Keras. `https://keras.io`, 2015.

[64] S. Y. Chong and X. Yao. Behavioral diversity, choices and noise in the iterated prisoner's dilemma. *IEEE Transactions on Evolutionary Computation*, 9(6):540–551, Dec 2005.

[65] T. CI. Travis ci. `https://travis-ci.org`.

[66] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[67] L. P. Coelho, T. Peng, and R. F. Murphy. Quantifying the distribution of probes between subcellular locations using unsupervised pattern unmixing. *Bioinformatics*, 26(12):i7–i12, 2010.

[68] S. J. Cox, T. Sluckin, and J. Steele. Group size, memory, and interaction rate in the evolution of cooperation. *Current Anthropology*, 40(3):369–376, 1999.

[69] T. Crick, B. A. Hall, S. Ishtiaq, and K. Takeda. """" share and enjoy""""": Publishing useful and usable scientific models. In *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 957–961. IEEE, 2014.

[70] P. J. Darwen and X. Yao. Why more choices cause less cooperation in iterated prisoner's dilemma. 2:987–994, 2001.

[71] R. das Neves Machado, B. Vargas-Quesada, and J. Leta. Intellectual structure in stem cell research: exploring brazilian scientific articles from 2001 to 2010. *Scientometrics*, 106(2):525–537, 2016.

[72] J. Delahaye. L'altruisme perfectionné. *Pour la Science (French Edition of Scientific American)*, 187:102–107, 1993.

[73] J. Delahaye. Logique, informatique et paradoxes, 1995.

[74] C. Donninger. *Is it Always Efficient to be Nice? A Computer Simulation of Axelrod's Computer Tournament.* Physica-Verlag HD, Heidelberg, 1986.

[75] Z. Drezner. Compounded genetic algorithms for the quadratic assignment problem. *Operations Research Letters*, 33(5):475–480, 2005.

[76] S. Dridi and E. Akçay. Learning to cooperate: The evolution of social rewards in repeated interactions. *The American Naturalist*, 191(1):58–73, 2018. PMID: 29244562.

[77] D. Easley, J. Kleinberg, et al. *Networks, crowds, and markets*, volume 8. Cambridge university press Cambridge, 2010.

[78] S. R. Eddy. Hidden markov models. *Current opinion in structural biology*, 6(3):361–365, 1996.

[79] H. Etzkowitz. Individual investigators and their research groups. *Minerva*, 30(1):28–50, 1992.

[80] G. W. Evans and C. M. Crumbaugh. Payment schedule, sequence of choice, and cooperation in the prisoner's dilemma game. *Psychonomic Science*, 5(2):87–88, Feb 1966.

[81] X. Feng and Y. Liu. Trilateral game analysis on information sharing among members in a virtual team. In *2008 IEEE Symposium on Advanced Management of Information for Globalized Enterprises (AMIGE)*, pages 1–5. IEEE, 2008.

[82] M. M. Flood. Some experimental games. *Management Science*, 5(1):5–26, 1958.

[83] D. B. Fogel, T. J. Hays, S. L. Hahn, and J. Quon. A self-learning evolutionary chess program. *Proceedings of the IEEE*, 92(12):1947–1954, 2004.

[84] N. Franken and A. P. Engelbrecht. Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma. *IEEE Transactions on evolutionary computation*, 9(6):562–579, 2005.

[85] P. Frazier, W. Powell, and S. Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS journal on Computing*, 21(4):599–613, 2009.

[86] P. I. Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[87] M. R. Frean. The prisoner's dilemma without synchrony. *Proceedings of the Royal Society of London B: Biological Sciences*, 257(1348):75–79, 1994.

[88] D. Fudenberg and E. Maskin. The folk theorem in repeated games with discounting or with incomplete information. In *A Long-Run Collaboration On Long-Run Games*, pages 209–230. World Scientific, 2009.

[89] T. Gaffney, M. Harper, and V. A. Knight. Memory depth of finite state machine strategies for the iterated prisoner's dilemma. *arXiv preprint arXiv:1912.04493*, 2019.

[90] P. S. Gallo and I. A. Dale. Experimenter bias in the prisoner's dilemma game. *Psychonomic Science*, 13(6):340–340, Jun 1968.

[91] M. Gaudesi, E. Piccolo, G. Squillero, and A. Tonda. Exploiting evolutionary modeling to prevail in iterated prisoner's dilemma tournaments. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(3):288–300, 2016.

[92] S. N. Geniole, A. E. Keyes, C. J. Mondloch, J. M. Carré, and C. M. McCormick. Facing aggression: Cues differ for female versus male faces. *PLOS one*, 7(1):e30366, 2012.

[93] Git-scm. Git. `https://git-scm.com`.

[94] GitLab. Gitlab. `https://about.gitlab.com/`.

[95] N. Glynatsi. Nikoleta-v3/Training-IPD-strategies-with-RNN: v1.0.1. `https://doi.org/10.5281/zenodo.3829971`, May 2020.

[96] N. Glynatsi, V. Knight, and Marc. Nikoleta-v3/meta-analysis-of-prisoners-dilemma- tournaments: v1.0.0. `https://doi.org/10.5281/zenodo.3766344`, April 2020.

[97] N. E. Glynatsi. A data set of 45686 Iterated Prisoner's Dilemma tournaments' results. `https://doi.org/10.5281/zenodo.3516652`, October 2019.

[98] N. E. Glynatsi. Articles' meta data on auction games. `https://doi.org/10.5281/zenodo.3406544`, September 2019.

[99] N. E. Glynatsi. Articles' meta data on the price of anarchy. `https://doi.org/10.5281/zenodo.3406542`, September 2019.

[100] N. E. Glynatsi. Articles' meta data on the prisoner's dilemma. `https://doi.org/10.5281/zenodo.3406536`, September 2019.

[101] N. E. Glynatsi. Raw data for: "Stability of defection, optimisation of strategies and the limits of memory in the Prisoner's Dilemma.". `https://doi.org/10.5281/zenodo.3402179`, September 2019.

[102] N. E. Glynatsi. A data set of 45686 Iterated Prisoner's Dilemma tournaments' results [RAW DATA]. `https://doi.org/10.5281/zenodo.3753498`, April 2020.

[103] N. E. Glynatsi. Best response sequences in the prisoner's dilemma. `https://doi.org/10.5281/zenodo.3685251`, February 2020.

[104] N. E. Glynatsi. Iterated Prisoner's Dilemma computer tournament with 218 strategies results: Axelrod-Python version 3.10. `https://doi.org/10.5281/zenodo.3831475`, May 2020.

[105] N. E. Glynatsi. Training long short-term memory (LSTM) networks produces successful Prisoner's Dilemma strategies: LSTM networks weights. `https://doi.org/10.5281/zenodo.3831431`, May 2020.

[106] N. E. Glynatsi. Training long short-term memory (LSTM) networks produces successful Prisoner's Dilemma strategies: Training data sets. `https://doi.org/10.5281/zenodo.3841932`, May 2020.

[107] N. E. Glynatsi and V. Knight. Nikoleta-v3/arcas: Arcas v 0.0.4. `https://doi.org/10.5281/zenodo.1127684`, December 2017.

[108] N. E. Glynatsi and V. Knight. Nikoleta-v3/Memory-size-in-the-prisoners-dilemma: initial release. `https://doi.org/10.5281/zenodo.3533146`, November 2019.

[109] N. E. Glynatsi, V. Knight, and T. E. Lee. An evolutionary game theoretic model of rhino horn devaluation. *Ecological Modelling*, 389:33–40, 2018.

[110] N. E. Glynatsi and V. A. Knight. Game theory and python. `https://github.com/Nikoleta-v3/Game-Theory-and-Python`, 2017.

[111] N. E. Glynatsi and V. A. Knight. A bibliometric study of research topics, collaboration and influence in the field of the iterated prisoner's dilemma, 2019.

[112] N. E. Glynatsi and V. A. Knight. A theory of mind: Best responses to memory-one strategies. the limitations of extortion and restricted memory, 2019.

[113] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[114] A. Goodman. Learning implicit communication strategies for the purpose of illicit collusion. *arXiv preprint arXiv:1802.06036*, 2018.

[115] J. Grimmer and B. M. Stewart. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political analysis*, 21(3):267–297, 2013.

[116] N. P. Group. Nature. https://www.nature.com/, 1869.

[117] X. Guan. Gaming and price spikes in electric power markets and possible remedies. In *Proceedings. International Conference on Power System Technology*, volume 1, pages 188–vol. IEEE, 2002.

[118] K. Gurney. Neural networks for perceptual processing: from simulation tools to theories. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1479):339–353, 2007.

[119] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[120] D. Hales. Cooperation without memory or space: Tags, groups and the prisoner's dilemma. In *International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 157–166. Springer, 2000.

[121] M. Harper, V. Knight, M. Jones, G. Koutsovoulos, N. E. Glynatsi, and O. Campbell. Reinforcement learning produces dominant strategies for the iterated prisoner's dilemma. *CoRR*, abs/1707.06307, 2017.

[122] M. Harper, V. Knight, M. Jones, G. Koutsovoulos, N. E. Glynatsi, and O. Campbell. Reinforcement learning produces dominant strategies for the iterated prisoner's dilemma. *PLOS ONE*, 12(12):1–33, 12 2017.

[123] P. G. Harrald and D. B. Fogel. Evolving continuous behaviors in the iterated prisoner's dilemma. *Biosystems*, 37(1):135 – 145, 1996.

[124] M. Harris. Many-core gpu computing with nvidia cuda. In *Proceedings of the 22nd annual international conference on Supercomputing*, pages 1–1, 2008.

[125] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[126] V. Hayes. *The Evolution of Cooperation: A Recreation of Axelrod's Computer Tournament.* The University of North Carolina at Greensboro, 2017.

[127] P. Hennig and C. J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(Jun):1809–1837, 2012.

[128] S. Hettrick. It's impossible to conduct research without software, say 7 out of 10 uk researchers. `https://www.software.ac.uk/blog/2014-12-04-its-impossible-conduct-research-without-software-say-7-out-10-uk-researchers`, 2014.

[129] C. Hilbe, K. Chatterjee, and M. A. Nowak. Partners and rivals in direct reciprocity. *Nature human behaviour*, 2(7):469–477, 2018.

[130] C. Hilbe, L. A. Martinez-Vaquero, K. Chatterjee, and M. A. Nowak. Memory-n strategies of direct reciprocity. *Proceedings of the National Academy of Sciences*, 114(18):4715–4720, 2017.

[131] C. Hilbe, M. A. Nowak, and K. Sigmund. Evolution of extortion in iterated prisoner's dilemma games. *Proceedings of the National Academy of Sciences*, page 201214834, 2013.

[132] C. Hilbe, M. A. Nowak, and A. Traulsen. Adaptive dynamics of extortion and compliance. *PLOS ONE*, 8(11):1–9, 11 2013.

