Cooperative Strategies in Multi-agent Systems Interim Report

Lauren Moore
Supervisor: Argyrios Deligkas
Full Unit Project 2020

Contents

1 Prisoner's Dilemma

1.1 Introduction to the Prisoner's Dilemma

The Prisoner's Dilemma is a game in which two participants play against each other in an attempt to either minimise or maximise their own score. While the analogy used to explain the game states both prisoners are attempting to minimise their score (prison sentence), most tournaments (including my own) has strategies attempting to maximise their score. Each participant or agent is given the option to either cooperate or defect, with scores determined by comparing the two participants' decisions. While the specific amount of points for each combination of decisions varies depending on the implementation, the participants always get an equal but small score when both cooperate, an equal but smaller score if both defect and uneven scores if one defects and one cooperates. The defecting agent will receive the largest amount of points possible while the cooperator will receive the smallest possible amount of points, traditionally 0. The Iterative Prisoner's Dilemma is a variation on this where the agents play the game multiple times in a row. A strategy is a set of logic or rules for how an agent will act normally depending on how their opponent has acted in previous rounds. The normal Prisoner's Dilemma only allows for very basic strategies as it is only played once (either cooperate, defect or random), while the iterative version allows for the deployment of much more complicated strategies as the agents can base their decisions on their opponents previous actions. When this iterative version is played with more than two strategies it is referred to as a tournament.

The method of evaluating strategies is generally how effectively they manage to minimise or maximise their score (depending on the version of the game being run) over multiple rounds of the game during a tournament. However it is necessary to run multiple tournaments to obtain an accurate analysis as factors such as the ordering and what other strategies are in the tournament will affect how effective a strategy appears to be. Study of the Prisoner's Dilemma and comparison of strategies is incredibly important as it has an incredibly large number of real world applications depending on the strategies entered into a tournament and how they are analysed. For example it can be used to attempt to maximise profits for a singular business (identifying a strategy that performs optimally against many others), or to maximise profits for two competing businesses (identifying two strategies that maximise payoff for both parties when played against each other). It also has important implementations with large numbers of players, such as the tragedy of the commons, which describes how when faced with a finite resource it is in players' best interest overall to all use as little of the resource as possible to avoid depleting it, which can be applied to concepts such as how power and resource consumption affect climate change.

1.2 Aims and Objectives

The overall aim of my project is to produce a program that runs a round robin tournament of the Iterative Prisoner's Dilemma. In order to achieve this there are multiple smaller aims, such as creating a GUI to allow the user to configure and run the tournament, and outputting results in as meaningful a way as possible. These aims are supported by the early deliverables set out in the project brief:

- Identification of a set of strategies - this will allow me to create a program with a good number of relevant strategies.

- Reports on the Iterative Prisoner's Dilemma and Tournaments - this will cement my knowledge and ensure my implementation of the IPD and tournaments is correct.
- Report on Previous work with the IPD - research for this will hopefully provide options for extensions to the project and show different examples of how results from tournaments are traditionally presented in academic settings.

1.3 Strategies

A strategy is the method a player or agent uses to determine which decision to make in each round. These can be as simple as always cooperating or as complicated as analysing all an opponents previous moves to attempt to work out which strategy it is following and react accordingly. There are two main types of strategy - basic and adaptive.

Basic strategies are the simplest type of strategies, as they do not consider any history or perform any analysis in order to make their decision. There is therefore an incredibly limited amount of these strategies, which are listed below.

- alwaysCooperate - A strategy that cooperates regardless of the game history and any other factors.
- alwaysDefect - A strategy that defects regardless of the game history and any other factors.
- alwaysRandom - A strategy that has a 50% chance of making each decision on each turn.
- varyingRandom - A strategy that has a random chance, assigned by the runner of the tournament, of cooperating.

Adaptive strategies are more sophisticated as they can perform analysis on the match's history. This allows for many more possibilities for strategies as they are able to react to their opponent's moves. The vast majority of strategies are adaptive.

- titForTat - A strategy that cooperates as its first choice, then copies its opponent's previous move. Rapoport et al. [1970]
- Spiteful - A strategy that always cooperates until its opponent defects, in which case it defects on every subsequent turn. Axelrod [2006]
- softMajority - A strategy that begins by cooperating and then continues to cooperate as long as the opponent has cooperated a number of times equal to or greater than its number of defections. Axelrod [2006]
- hardMajority - A strategy that begins by defecting and then continues to defect as long as the opponent has defected a number of times equal to or greater than its number of cooperations. Axelrod [2006]
- varyingMajority - A strategy that begins by cooperating then if the opponent has cooperated more times than it has defected it will cooperate. The strategy only considers a set number of previous moves, which is assigned by the runner of the tournament at the beginning.
- periodicDDC - A strategy that plays defect, defect, cooperate on a continuous cycle.