[133] C. Hilbe, A. Traulsen, and K. Sigmund. Partners or rivals? strategies for the iterated prisoner's dilemma. *Games and Economic Behavior*, 92:41 – 52, 2015.

[134] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[135] N. P. C. Hong, T. Crick, I. P. Gent, L. Kotthoff, and K. Takeda. Top tips to make your research irreproducible. *arXiv preprint arXiv:1504.00062*, 2015.

[136] S. Horti. The evolution of trust is a cute explain-o-game about cooperation. `http://ncase.me/trust/`.

[137] E. S. Hou, N. Ansari, and H. Ren. A genetic algorithm for multiprocessor scheduling. *IEEE Transactions on Parallel and Distributed systems*, 5(2):113–120, 1994.

[138] B. I. Hutchins, X. Yuan, J. M. Anderson, and G. M. Santangelo. Relative citation ratio (rcr): A new metric that uses citation rates to measure influence at the article level. *PLoS biology*, 14(9):e1002541, 2016.

[139] G. Ichinose, Y. Tenguishi, and T. Tanizawa. Robustness of cooperation on scale-free networks under continuous topological change. *Physical Review E*, 88(5):052808, 2013.

[140] IEEE. Ieee xplore digital library. http://ieeexplore.ieee.org/Xplore/home.jsp.

[141] A. Inc. Anaconda. `https://www.anaconda.com`.

[142] G. Inc. Github. `https://github.com/`.

[143] G. Inc. Github actions. `https://github.com/marketplace?type=actions`.

[144] M. Inglis and C. Foster. Five decades of mathematics education research. *Journal for Research in Mathematics Education*, 49(4):462–500, 2018.

[145] H. Ishibuchi, H. Ohyanagi, and Y. Nojima. Evolution of strategies with different representation schemes in a spatial iterated prisoner's dilemma game. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(1):67–82, 2011.

[146] L. R. Izquierdo and S. S. Izquierdo. Dynamics of the bush-mosteller learning algorithm in 2x2 games. *Reinforcement Learning: Theory and Applications. I-Tech Education and Publishing*, pages 199–224, 2008.

[147] A. K. Jain, J. Mao, and K. M. Mohiuddin. Artificial neural networks: A tutorial. *Computer*, 29(3):31–44, 1996.

[148] M. Janssen. Evolution of cooperation when feedback to reputation scores is voluntary. *Journal of Artificial Societies and Social Simulation*, 9(1), 2006.

[149] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383, 2001.

[150] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[151] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [misc; accessed ¡today¿].

[152] G. Jones, P. Willett, R. C. Glen, A. R. Leach, and R. Taylor. Development and validation of a genetic algorithm for flexible docking. *Journal of molecular biology*, 267(3):727–748, 1997.

[153] G. Jonsson and S. Vavasis. Accurate solution of polynomial equations using macaulay resultant matrices. *Mathematics of computation*, 74(249):221–262, 2005.

[154] Jupyter. Jupyter notebook. `https://jupyter.org`.

[155] G. Kaiping, S. Cox, and T. Sluckin. Cooperation and punishment in community-structured populations with migration. *Journal of theoretical biology*, 405:116–126, 2016.

[156] G. Kaiping, G. S. Jacobs, S. Cox, and T. Sluckin. Nonequivalence of updating rules in evolutionary games under high mutation rates. *Physical Review E*, 90(4):042726, 2014.

[157] B. Karlik and A. V. Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2011.

[158] A. Kaznatcheev, J. Peacock, D. Basanta, A. Marusyk, and J. G. Scott. Fibroblasts and alectinib switch the evolutionary games that non-small cell lung cancer plays. *bioRxiv*, 2017.

[159] G. Kendall, X. Yao, and S. Y. Chong. *The iterated prisoners' dilemma: 20 years on*, volume 4. World Scientific, 2007.

[160] J. Kepner and J. Gilbert. *Graph algorithms in the language of linear algebra*. SIAM, 2011.

[161] N. Kinch. Game theory and the evolution of trust. `https://medium.com/greater-than-experience-design/game-theory-and-the-evolution-of-trust-6da95b33407a`.

[162] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[163] V. Knight, M. Harper, N. E. Glynatsi, and O. Campbell. Evolution reinforces cooperation with the emergence of self-recognition mechanisms: An empirical

study of strategies in the moran process for the iterated prisoner's dilemma. *PLOS ONE*, 13(10):1–33, 10 2018.

[164] V. A. Knight, O. Campbell, M. Harper, K. M. Langner, J. Campbell, T. Campbell, A. Carney, M. Chorley, C. Davidson-Pilon, K. Glass, et al. An open framework for the reproducible study of the iterated prisoner's dilemma. *Journal of Open Research Software*, 4(1), 2016.

[165] V. A. Knight, M. Harper, N. E. Glynatsi, and O. Campbell. Evolution reinforces cooperation with the emergence of self-recognition mechanisms: an empirical study of the moran process for the iterated prisoner's dilemma. *CoRR*, abs/1707.06920, 2017.

[166] V. A. Knight, M. Harper, N. E. Glynatsi, and J. Gillard. Recognising and evaluating the effectiveness of extortion in the iterated prisoner's dilemma. *CoRR*, abs/1904.00973, 2019.

[167] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. In *Proceedings of the 16th Annual Conference on Theoretical Aspects of Computer Science*, STACS'99, pages 404–413, Berlin, Heidelberg, 1999. Springer-Verlag.

[168] J. R. Koza. Genetic programming. 1997.

[169] D. Kraines and V. Kraines. Pavlov and the prisoner's dilemma. *Theory and decision*, 26(1):47–79, 1989.

[170] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[171] B. Kröse, B. Krose, P. van der Smagt, and P. Smagt. An introduction to neural networks. 1993.

[172] S. Kuhn. Prisoner's dilemma. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2017 edition, 2017.

[173] S. Kyvik and I. Reymert. Research collaboration in groups and networks: differences across academic fields. *Scientometrics*, 113(2):951–967, 2017.

[174] A. Landherr, B. Friedl, and J. Heidemann. A critical review of centrality measures in social networks. *Business & Information Systems Engineering*, 2(6):371–385, Dec 2010.

[175] C. Lee, M. Harper, and D. Fryer. The art of war: Beyond memory-one strategies in population games. *PLOS ONE*, 10(3):1–16, 03 2015.

[176] LessWrong. Zoo of strategies. `http://lesswrong.com/lw/7f2/prisoners_dilemma_tournament_results/`, 2011.

[177] J. Li. How to design a strategy to win an ipd tournament. pages 89–104, 04 2007.

[178] J. Li, P. Hingston, S. Member, and G. Kendall. Engineering Design of Strategies for Winning Iterated Prisoner ' s Dilemma Competitions. 3(4):348–360, 2011.

[179] J. Li and G. Kendall. A strategy with novel evolutionary features for the iterated prisoner's dilemma. *Evolutionary Computation*, 17(2):257–274, 2009.

[180] J. Li, L. Zhu, and M. Gummerum. The relationship between moral judgment and cooperation in children with high-functioning autism. *Scientific Reports*, 4:4314, 2014.

[181] S. Li. Strategies in the stochastic iterated prisoner's dilemma. *REU Papers*, 2014.

[182] Y. Li and W. Ma. Applications of artificial neural networks in financial economics: a survey. In *2010 International symposium on computational intelligence and design*, volume 1, pages 211–214. IEEE, 2010.

[183] P. Liu and H. Xia. Structure and evolution of co-authorship network in an interdisciplinary research field. *Scientometrics*, 103(1):101–134, Apr 2015.

[184] D. R. Lutzker. Sex role, cooperation and competition in a two-person, non-zero sum game. *Journal of Conflict Resolution*, 5(4):366–368, 1961.

[185] R. M. M. M. A. Nowak. Evolutionary games and spatial chaos. *Letters to nature*, 359:826–829, 1992.

[186] F. S. Macaulay. Some formulae in elimination. *Proceedings of the London Mathematical Society*, 1(1):3–27, 1902.

[187] D. Mack, P. N. Auburn, and G. P. Knight. Sex role identification and behavior in a reiterated prisoner's dilemma game. *Psychonomic Science*, 24(6):280–282, Jun 1971.

[188] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal. Long short term memory networks for anomaly detection in time series. In *Proceedings*, volume 89. Presses universitaires de Louvain, 2015.

[189] S. Mallapaty. The brane. `https://thebrane.com`, 2016.

[190] S. Mallapaty. Paper authorship goes hyper. `https://www.natureindex.com/news-blog/paper-authorship-goes-hyper`, 2018.

[191] R. E. Marks and H. Schnabl. *Genetic Algorithms and Neural Networks: A Comparison Based on the Repeated Prisoners Dilemma*, pages 197–219. Springer US, Boston, MA, 1999.

[192] P. Mathieu, B. Beaufils, and J.-P. Delahaye. Studies on dynamics in the classical iterated prisoner's dilemma with few strategies. In *European conference on artificial evolution*, pages 177–190. Springer, 1999.

[193] P. Mathieu and J. P. Delahaye. New winning strategies for the iterated prisoner's dilemma. *Journal of Artificial Societies and Social Simulation*, 20(4):12, 2017.

[194] P. Mathieu and J.-P. Delahaye. Experimental criteria to identify efficient probabilistic memory-one strategies for the iterated prisoner's dilemma. *Simulation Modelling Practice and Theory*, 97:101946, 2019.

[195] Y. Matsumoto, T. Yamagishi, Y. Li, and T. Kiyonari. Prosocial behavior increases with age across five economic games. *PloS one*, 11(7):e0158671, 2016.

[196] J. L. McClelland, D. E. Rumelhart, P. R. Group, et al. Parallel distributed processing. *Explorations in the Microstructure of Cognition*, 2:216–271, 1986.

[197] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[198] G. McKiernan. arxiv. org: the los alamos national laboratory e-print server. *International Journal on Grey Literature*, 1(3):127–138, 2000.

[199] M. Media. Springer publishing. http://www.springer.com/, 1950.

[200] S. Media. Sourceforge. `https://sourceforge.net/`.

[201] S. Mei, Y. Wang, and G. Wen. Automatic fabric defect detection with a multi-scale convolutional denoising autoencoder network model. *Sensors*, 18(4):1064, 2018.

[202] Mercurial-scm. Mercurial. `https://www.mercurial-scm.org/`.

[203] A. Meurer, C. Smith, M. Paprocki, O. Čertík, S. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. Moore, S. Singh, T. Rathnayake, S. Vig, B. Granger, R. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, and A. Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3, 2017.

[204] M. Milinski. Tit for tat in sticklebacks and the evolution of cooperation. *Nature*, 325:433–435, January 1987.

[205] J. H. Miller. The coevolution of automata in the repeated prisoner's dilemma. *Journal of Economic Behavior and Organization*, 29(1):87 – 112, 1996.

[206] J. H. Miller, C. T. Butts, and D. Rode. Communication and cooperation. *Journal of Economic Behavior & Organization*, 47(2):179–195, 2002.

[207] M. Minsky and S. Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 1969.

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

[209] M. Mohanraj, S. Jayaraj, and C. Muraleedharan. Applications of artificial neural networks for thermal analysis of heat exchangers–a review. *International Journal of Thermal Sciences*, 90:150–172, 2015.

[210] P. Molander. The optimal level of generosity in a selfish, uncertain environment. *The Journal of Conflict Resolution*, 29(4):611–618, 1985.

[211] J. A. Molina, J. I. Giménez-Nadal, J. A. Cuesta, C. Gracia-Lazaro, Y. Moreno, and A. Sanchez. Gender differences in cooperation: experimental evidence on high school students. *PloS one*, 8(12):e83700, 2013.

[212] E. F. Moore. Gedanken-experiments on sequential machines. *Automata studies*, 34:129–153, 1956.

[213] J. H. Nachbar. Evolution in the finitely repeated prisoner's dilemma. *Journal of Economic Behavior & Organization*, 19(3):307–326, 1992.

[214] J. Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

[215] J. V. Neumann and O. Morgenstern. Theory of games and economic behavior. *Princeton University Press*, page 625, 1944.

[216] S. Neumann, S. Sood, M. Hollander, F. Wan, A. Ahmed, and M. Hancock. Using bots in strategizing group compositions to improve decision–making processes. In D. D. Schmorrow and C. M. Fidopiastis, editors, *Augmented Cognition: Users and Contexts*, pages 305–325, Cham, 2018. Springer International Publishing.

[217] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *International journal of computer vision*, 79(3):299–318, 2008.

[218] V. A. K. Nikoleta E. Glynatsi and M. Harper. Properties of winning iterated prisoner's dilemma strategies, 2020.

[219] R. V. Noorden. The science that's never been cited. `https://www.nature.com/articles/d41586-017-08404-0`, 2017.

[220] M. Nowak and K. Sigmund. A strategy of win-stay, lose-shift that outperforms tit-for-tat in the prisoner's dilemma game. *Nature*, 364(6432):56–58, 1993.

[221] M. A. Nowak and K. Sigmund. Game-dynamical aspects of the prisoner's dilemma. *Applied Mathematics and Computation*, 30(3):191 – 213, 1989.

[222] M. A. Nowak and K. Sigmund. The evolution of stochastic strategies in the prisoner's dilemma. *Acta Applicandae Mathematica*, 20(3):247–265, Sep 1990.

[223] M. A. Nowak and K. Sigmund. Tit for tat in heterogeneous populations. *Nature*, 355:250–253, January 1992.

[224] M. A. Nowak and K. Sigmund. The dynamics of indirect reciprocity. *Journal of theoretical Biology*, 194(4):561–574, 1998.

[225] N. Nurseitov, M. Paulson, R. Reynolds, and C. Izurieta. Comparison of json and xml data interchange formats: a case study. *Caine*, 2009:157–162, 2009.

[226] U. of Southampton. University of southampton team wins prisoner's dilemma competition. `https://www.southampton.ac.uk/news/2004/10/team-wins-competition.page`, 2004.

[227] H. Ohtsuki. Evolutionary dynamics of coordinated cooperation. *Frontiers in Ecology and Evolution*, 6:62, 2018.

[228] H. Ohtsuki, C. Hauert, E. Lieberman, and M. A. Nowak. A simple rule for the evolution of cooperation on graphs and social networks. *Nature*, 441(7092):502, 2006.

[229] D. A. O'Neil and M. D. Petty. Synthesizing social networks with iterated prisoners' dilemma. In *Proceedings of the International Conference on Modeling, Simulation and Visualization Methods (MSV)*, pages 3–10. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2019.

[230] R. J. Ormerod. Or as rational choice: A decision and game theory perspective. *Journal of the Operational Research Society*, 61(12):1761–1776, 2010.

[231] E. Osaba, R. Carballedo, F. Diaz, E. Onieva, P. Lopez, and A. Perallos. On the influence of using initialization functions on genetic algorithms solving combinatorial optimization problems: a first study on the tsp. pages 1–6, 2014.

[232] M. J. Osborne et al. *An introduction to game theory*, volume 3. Oxford university press New York, 2004.

[233] L. Pan, D. Hao, Z. Rong, and T. Zhou. Zero-determinant strategies in iterated public goods game. *Scientific reports*, 5:13096, 2015.

[234] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[235] H. Percival. *Test-driven development with Python: obey the testing goat: using Django, Selenium, and JavaScript.* " O'Reilly Media, Inc.", 2014.

[236] prase. Prisoner's dilemma tournament results. `https://www.lesswrong.com/posts/hamma4XgeNrsvAJv5/prisoner-s-dilemma-tournament-results`, 2011.

[237] W. H. Press and F. G. Dyson. Iterated prisoner's dilemma contains strategies that dominate any evolutionary opponent. *Proceedings of the National Academy of Sciences*, 109(26):10409–10413, 2012.

[238] A. Pritchard et al. Statistical bibliography or bibliometrics. *Journal of documentation*, 25(4):348–349, 1969.

[239] P. Pudaite. On the initial viability of cooperation. *Merriam Laboratory for Analytic ARTICLES 1389 Political Research*, 1985.

[240] D. Raina and B. M. Gupta. Four aspects of the institutionalization of physics research in india (1990–1950): Substantiating the claims of historical sociology through bibliometrics. *Scientometrics*, 42(1):17–40, 1998.

[241] R. Raina, A. Madhavan, and A. Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880, 2009.

[242] A. Rapoport and A. M. Chammah. *Prisoner's Dilemma: A Study in Conflict and Cooperation, by Anatol Rapoport and Albert M. Chammah, with the Collaboration of Carol J. Orwant*. University of Michigan Press, 1965.

[243] A. Rapoport, D. A. Seale, and A. M. Colman. Is tit-for-tat the answer? on the conclusions drawn from axelrod's tournaments. *PLOS ONE*, 10(7):1–11, 07 2015.

[244] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[245] D. Resultant. The kapur-saxena-yang variant of the.

[246] R. L. Riolo, M. D. Cohen, and R. Axelrod. Evolution of cooperation without reciprocity. *Nature*, 414(6862):441, 2001.

[247] A. J. Robson. Efficiency in evolutionary games: Darwin, nash and the secret handshake. *Journal of theoretical Biology*, 144(3):379–396, 1990.

[248] M. Röder, A. Both, and A. Hinneburg. Exploring the space of topic coherence measures. In *Proceedings of the eighth ACM international conference on Web search and data mining*, pages 399–408. ACM, 2015.

[249] L. Roemheld. Evolutionary extortion and mischief: Zero determinant strategies in iterated 2x2 games. *arXiv preprint arXiv:1308.2576*, 2013.

[250] A. Rogers, R. Dash, S. Ramchurn, P. Vytelingum, and N. Jennings. Coordinating team players within a noisy iterated prisoner's dilemma tournament. *Theoretical computer science.*, 377(1-3):243–259, 2007.

[251] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[252] P. J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[253] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[254] N. B. Ruparelia. The history of version control. *ACM SIGSOFT Software Engineering Notes*, 35(1):5–9, 2010.

[255] H. Sak, A. Senior, and F. Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[256] G. Salmon. *Lessons introductory to the modern higher algebra*. Stechert, 1924.

[257] T. W. Sandholm and R. H. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37(1-2):147–166, 1996.

[258] G. K. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig. Ten simple rules for reproducible computational research. *PLoS computational biology*, 9(10), 2013.

[259] L. A. Santorelli, C. R. Thompson, E. Villegas, J. Svetz, C. Dinh, A. Parikh, R. Sucgang, A. Kuspa, J. E. Strassmann, D. C. Queller, et al. Facultative cheater mutants reveal the genetic complexity of cooperation in social amoebae. *Nature*, 451(7182):1107, 2008.

[260] J. S. Sartakhti, M. H. Manshaei, D. Basanta, and M. Sadeghi. Evolutionary emergence of angiogenesis in avascular tumors using a spatial public goods game. *PloS one*, 12(4):e0175063, 2017.

[261] V. Sekara, P. Deville, S. E. Ahnert, A. Barabási, R. Sinatra, and S. Lehmann. The chaperone effect in scientific publishing. *Proceedings of the National Academy of Sciences*, 115(50):12603–12607, 2018.

[262] R. Selten and P. Hammerstein. Gaps in harley's argument on evolutionarily stable learning rules and in the logic of "tit for tat". *Behavioral and Brain Sciences*, 7(1):115–116, 1984.

[263] J. Sensenig, T. E. Reed, and J. S. Miller. Cooperation in the prisoner's dilemma as a function of interpersonal distance. *Psychonomic Science*, 26(2):105–106, Feb 1972.

[264] N. Shahid, T. Rappon, and W. Berta. Applications of artificial neural networks in health care organizational decision-making: A scoping review. *PloS one*, 14(2), 2019.

[265] A. Shameem and R. Jason. Multi-core programming-increasing performance through software multithreading. *Intel, Hillsboro, OR*, 2005.

[266] K. Sigmund. *The calculus of selfishness*, volume 6. Princeton University Press, 2010.

[267] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[268] M. Sistrom, D. Park, H. E. O'Brien, Z. Wang, D. S. Guttman, J. P. Townsend, and P. E. Turner. Genomic and gene-expression comparisons among phage-resistant type-iv pilus mutants of pseudomonas syringae pathovar phaseolicola. *PloS one*, 10(12):e0144514, 2015.

[269] J. M. Smith. The theory of games and the evolution of animal conflicts. *Journal of Theoretical Biology*, 47(1):209 – 221, 1974.

[270] J. M. Smith. Game theory and the evolution of behaviour. *Proceedings of the Royal Society of London. Series B, Biological Sciences*, 205(1161):475–488, 1979.

[271] J. M. Smith and G. R. Price. The logic of animal conflict. *Nature*, 246(5427):15–18, 1973.

[272] S. Soffer, A. Ben-Cohen, O. Shimon, M. M. Amitai, H. Greenspan, and E. Klang. Convolutional neural networks for radiologic images: a radiologist's guide. *Radiology*, 290(3):590–606, 2019.

[273] D. Steinkraus, I. Buck, and P. Simard. Using gpus for machine learning algorithms. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 1115–1120. IEEE, 2005.

[274] A. J. Stewart and J. B. Plotkin. Extortion and cooperation in the prisoner's dilemma. *Proceedings of the National Academy of Sciences*, 109(26):10134–10135, 2012.

[275] A. J. Stewart and J. B. Plotkin. From extortion to generosity, evolution in the iterated prisoner's dilemma. *Proceedings of the National Academy of Sciences*, 110(38):15348–15353, 2013.

[276] A. J. Stewart and J. B. Plotkin. Small groups and long memories promote cooperation. *Scientific reports*, 6:26889, 2016.

[277] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[278] C. R. Sugimoto, D. Li, T. G. Russell, S. C. Finlay, and Y. Ding. The shifting sands of disciplinary development: Analyzing north american library and information science dissertations using latent dirichlet allocation. *Journal of the American Society for Information Science and Technology*, 62(1):185–204, 2011.