- periodicCCD - A strategy that plays cooperate, cooperate, defect on a continuous cycle.
- periodicCD - A strategy that plays cooperate, defect on a continuous cycle.
- mistrust - A strategy that is similar to tit for tat (and is sometimes called suspicious tit for tat because of this). It begins by defecting and then copies its opponent's previous move.
- pavlov - A strategy that begins by cooperating, then cooperates only if both players made the same move in the previous round. Wedekind and Milinski [1996]
- titForTwoTats - A strategy that cooperates on the first two moves, then defects if its opponent defected on both of the two previous moves.
- hardTitForTat - A strategy that cooperates on the first two moves, then defects if its opponent defected on one or more of the two previous moves.
- gradual - A strategy cooperates until its opponent defects. Whenever its opponent defects the strategy will defect the total number of times its opponent has defected, then cooperate twice in a row. Beaufils and Jean-Paul [1996]
- prober - A strategy that plays the moves defect, cooperate, cooperate then always defects if the opposing strategy cooperated in moves 2 and 3. If this condition is not met it plays the same as tit for tat. Mathieu et al. [2000]
- mem2 - A strategy that makes the same decisions as tit for tat in the first two moves. The strategy then chooses a new strategy to follow every two moves based on the outcome of the last two moves:
  - if the result was both cooperate each time, follow tit for tat
  - if the result was cooperate defect or defect cooperate each time follow tf2t
  - if the result was any other then play always defect
  - if always defect has been chosen twice then it will always choose always defect

  Li and Kendall [2013]
- titForTatWithForgiveness - A strategy similar to tit for tat in that it begins by cooperating then copies its opponent's previous move in every subsequent round. However, it has a random chance (normally quite low) of cooperating even if its opponent defected on the previous round.
- scoreBased - A strategy that makes its decision based on the difference between its score and its opponent's score. If the strategy's score is higher by a certain amount it will cooperate, otherwise it will defect.

List of strategies based on - Mathieu and Delahaye [2017], with some strategies added from my other research and some suggested by my advisor.

1.4 Iterative Prisoner's Dilemma

The Iterative Prisoner's Dilemma is a variation on the Prisoner's Dilemma in which multiple rounds of the game are played with the same two players. This allows for far more sophisticated and interesting strategies as they can access and make decisions on their opponent's previous moves. The consequences of this are that players are able to make predictions about what their opponent will decide in the next round and, arguably more significantly, it encourages players to choose strategies that cooperate often as otherwise their opponent is able and likely to retaliate.

Which strategies perform the best in the Iterative Prisoner's Dilemma depends on multiple factors, including some outside of the control of the strategy itself. A good strategy must obviously be able to read and analyse the decisions made by its opponent in the past, however its performance will also be affected by which strategies it is playing against as well as the order in which it plays against them. This is explored in multiple articles on the subject, including Axelrod's article discussing his famous first tournament. In this article he mentions how strategies that are inclined to be cooperative do well when playing in a tournament against each other as they often establish mutual cooperation.

2 Tournaments

2.1 Introduction to tournaments

Tournaments are incredibly significant when discussing and researching the Iterative Prisoner's Dilemma as they are the main method for evaluating the performance of algorithms/strategies. They work by allowing multiple people to enter one or more strategies, which will play a number of rounds of the Prisoner's Dilemma against other participants. The type of tournament dictates whether or not they play every other strategy or just some, and many also include a rule that strategies play against a copy of themselves.

2.2 Types of tournament

Round robin style tournaments work by playing every strategy against every other strategy and, depending on the rules of the tournament, a copy of themselves. Many tournaments around the beginning of the Prisoner's Dilemma's popularity were round robin tournaments.