[279] S. Suzuki and E. Akiyama. Reputation and the evolution of cooperation in sizable groups. *Proceedings of the Royal Society of London B: Biological Sciences*, 272(1570):1373–1377, 2005.

[280] I. Tabiai. Exploring the evolution of open access publications with arcas. `http://iltabiai.github.io`, 2019.

[281] J. T. Tedeschi, D. S. Hiester, S. Lesnick, and J. P. Gahagan. Start effect and response bias in the prisoner's dilemma game. *Psychonomic Science*, 11(4):149–150, 1968.

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

[283] P. E. Turner and L. Chao. Prisoner's dilemma in an rna virus. *Nature*, 398(6726):441, 1999.

[284] E. Tzafestas. Toward adaptive cooperative behavior. 2:334–340, Sep 2000.

[285] E. Tzafestas. Toward adaptive cooperative behavior. *From Animals to animals: Proceedings of the 6th International Conference on the Simulation of Adaptive Behavior (SAB-2000)*, 2:334–340, 2000.

[286] P. Van-Den-Berg and F. J. Weissing. The importance of mechanisms for the evolution of cooperation. In *Proc. R. Soc. B*, volume 282, page 20151382. The Royal Society, 2015.

[287] I. Vasilev, D. Slater, G. Spacagna, P. Roelants, and V. Zocca. *Python Deep Learning: Exploring deep learning techniques and neural network architectures with Pytorch, Keras, and TensorFlow*. Packt Publishing Ltd, 2019.

[288] J. Wang, C. Xia, Y. Wang, S. Ding, and J. Sun. Spatial prisoner's dilemma games with increasing size of the interaction neighborhood on regular lattices. *Chinese science bulletin*, 57(7):724–728, 2012.

[289] K. Wang. Iterated prisoners dilemma with reinforcement learning. *Course Project. Stanford University. url: http://web. stanford. edu/class/psych209/Readings/2017ProjectExamples/wangkeven_17581_1628229_psych209_paper. pdf*, 2017.

[290] L. Wang, X. Duan, Q. Zhang, Z. Niu, G. Hua, and N. Zheng. Segment-tube: Spatio-temporal action localization in untrimmed videos with per-frame segmentation. *Sensors*, 18(5):1657, 2018.

[291] X. Wang, X. Ma, and W. E. L. Grimson. Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models. *IEEE Transactions on pattern analysis and machine intelligence*, 31(3):539–555, 2008.

[292] P. Werbos. Beyond regression: New tools for prediction and analysis in the behavioral sciences. ph. d. thesis, harvard university, cambridge, ma, 1974. 1974.

[293] D. Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

[294] G. Wilson, D. A. Aruliah, C. T. Brown, N. P. C. Hong, M. Davis, R. T. Guy, S. H. Haddock, K. D. Huff, I. M. Mitchell, M. D. Plumbley, et al. Best practices for scientific computing. *PLoS biology*, 12(1), 2014.

[295] J. Wu and R. Axelrod. How to cope with noise in the iterated prisoner's dilemma. *Journal of Conflict Resolution*, 39(1):183–189, 1995.

[296] T. Wu, F. Fu, and L. Wang. Phenotype affinity mediated interactions can facilitate the evolution of cooperation. *Journal of theoretical biology*, 462:361–369, 2019.

[297] B. J. Wythoff. Backpropagation neural networks: a tutorial. *Chemometrics and Intelligent Laboratory Systems*, 18(2):115–155, 1993.

[298] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. In *Feature extraction, construction and selection*, pages 117–136. Springer, 1998.

[299] M. Youngblood and D. Lahti. A bibliometric analysis of the interdisciplinary field of cultural evolution. *Palgrave Communications*, 4(1):120, 2018.

[300] C. Zhang, J. Zhang, G. Xie, L. Wang, and M. Perc. Evolution of interactions and cooperation in the spatial prisoner's dilemma game. *PLOS ONE*, 6(10):1–7, 10 2011.

Appendix A

Centrality measures distributions

A.1 Distributions for G and $\bar{G}$

Betweeness and closeness centralities distributions for G and $\bar{G}$.

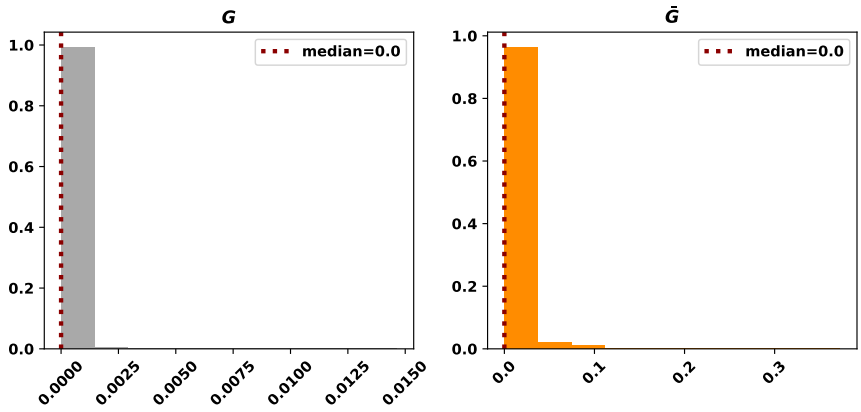


Figure A.1: Distributions of betweenness centrality in G and $\bar{G}$

A.2 Distributions for topic networks

Betweeness and closeness centralities distributions for graphs of topics A to E.

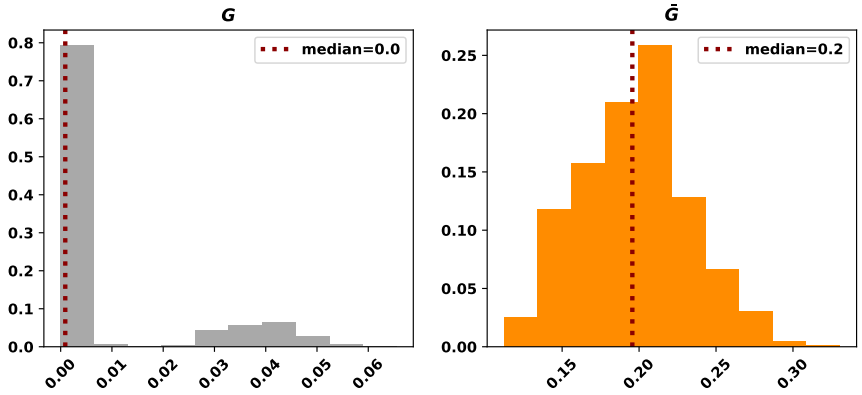


Figure A.2: Distributions of closeness centrality in G and $\bar{G}$

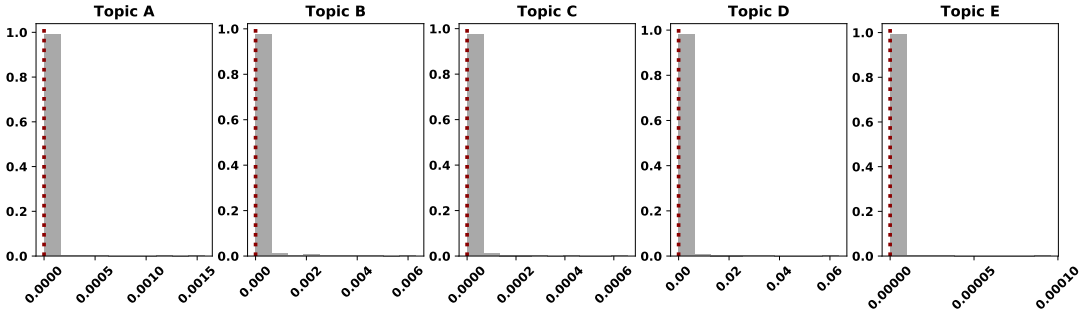


Figure A.3: Distributions of betweenness centrality in topics' networks.

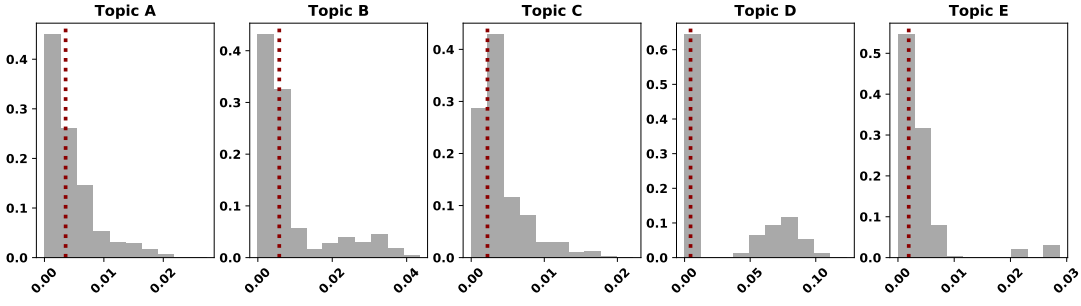


Figure A.4: Distributions of closeness centrality in topics' networks.

Appendix B

List of strategies

B.1 List of strategies considered in Chapter 4

The strategies considered in Chapter 4, which are from APL version 3.0.0.

1. ϕ [7]
2. π [7]
3. e [7]
4. ALLCorALLD [7]
5. Adaptive [178]
6. Adaptive Pavlov 2006 [159]
7. Adaptive Pavlov 2011 [178]
8. Adaptive Tit For Tat: 0.5 [285]
9. Aggravater [7]
10. Alexei [236]
11. Alternator [34, 208]
12. Alternator Hunter [7]
13. Anti Tit For Tat [132]
14. AntiCycler [7]
15. Appeaser [7]
16. Arrogant QLearner [7]
17. Average Copier [7]
18. Backstabber [7]
19. Better and Better [4]
20. Bully [213]
21. Calculator [4]
22. Cautious QLearner [7]
23. Champion [33]
24. CollectiveStrategy [179]
25. Contrite Tit For Tat [295]
26. Cooperator [34, 208, 237]
27. Cooperator Hunter [7]
28. Cycle Hunter [7]
29. Cycler CCCCCD [7]
30. Cycler CCCD [7]
31. Cycler CCCDCD [7]
32. Cycler CCD [208]
33. Cycler DC [7]
34. Cycler DDC [208]
35. DBS [31]
36. Davis [32]
37. Defector [34, 208, 237]
38. Defector Hunter [7]

39. Double Crosser [7]
40. Desperate [286]
41. DoubleResurrection [8]
42. Doubler [4]
43. Dynamic Two Tits For Tat [7]
44. EasyGo [178, 4]
45. Eatherley [33]
46. Eventual Cycle Hunter [7]
47. Evolved ANN [7]
48. Evolved ANN 5 [7]
49. Evolved ANN 5 Noise 05 [7]
50. Evolved FSM 16 [7]
51. Evolved FSM 16 Noise 05 [7]
52. Evolved FSM 4 [7]
53. Evolved HMM 5 [7]
54. EvolvedLookerUp1 1 1 [7]
55. EvolvedLookerUp2 2 2 [7]
56. Eugine Nier [236]
57. Feld [32]
58. Firm But Fair [87]
59. Fool Me Forever [7]
60. Fool Me Once [7]
61. Forgetful Fool Me Once [7]
62. Forgetful Grudger [7]
63. Forgiver [7]
64. Forgiving Tit For Tat [7]
65. Fortress3 [27]
66. Fortress4 [27]
67. GTFT [91, 220]
68. General Soft Grudger [7]
69. Gradual [43]
70. Gradual Killer [4]
71. Grofman[32]
72. Grudger [32, 41, 43, 286, 178]
73. GrudgerAlternator [4]
74. Grumpy [7]
75. Handshake [247]
76. Hard Go By Majority [208]
77. Hard Go By Majority: 10 [7]
78. Hard Go By Majority: 20 [7]
79. Hard Go By Majority: 40 [7]
80. Hard Go By Majority: 5 [7]
81. Hard Prober [4]
82. Hard Tit For 2 Tats [274]
83. Hard Tit For Tat [6]
84. Hesitant QLearner[7]
85. Hopeless [286]
86. Inverse [7]
87. Inverse Punisher [7]
88. Joss [32, 274]
89. Knowledgeable Worse and Worse [7]
90. Level Punisher [8]
91. Limited Retaliate 2 [7]
92. Limited Retaliate 3 [7]
93. Limited Retaliate [7]
94. MEM2 [181]
95. Math Constant Hunter [7]
96. Meta Hunter Aggressive [7]
97. Meta Hunter [7]
98. Meta Majority [7]
99. Meta Majority Finite Memory [7]

100. Meta Majority Long Memory [7]
101. Meta Majority Memory One [7]
102. Meta Minority [7]
103. Meta Mixer [7]
104. Meta Winner [7]
105. Meta Winner Deterministic [7]
106. Meta Winner Ensemble [7]
107. Meta Winner Finite Memory [7]
108. Meta Winner Long Memory [7]
109. Meta Winner Memory One [7]
110. Meta Winner Stochastic [7]
111. NMWE Deterministic [7]
112. NMWE Finite Memory [7]
113. NMWE Long Memory [7]
114. NMWE Memory One [7]
115. NMWE Stochastic [7]
116. Naive Prober [178]
117. Negation [6]
118. Nice Average Copier [7]
119. Nice Meta Winner [7]
120. Nice Meta Winner Ensemble [7]
121. Nydegger [32]
122. Omega TFT [159]
123. Once Bitten [7]
124. Opposite Grudger [7]
125. PSO Gambler 1 1 1 [7]
126. PSO Gambler 2 2 2 [7]
127. PSO Gambler 2 2 2 Noise 05 [7]
128. PSO Gambler Mem1 [7]
129. Predator [27]
130. Prober [178]
131. Prober 2 [4]
132. Prober 3 [4]
133. Prober 4 [4]
134. Pun1 [27]
135. Punisher [7]
136. Raider [29]
137. Random Hunter [7]
138. Random: 0.5 [32, 285]
139. Remorseful Prober [178]
140. Resurrection [8]
141. Retaliate 2 [7]
142. Retaliate 3 [7]
143. Retaliate [7]
144. Revised Downing [32]
145. Ripoff [23]
146. Risky QLearner [7]
147. SelfSteem [56]
148. ShortMem [56]
149. Shubik [32]
150. Slow Tit For Two Tats [7]
151. Slow Tit For Two Tats 2 [4]
152. Sneaky Tit For Tat [7]
153. Soft Go By Majority [34, 208]
154. Soft Go By Majority 10 [7]
155. Soft Go By Majority 20 [7]
156. Soft Go By Majority 40 [7]

157. Soft Go By Majority 5 [7]
158. Soft Grudger [178]
159. Soft Joss [4]
160. SolutionB1 [21]
161. SolutionB5 [21]
162. Spiteful Tit For Tat [4]
163. Stalker [56]
164. Stein and Rapoport [32]
165. Stochastic Cooperator [13]
166. Stochastic WSLS [7]
167. Suspicious Tit For Tat [43, 132]
168. TF1 [7]
169. TF2 [7]
170. TF3 [7]
171. Tester [33]
172. ThueMorse [7]
173. ThueMorseInverse [7]
174. Thumper [23]
175. Tit For 2 Tats [34]
176. Tit For Tat [32]
177. Tricky Cooperator [7]
178. Tricky Defector [7]
179. Tullock [32]
180. Two Tits For Tat [34]
181. VeryBad [56]
182. Willing [286]
183. Win-Shift Lose-Stay [178]
184. Win-Stay Lose-Shift [169, 220, 274]
185. Winner12 [193]
186. Winner21 [193]
187. Worse and Worse[4]
188. Worse and Worse 2[4]
189. Worse and Worse 3[4]
190. ZD-Extort-2 v2 [172]
191. ZD-Extort-2 [274]
192. ZD-Extort-4 [7]
193. ZD-GEN-2 [172]
194. ZD-GTFT-2 [274]
195. ZD-SET-2 [172]

B.2 List of strategies considered in Chapter 6

The strategies considered in Chapter 6, which are from APL version 4.2.0.

1. e [7]
2. ALLCorALLD [7]
3. AON2 [130]
4. Adaptive [178]
5. Adaptive Pavlov 2006 [159]
6. Adaptive Pavlov 2011 [178]
7. Adaptive Tit For Tat [285]
8. Aggravater [7]
9. Alexei [236]
10. Alternator [34, 208]
11. Alternator Hunter [7]
12. Anti Tit For Tat [132]
13. AntiCycler [7]
14. Appeaser [7]
15. Arrogant QLearner [7]
16. Average Copier [7]
17. Backstabber [7]
18. Better and Better [4]
19. Black [33]
20. Borufsen [33]
21. Bully [213]

22. Bush Mosteller [146]
23. Calculator [4]
24. Cautious QLearner [7]
25. Cave [33]
26. Champion [33]
27. Colbert [33]
28. CollectiveStrategy [179]
29. Contrite Tit For Tat [295]
30. Cooperator [34, 208, 237]
31. Cooperator Hunter [7]
32. Cycle Hunter [7]
33. Cycler CCCCCD [7]
34. Cycler CCCD [7]
35. Cycler CCCDCD [7]
36. Cycler CCD [208]
37. Cycler DC [7]
38. Cycler DDC [208]
39. Davis [32]
40. Defector [34, 208, 237]
41. Defector Hunter [7]
42. Delayed AON1 [130]
43. Double Crosser [7]
44. Desperate [286]
45. DoubleResurrection [8]
46. Doubler [4]
47. Dynamic Two Tits For Tat [7]
48. EasyGo [178, 4]
49. Eatherley [33]
50. Eventual Cycle Hunter [7]
51. Evolved ANN [7]
52. Evolved ANN 5 [7]
53. Evolved ANN 5 Noise 05 [7]
54. Evolved FSM 16 [7]
55. Evolved FSM 16 Noise 05 [7]
56. Evolved FSM 4 [7]
57. Evolved HMM 5 [7]
58. EvolvedLookerUp1 1 1 [7]
59. EvolvedLookerUp2 2 2 [7]
60. Eugine Nier [236]
61. Feld [32]
62. Firm But Fair [87]
63. Fool Me Forever [7]
64. Fool Me Once [7]
65. Forgetful Fool Me Once [7]
66. Forgetful Grudger [7]
67. Forgiver [7]
68. Forgiving Tit For Tat [7]
69. Fortress3 [27]
70. Fortress4 [27]
71. GTFT [91, 220]
72. General Soft Grudger [7]
73. Getzler [33]
74. Gladstein [33]
75. GraaskampKatzen [33]
76. Gradual [43]
77. Gradual Killer [4]
78. Grofman[32]
79. Grudger [32, 41, 43, 286, 178]
80. GrudgerAlternator [4]
81. Grumpy [7]
82. Handshake [247]
83. Hard Prober [4]
84. Hard Tit For 2 Tats [274]
85. Hard Tit For Tat [6]
86. Harrington [33]
87. Hesitant QLearner[7]
88. Hopeless [286]

89. Inverse [7]
90. Inverse Punisher [7]
91. Joss [32, 274]
92. Kluepfel [33]
93. Knowledgeable Worse and Worse [7]
94. Level Punisher [8]
95. Leyvraz [33]
96. Limited Retaliate 2 [7]
97. Limited Retaliate 3 [7]
98. Limited Retaliate [7]
99. MEM2 [181]
100. Math Constant Hunter [7]
101. Meta Hunter Aggressive [7]
102. Meta Hunter [7]
103. Michaelos [176]
104. Mikkelson [33]
105. More Tideman and Chieruzzi [33]
106. Grofman [33]
107. N Tit(s) For M Tat(s) [7]
108. Naive Prober [178]
109. Negation [6]
110. Nice Average Copier [7]
111. Nydegger [32]
112. Omega TFT [159]
113. Once Bitten [7]
114. Opposite Grudger [7]
115. PSO Gambler 1 1 1 [7]
116. PSO Gambler 2 2 2 [7]
117. PSO Gambler 2 2 2 Noise 05 [7]
118. PSO Gambler Mem1 [7]
119. Predator [27]
120. Prober [178]
121. Prober 2 [4]
122. Prober 3 [4]
123. Prober 4 [4]
124. Pun1 [27]
125. Punisher [7]
126. Raider [29]
127. Random Hunter [7]
128. Random Tit for Tat [7]
129. Random: 0.5 [32, 285]
130. Remorseful Prober [178]
131. Resurrection [8]
132. Retaliate 2 [7]
133. Retaliate 3 [7]
134. Retaliate [7]
135. Revised Downing [32]
136. Richard Hufford [33]
137. Ripoff [23]
138. Risky QLearner [7]
139. SelfSteem [56]
140. ShortMem [56]
141. Shubik [32]
142. Slow Tit For Two Tats 2 [4]
143. Sneaky Tit For Tat [7]
144. Soft Grudger [178]
145. Soft Joss [4]
146. SolutionB1 [21]
147. SolutionB5 [21]
148. Spiteful Tit For Tat [4]
149. Stalker [56]
150. Stein and Rapoport [32]
151. Stochastic Cooperator [13]
152. Stochastic WSLS [7]

153. Suspicious Tit For Tat [43, 132]
154. TF1 [7]
155. TF2 [7]
156. TF3 [7]
157. Tester [33]
158. ThueMorse [7]
159. ThueMorseInverse [7]
160. Thumper [23]
161. Tideman and Chieruzzi
162. Tit For 2 Tats [34]
163. Tit For Tat [32]
164. Tranquilizer [32]
165. Tricky Cooperator [7]
166. Tricky Defector [7]
167. Tricky Level Punisher
168. Tullock [32]
169. Two Tits For Tat [34]
170. VeryBad [56]
171. Weiner [33]
172. White [33]
173. Willing [286]
174. Win-Shift Lose-Stay [178]
175. Win-Stay Lose-Shift [169, 220, 274]
176. Winner12 [193]
177. Winner21 [193]
178. WmAdams [33]
179. Worse and Worse[4]
180. Worse and Worse 2[4]
181. Worse and Worse 3[4]
182. Yamachi [33]
183. ZD-Extort-2 v2 [172]
184. ZD-Extort-2 [274]
185. ZD-Extort-4 [7]
186. ZD-Extort3 [7]
187. ZD-Extortion [249]
188. ZD-GEN-2 [172]
189. ZD-GTFT-2 [274]
190. ZD-Mem2 [7]
191. ZD-Mischief [249]
192. ZD-SET-2 [172]

B.3 List of strategies considered in Chapter 7

The strategies considered in Chapter 7 for the training subsets.