Evolutionary tournaments are a type of tournament where the score of a strategy in a particular round (also referred to as a generation) determines how many instances of the strategy appear in the subsequent round. These can take place in a similar fashion to the round robin or on a 2d grid in order to better represent actual populations of organisms. In this case the strategy with the largest score after their game takes over the space of the losing strategy, leading to strategies that win being more represented. The results of these tournaments are particularly significant for researchers who wish to apply their results to real life behaviour of both humans and animals. They are mainly concerned with attempting to explain how cooperative behaviours persist in species despite more selfish behaviours having better short term results for an individual. Multiple theories have emerged from or been researched by these tournaments, including direct and indirect cooperation and group selection. Direct cooperation is the theory that repeated cooperation with someone you've met before is likely if they have previously cooperated and indirect cooperation is the same concept but regarding someone who you've only heard cooperates. Group selection, originally proposed by Darwin, states that natural selection can be applied to a group as a whole as well as just individuals. The application of this in the Prisoner's Dilemma is that a group of nice strategies all interacting with each other is likely to thrive and be better represented in subsequent rounds. (Nowak and May [1992])

Knockout tournaments work by playing pairs of strategies against each other. The strategy that gains the least points in the round is then eliminated, with this pattern continuing until only one strategy remains. This type of tournament is not commonly used as it has multiple flaws, such as the seeding (which strategies play against which) affecting which strategy wins. In addition, it is not fair to use this type of tournament to declare a second place due to the fact that any strategy played by the winning strategy could theoretically be the second best, even if it was knocked out early in the tournament. There are, however, ways of improving the traditional knockout tournament in order to gain more useful results such as holding a double-elimination tournament instead. In this tournament an entrant can lose a single round without being eliminated, which reduces the risk of an effective strategy being knocked out due to bad luck (e.g. unlucky seeding or unlucky random number generation in a match).

Another decision made by the designer of a tournament that will affect the outcome is whether or not the game lasts a set number of rounds. A large number of strategies are designed to

cooperate where possible in order to minimise or avoid retaliation from their opponent. However if the players know when the last round is there is no incentive to cooperate on that round as there is no chance of retaliation. The problem that this causes is that there is also no reason to cooperate on the second to last round (or any round before that), as the opponent is nearly guaranteed to defect on the next round regardless. In order to combat this most modern tournaments are of indeterminate length rather than being fixed, meaning that at the end of every round there is a possibility that the game will end. Davis and Brams [1999]

2.3 Winning strategies and notable tournaments

The most notable tournament in the field of the Prisoner's Dilemma is the first tournament run by Robert Axelrod in 1980. It consisted of 14 entered strategies as well as one that randomly chose between cooperating and defecting, which played in a round robin style tournament (every strategy played against every other strategy and a copy of itself). The winning strategy in this tournament, tit for tat, is still discussed today and many different versions and possible improvements have been suggested since its invention. For example, the a strategy commonly referred to as tit for tat with forgiveness follows the basic principles of tit for tat, but also has a small chance of cooperating on a turn where it is meant to defect. The benefit of this is that it can (depending in the strategy it is playing against) allow the strategy to re-establish cooperation with its opponent if they are stuck in a cycle of both defecting. While it was not the first appearance of tit for tat, Axelrod's first tournament is responsible for its popularisation within the field. The impact of this is that many people have attempted to improve on this, with strategies such as tit for tat with forgiveness and slow tit for tat. (Axelrod [1980a])

Axelrod ran more than one tournament, with his second introducing the idea of the tournament having a random probability of ending rather than running for a set number of rounds. Tit for tat won this tournament again, further cementing its place as a very significant strategy in the field. The whole results table also impacted the field as it partially disproved some of the conclusions drawn from the first tournament. Axelrod originally theorised that strategies which were nice and forgiving did better overall. However in the second tournament some of these strategies, in Axelrod's words, 'fell victim to' strategies which were designed to take advantage of their cooperative natures. Additionally, in his article regarding this tournament Axelrod began considering the evolutionary type of tournament instead of the round robins run in his first two. He studied cooperation through the lens of evolutionary biology in order to explain why it persists in nature, becoming so interested in this that he released an entire book on the subject, entitled the evolution of cooperation. (Axelrod [1980b])

In 2004 and 2005 the IEEE Congress on Evolutionary Computing ran Prisoner's Dilemma tournaments to mark the 20th anniversary of Axelrod's publication. They had two significant differences to the original - the concept of noise (a small probability of a strategy's move would be mis-executed) was introduced and teams/researchers were invited to submit more than one strategy. The second change proved to be the most significant, as strategies from the University of Southampton won both tournaments by entering 'teams' of strategies that intended to recognise each other and work together. They did this by employing a pre-agreed series of moves in order to recognise each other, then either all cooperate or act preferentially towards one member of the team to maximise their individual score. (Rogers et al. [2007])

3 Previous work with the Iterative Prisoner's Dilemma

3.1 Theory behind strategies

There are many different strategies and while they are mostly just classified as either basic or adaptive, there are other ways of analysing and grouping strategies. One of these methods is grouping them by shared properties, such as niceness and whether or not they are forgiving. Axelrod [1980a] These properties provide a possible extension to my project as I could use my program to run tournaments to investigate how they affect a strategy's performance. In addition, I could also investigate how the properties of its competitors' affect a strategy's performance.