The top 18 performing strategies of the 218 opponents standard tournament [104]:

1. PSOGambler Mem 1
2. Evolved ANN 5
3. Double Crosser
4. OmegaTFT
5. EvolvedLookerUp 2 2 2
6. Fool Me Once
7. Gradual
8. PSOGambler 2 2 2 Noise 05
9. Evolved HMM 5
10. PSOGambler 1 1 1
11. PSOGambler 2 2 2
12. Evolved ANN
13. Evolved FSM 16
14. Winner 12
15. Back Stabber
16. Evolved FSM 16 Noise 05
17. Evolved ANN Noise 05
18. Evolved FSM 4

The 15 representative strategies whose ranks are across the 218 ranks of the standard tournament:

1. Eatherley
2. Cautious Q Learner
3. Forgiver
4. Gladstein
5. Punisher
6. Easy Go
7. Tricky Cooperator
8. Anti Tit For Tat
9. Tit For Tat
10. PSO Gambler 2 2 2 Noise 05
11. Evolved HMM 5
12. e
13. Pun1
14. A Pavlov 2006
15. GraaskampKatzen
16. Hard Tit For Tat

The 11 strategies which are classified as a set of basic strategies in the APL:

1. Alternator
2. AntiTitForTat
3. Bully
4. Cooperator
5. Cycler DC
6. Defector
7. Grudger
8. Suspicious Tit For Tat
9. Tit For Tat
10. Win Shift Lose Stay
11. Win Stay Lose Shift

Appendix C

Further analysis on features importance

C.1 Correlation coefficients of strategies features

A graphical representation of the correlation coefficients for the features of Table 4.5, Chapter 4.

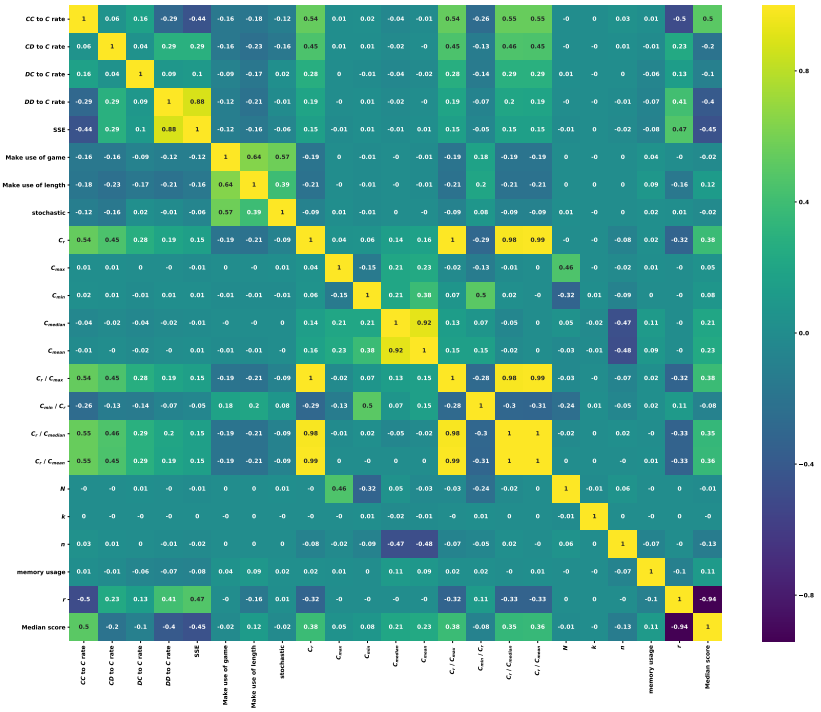


Figure C.1: Correlation coefficients of features in Table 4.5 for standard tournaments

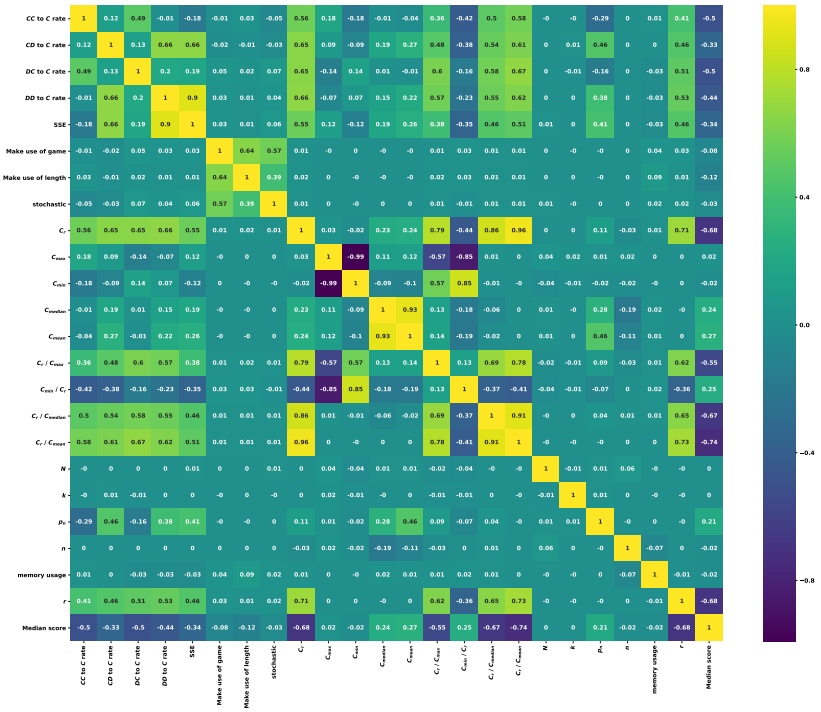


Figure C.2: Correlation coefficients of features in Table 4.5 for noisy tournaments

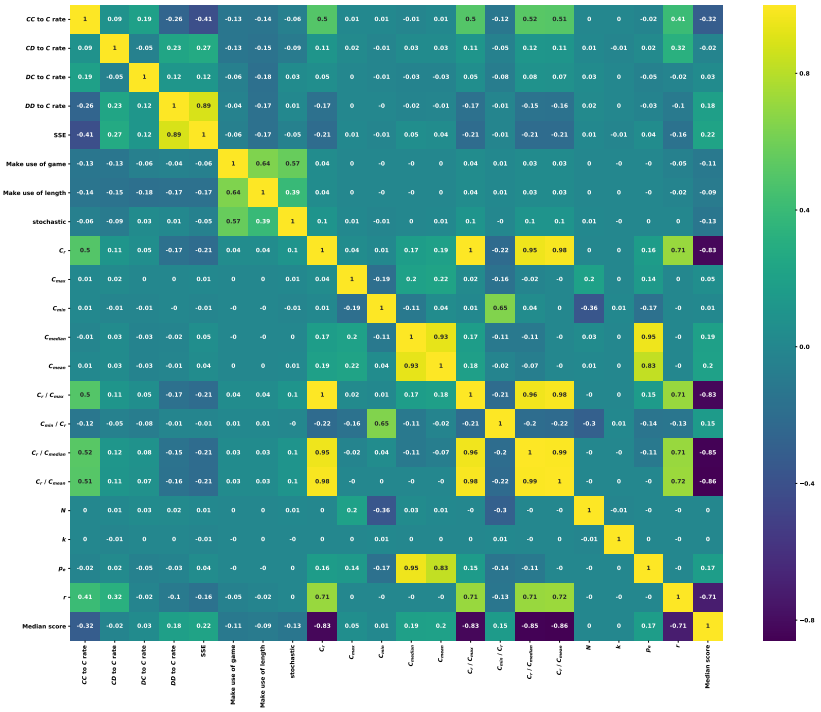


Figure C.3: Correlation coefficients of features in Table 4.5 for probabilistic ending tournaments

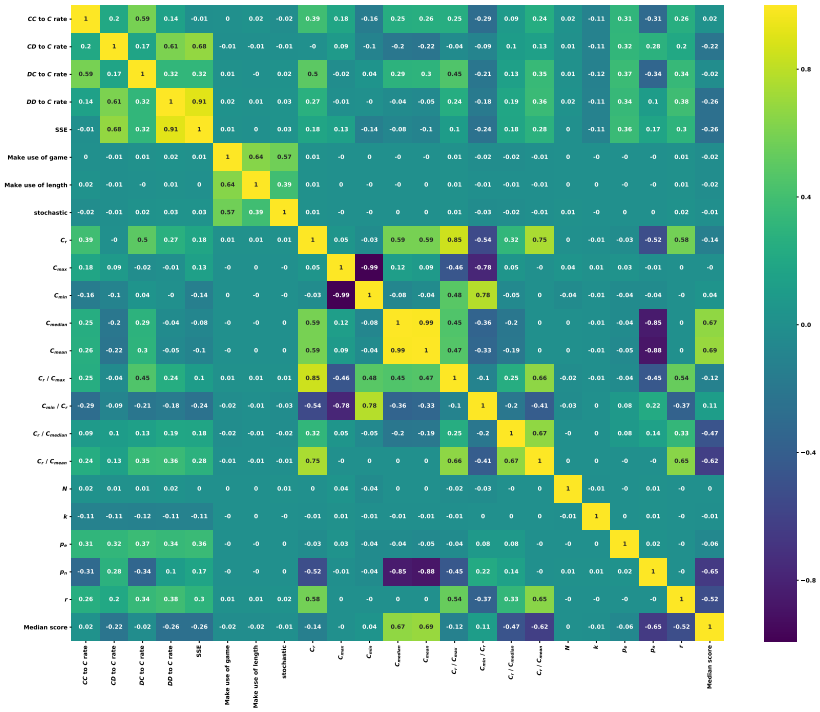


Figure C.4: Correlation coefficients of features in Table 4.5 for noisy probabilistic ending tournaments

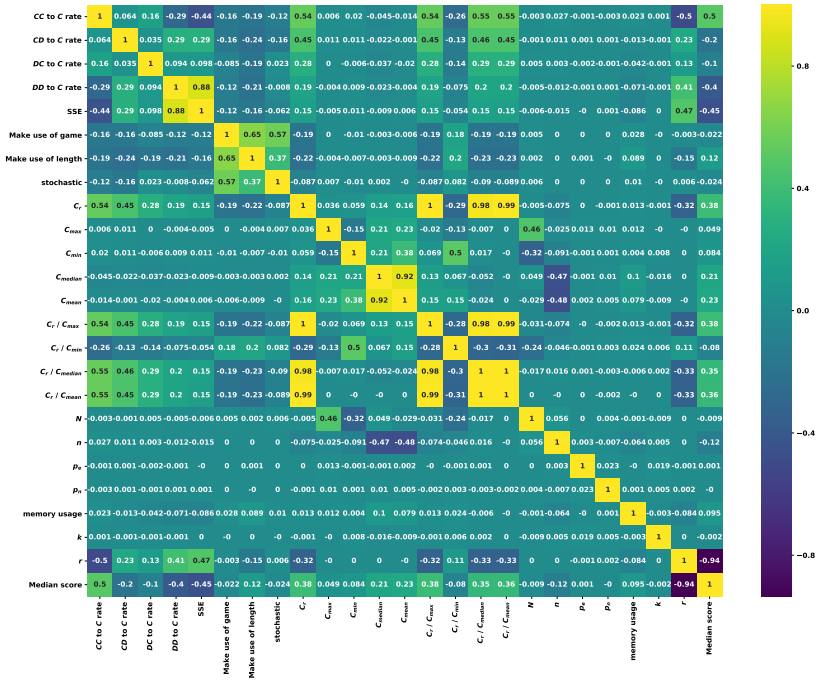


Figure C.5: Correlation coefficients of features in Table 4.5 for data set

C.2 Multivariate linear regressions on median score

A multivariate linear regression has also been fitted to model the relationship between the features and the median score. The features included are given by Table C.1 alongside their corresponding p values in the distinct tournaments and their regression coefficients.