A strategy is considered to be nice if does not defect before its opponent does. This is very important as many adaptive strategies look to punish opponents that defect, so defecting first could cause a cycle of retaliatory defecting.

A strategy is considered to be forgiving if it does not continue defecting long after its opponent has began or returned to cooperating. This is important as not entering into mutual cooperation is likely to cause the opponent to begin defecting again. If a strategy's opponent is defecting then its maximum possible score is severely limited, as its options are cooperate and get the lowest possible round score or defect and get the second lowest possible round score (if the traditional payoffs are being used).

A strategy is known as retaliating if it begins defecting for some period of time (not necessarily forever) once its opponent defects. This is important as strategies without this feature are likely to be exploited by some strategies as they provide no punishment for defecting against them.

A strategy is non-envious if it is not trying to score more than its opponent. This can be beneficial as a strategy attempting to defect in order to gain more points than its opponent could cause patterns of mutual defection, minimising the scores of both itself and its opponent.

3.2 Literature Survey

Mathieu and Delahaye [2017] - This paper has a similar aim to my project in that they are attempting to use tournaments in order to analyse the performance of prominent strategies in the area of research. In relation to my project aims for this term it was particularly useful when gathering the list of strategies I wanted to implement. The list featured in this paper formed the basis of my list, along with suggestions from my advisor and some other strategies I discovered during my research. While I have not worked on effectively presenting my tournament results during my work this term I will definitely heavily consider the ideal method of presentation this term, and the results presentation in this paper may be a good format to consider. The paper then goes on to attempt to define new, more successful strategies based on which strategies performed well in their early tournaments, which is an interesting possible extension to my project.

Axelrod [1980a] - This paper by Robert Axelrod describes his famous 1980 tournament and also draws some conclusions based on the data produced by it. Possibly the most significant impact on my work is the fact that it popularised the tit for tat strategy, which led to both that and multiple variants and attempted improvements being included in my list of strategies. I also found Axelrod's theories on properties of strategies affecting their performance incredibly interesting as a method of analysis and have therefore included it in my report, as well as considering it as an extension for my project next term.

Axelrod [1980b] - This paper, again by Robert Axelrod, details his second attempt at running a tournament. Tit for tat was victorious again, proving that it is a strategy worth including and studying in my own work. It also expanded on his previous theories regarding properties of strategies but altered some conclusions, such as nice strategies seemingly gaining more points. This tournament showed that while nice strategies are likely to be successful when interacting with eachother, they will become exploited if they are faced with many not nice strategies. While both this and Axelrod's original paper are significant and laid the foundation for some modern research the fact that they are now 40 years old means that they cannot form the entirety of my research. Many breakthroughs and changes have happened in the field since then, including the rise of the pavlov (or win-stay, lose-shift) strategy and the interest in evolutionary tournaments instead of round robins.

Nowak and May [1992] - This article shows one of the ways that research in the field developed after the initial interest in Axelrod's tournaments. It illustrates one of the ways in which an evolutionary tournament can be run, using a 2d grid and having each strategy play against its immediate neighbours. This could be a possible extension for my project, especially as the graph view of the results each round would be interesting to simulate and possibly more meaningfully communicate the results of a game when compared to a score.

Rogers et al. [2007] - This paper was authored by the team that won the Prisoner's Dilemma tournaments run by the IEEE Congress on Evolutionary Computing. It describes the unusual method of entering multiple strategies into a tournament which will function similar to a team by identifying each other. The ability to recognise other strategies is discussed in other papers, particularly ones focused on evolutionary tournaments, but has not been achieved and utilised this effectively in round robin tournaments before. Both this recognition-based cooperation and the introduction of noise could be used as extensions to my project. While they would both be possible extensions to the round robin style of tournament they may have more significance and allow me to draw more context-based conclusions if I chose to implement them in conjunction with an evolutionary tournament. This is because they both have real world counterparts in organisms (trait recognition and miscommunication/mistakes), which is the context that evolutionary tournaments are generally applied to.

3.3 Proof of Concept Program

My proof of concept program implements all the strategies listed in this report and uses them to run games of the Iterative Prisoner's Dilemma. This program refers to a round as a single decision made by each strategy and a game as a set of rounds played between the same two strategies.