	Standard		Noisy		Probabilistic ending		Noisy probabilistic ending		Overall	
	R adjusted: 0.576		R adjusted: 0.679		R adjusted: 0.816		R adjusted: 0.930		R adjusted: 0.318	
	Coefficient	p-value	Coefficient	p-value	Coefficient	p-value	Coefficient	p-value	Coefficient	p-value
CC to C rate	0.043	0.000	-0.380	0.000	0.224	0.000	0.078	0.0	0.308	0.0
CD to C rate	-0.325	0.000	0.124	0.000	0.060	0.000	0.073	0.0	-0.014	0.0
C_r / C_{max}	-	-	-1.044	0.000	-	-	-1.251	0.0	-	-
C_r / C_{mean}	0.553	0.000	-0.101	0.000	-1.136	0.000	-0.089	0.0	-0.665	0.0
C_{max}	0.059	0.000	-	-	-0.044	0.086	-1.396	0.0	-	-
C_{mean}	1.837	0.000	3.078	0.000	1.506	0.000	3.645	0.0	-	-
C_{min}	0.156	0.000	1.528	0.000	0.311	0.000	-	-	-	-
C_{min} / C_r	-0.049	0.000	-0.378	0.000	-0.204	0.000	-	-	-0.257	0.0
DC to C rate	-0.204	0.000	0.074	0.000	0.066	0.000	0.066	0.0	0.007	0.0
k	-0.000	0.853	-0.000	0.987	0.000	0.008	0.000	0.0	-	-
n	-0.000	0.000	-	-	-	-	-	-	-	-
p_e	-	-	-	-	0.025	0.000	-0.095	0.0	-	-
p_n	-	-	0.124	0.000	-	-	-	-	-	-
SSE	-0.294	0.000	-0.319	0.000	0.055	0.000	0.010	0.0	-0.015	0.0
constant	0.925	0.000	1.536	0.000	2.466	0.000	2.299	0.0	2.924	0.0
memory usage	0.010	0.000	-0.004	0.000	-	-	-	-	-	-

Table C.1: Results of multivariate linear regressions with the median score as the dependent variable. R squared is reported for each model.

C.3 Evaluation based on clustering and random forest.

The final method to evaluate the features importance in a strategy's success is a combination of a clustering task and a random forest algorithm. Initially the performances are clustered into different clusters based on them being successful or not. The performances are clustered into successful and unsuccessful clusters based on 4 different approaches. More specifically:

- **Approach 1:** The performances are divided into two clusters based on whether their performance was in the top 5% of their respective tournaments. Thus, whether r was smaller or larger than 0.05.
- **Approach 2:** The performances are divided into two clusters based on whether their performance was in the top 25% of their respective tournaments. Thus,

whether r was smaller or larger than 0.25.

- **Approach 3:** The performances are divided into two clusters based on whether their performance was in the top 50% of their respective tournaments. Thus, whether r was smaller or larger than 0.50.

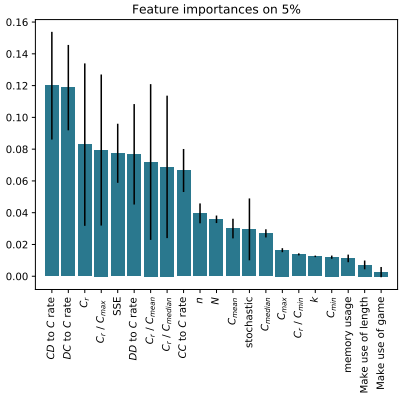
- **Approach 4:** The performances are clustered based on their normalised rank and their median score by a $k-$means algorithm [20]. The number of clusters is not deterministically chosen but it is based on the silhouette coefficients [252].

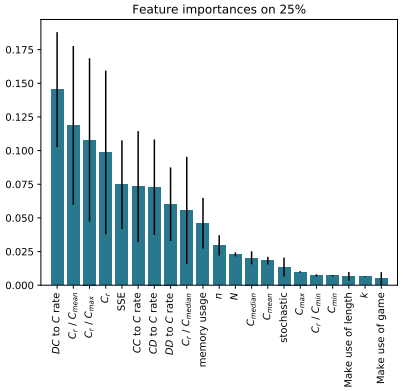
Once the performances have been assigned to a cluster for each approach a random forest algorithm [55] is applied. The problem is a supervised problem where the random forest algorithm predicts the cluster to which a performance has been assigned to using the features of Table 4.5. The random forest models are trained on a training set of 70% of the tournaments results. The accuracy of each model based on R^2 and the number of clusters for each tournament type (because in the case of Approach 4 it is not deterministically chosen) are given by Table C.2. The out of the bag error (OOB) [125] has also been calculated. The models fit well, and a high value of both the accuracy measures on the test data and the OOB error indicate that the model is not over fitting.

Tournament type	Clustering Approach	Number of clusters	R^2 training data	R^2 test data	R^2 OOB score
standard	Approach 1	2	0.998831	0.987041	0.983708
	Approach 2	2	0.998643	0.978626	0.969202
	Approach 3	2	0.998417	0.985217	0.976538
	Approach 4	2	0.998794	0.990677	0.982959
noisy	Approach 1	2	0.997786	0.972229	0.968332
	Approach 2	2	0.997442	0.963254	0.955219
	Approach 3	2	0.997152	0.953164	0.940528
	Approach 4	3	0.996923	0.950728	0.935444
probabilistic ending	Approach 1	2	0.997909	0.981490	0.978120
	Approach 2	2	0.997883	0.973492	0.967150
	Approach 3	2	0.990448	0.890068	0.875822
	Approach 4	2	0.999636	0.995183	0.992809
noisy probabilistic ending	Approach 1	2	0.995347	0.957846	0.952353
	Approach 2	2	0.992813	0.909346	0.898613
	Approach 3	2	0.990579	0.824794	0.806540
	Approach 4	4	0.989465	0.841652	0.824052
over 45,600 tournaments	Approach 1	2	0.997271	0.972914	0.969198
	Approach 2	2	0.996323	0.951194	0.940563
	Approach 3	2	0.993707	0.906941	0.891532
	Approach 4	3	0.993556	0.913335	0.898453

Table C.2: Accuracy metrics for random forest models.

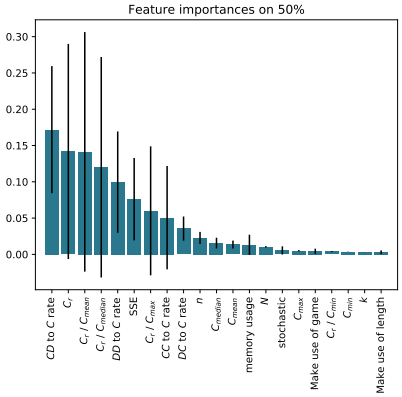
The importance that the features of Table 4.5 had on each random forest model are given by Figures C.6, C.7, C.8, C.9 and C.10. These show that the classifiers stochastic, make use of game and make use of length have no significant effect, and several of the features that are highlighted by the importance are inline with the correlation results. Moreover, the smoothing parameter k appears to no have a significant effect either. The most important features based on the random forest analysis were C_r/C_{median} and C_r/C_{mean}.

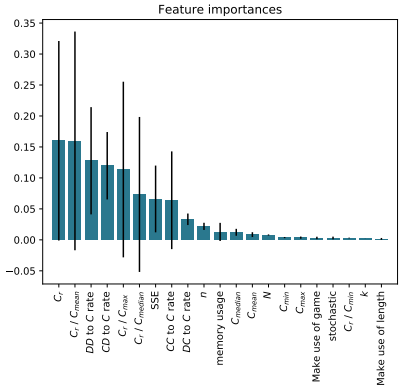




(a) Importance of features for clusters on 5% performance.

(b) Importance of features for clusters on 25% performance.

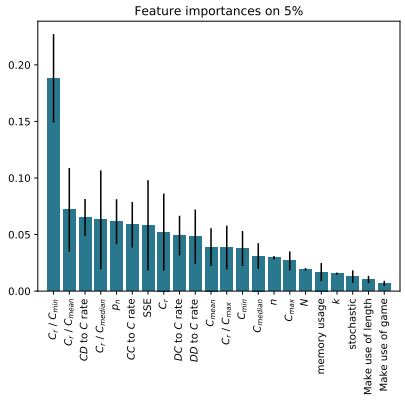




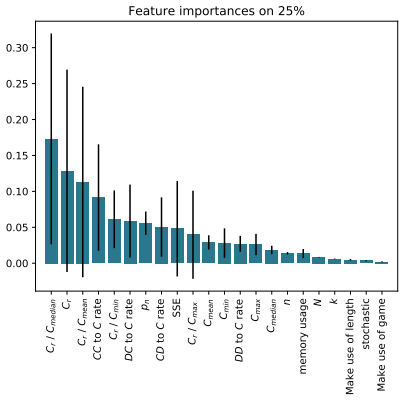
(c) Importance of features for clusters on 50% performance.

(d) Importance of features for clusters based on $k$means algorithm.

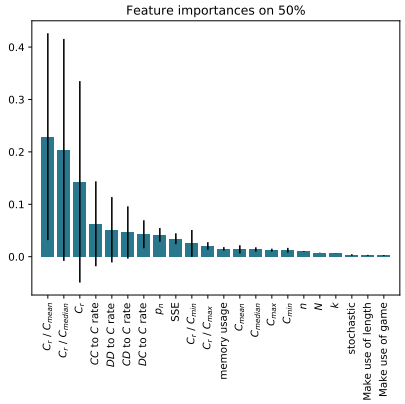
Figure C.6: Importance of features in standard tournaments for different clustering methods.



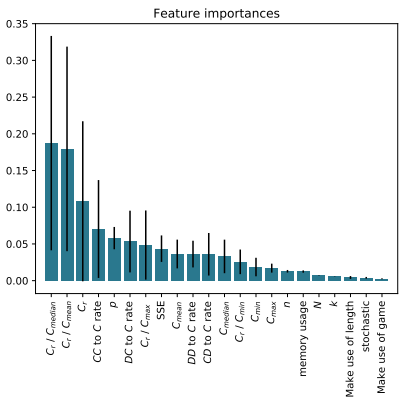
(a) Importance of features for clusters on 5% performance.



(b) Importance of features for clusters on 25% performance.

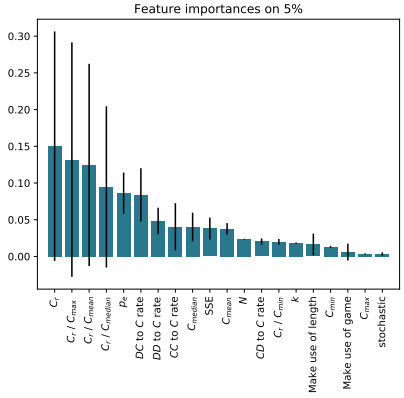


(c) Importance of features for clusters on 50% performance.

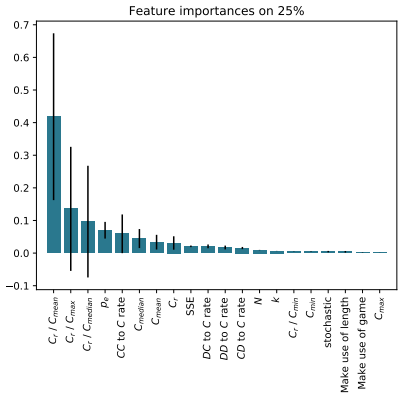


(d) Importance of features for clusters based on $k$means algorithm.

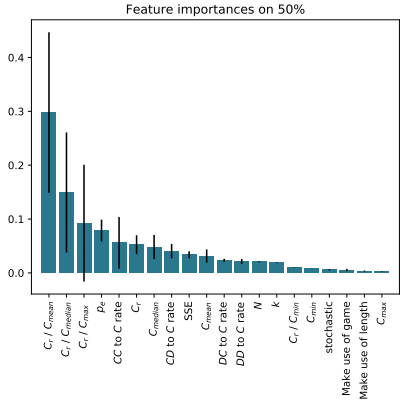
Figure C.7: Importance of features in noisy tournaments for different clustering methods.



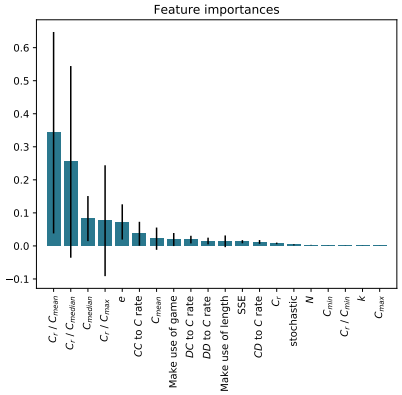
(a) Importance of features for clusters on 5% performance.



(b) Importance of features for clusters on 25% performance.

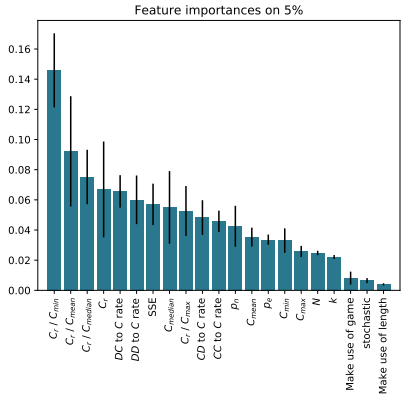


(c) Importance of features for clusters on 50% performance.

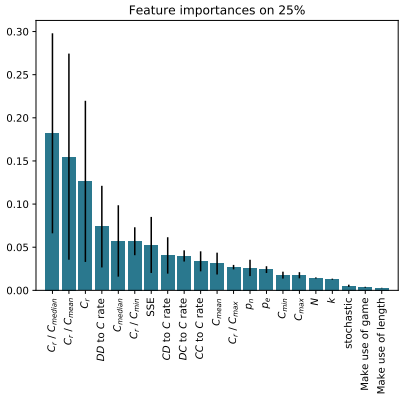


(d) Importance of features for clusters based on $k$means algorithm.

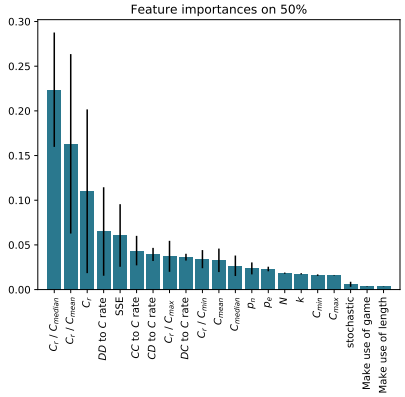
Figure C.8: Importance of features in probabilistic ending tournaments for different clustering methods.



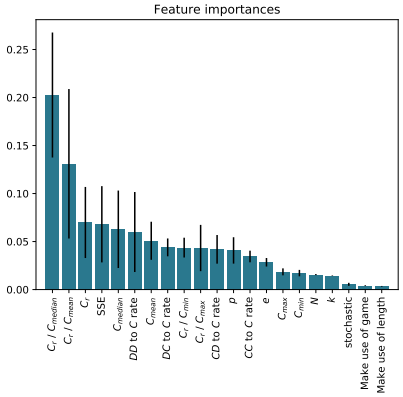
(a) Importance of features for clusters on 5% performance.



(b) Importance of features for clusters on 25% performance.

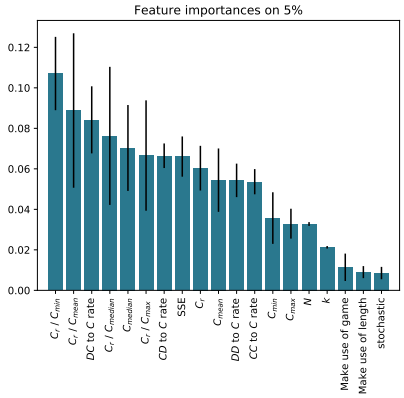


(c) Importance of features for clusters on 50% performance.

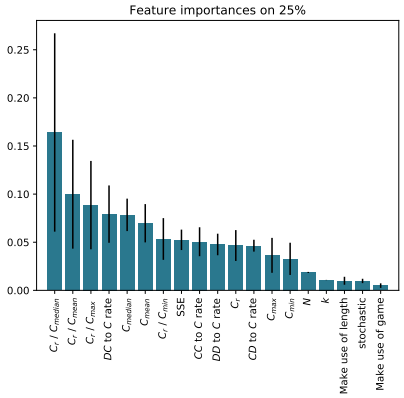


(d) Importance of features for clusters based on $k$means algorithm.

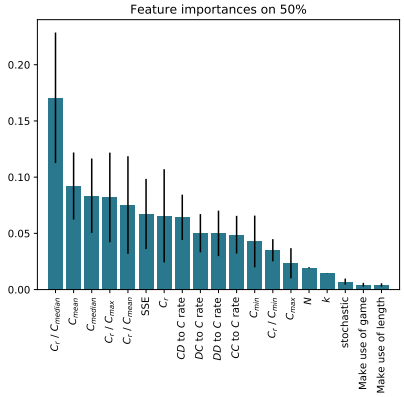
Figure C.9: Importance of features in noisy probabilistic ending tournaments for different clustering methods.



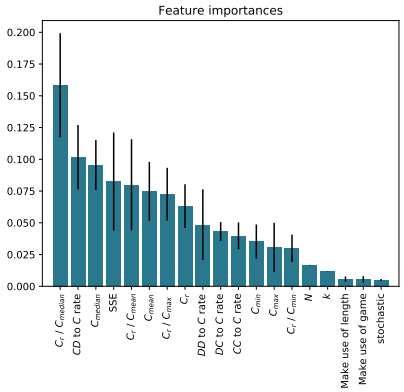
(a) Importance of features for clusters on 5% performance.



(b) Importance of features for clusters on 25% performance.



(c) Importance of features for clusters on 50% performance.



(d) Importance of features for clusters based on $k$means algorithm.

Figure C.10: Importance of features over all the tournaments for different clustering methods.

Appendix D

Table of parameters (per Chapter)

The parameters used throughout the thesis. There are given by Table D.1 alongside a brief explanation. Note that they are reported by Chapter.

	Parameter	Explanation
Chapter 3	N	number of abstracts
	n	number of topics
	c_i^j	the percentage contributions of the i^{th} topic for the j^{th} abstract
	c^*	the highest percentage contribution
Chapter 4	N	number of strategies
	k	number of tournament repetitions
	n	number of turns per match
	p_n	the probability of noise
	p_e	the probability of the match ending in the next turn
	C_r	the cooperation rate of a strategy
	R	the rank of a strategy
	r	the normalised rank of strategy calculated as $R/N - 1$
	$\bar{r}$	the median normalised rank of a strategy
	$C_{\max}$	the maximum cooperation rate of a given tournament
	$C_{\min}$	the minimum cooperation rate of a given tournament
	C_{mean}	the mean cooperation rate of a given tournament
	C_{median}	the median cooperation rate of a given tournament
	C_r / $C_{\max}$	a strategy's cooperation rate divided by $C_{\max}$
	$C_{\min}$ / C_r	$C_{\min}$ divided by a strategy's cooperation rate
	C_r / C_{median}	a strategy's cooperation rate divided by C_{median}
	C_r / C_{mean}	a strategy's cooperation rate divided by C_{mean}
Chapter 5	N	number of opponents
	p	a vector describing a given memory-one/reactive player
	q	a vector describing a given memory-one opponent
	K	the number of self interactions
Chapter 6	N	number of turns
	S	a sequence player
	Q	an given opponent which can be any IPD strategy
	P	a population of potential best response sequences
	b	the bottleneck
	G	a number of genetic algorithm generations
	p_m	the probability that each gene of an sequence being flipped
	K	maximum size of a population
Chapter 7	x	input
	y	expected output
	$\hat{y}$	predicted output
	ϕ	activation function
	N	number of turns in a sequence
	p_o	the probability of opening with a cooperation
	t	the number of time steps in a sequence
	s	the size of a tournament
	k	number of tournament repetitions
	n	number of turns per match
	R	the rank of a strategy
	r	the normalised rank of strategy calculated as $R/N - 1$
	$\bar{r}$	the median normalised rank of a strategy
	p	the probability of cooperating at each turn

Table D.1: The parameters used throughout the thesis per Chapter.