Initially I implemented the most basic three strategies mentioned in the report - alwaysCooperate, alwaysDefect and alwaysRandom. These were enough to allow me to decide the best way to implement these strategies, which I decided was having each strategy be its own class which was a subclass of an overall strategy superclass. This allowed me to ensure each strategy had a method that returned its decision in each round as well as associating the amount of points earned with the instance of a particular strategy.

In order to implement the more complex adaptive strategies I needed to then add a data structure storing the history (each player's moves) of a game so the strategy could analyse it. This led to the development of the game class, which took a number of rounds and two strategies as parameters and played the number of rounds specified in the parameter. The history could then

be stored as two variables in that class (one for each strategy) and accessed by the two strategies playing that game.

```
AlwaysCooperate playing against AlwaysDefect
Round 1: c vs d 0 points vs 5 points
Round 2: c vs d 0 points vs 5 points
Round 3: c vs d 0 points vs 5 points
Round 4: c vs d 0 points vs 5 points
Round 5: c vs d 0 points vs 5 points
Round 6: c vs d 0 points vs 5 points
Round 7: c vs d 0 points vs 5 points
Round 8: c vs d 0 points vs 5 points
Round 9: c vs d 0 points vs 5 points
Round 10: c vs d 0 points vs 5 points
```

Figure 1: The output of the program showing the results of a game

I built upon the proof of concept program discussed in the first report in order to implement functionality to run a round robin tournament. While the project brief and my plan describes them as two separate deliverables, it was agreed with my advisor that it would make more sense to combine them into one.

A round robin tournament is formed of multiple games of the Iterative Prisoner's Dilemma so my program works by calling the playGame function in my game class multiple times. I began by creating a new tournament class named roundRobin which takes a list of all the participating strategies and the total number of rounds in each game. The runTournament function in this class calls the runGame function for each possible pair of strategies, including making strategies play against a dummy version of themselves.

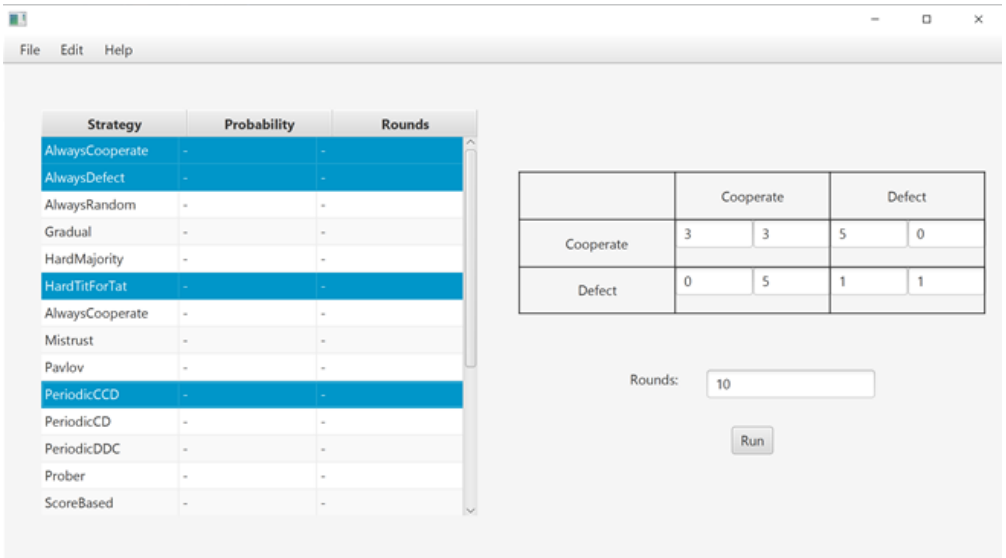


Figure 2: The screen of the program that allows configuration of a tournament (Blue rows have been selected and will be entered when tournament is run)

It was necessary to add a function that ensures that the length of each game is not fixed. As discussed in the types of tournament section, this is important because fixed-length tournaments have an optimal strategy (alwaysDefect), whereas infinite or randomly ending tournaments do not. After discussion with my advisor I implemented a function that takes the total number of rounds and runs three games of random length. This is done by first generating a random value x1 between 1 and the total number of rounds, which is the length of the first game. The length of the second game x2 is then generated, which is between 1 and the total number of rounds - x1. The length of the final game is the total number of rounds - x1 - x2. While the values of x1 and x2 are randomised, they are the same for every pairing in order to maintain fairness.

Once all the logic for the running of the tournament was implemented, I began to implement a GUI. This allows the user to configure the tournament by selecting the participating strategies, payoff values and the number of rounds in each game. It also attempts to present the results in a readable, concise format (ordered results list and grid showing points earned in each game) as well as the full, verbose results in a separate screen.

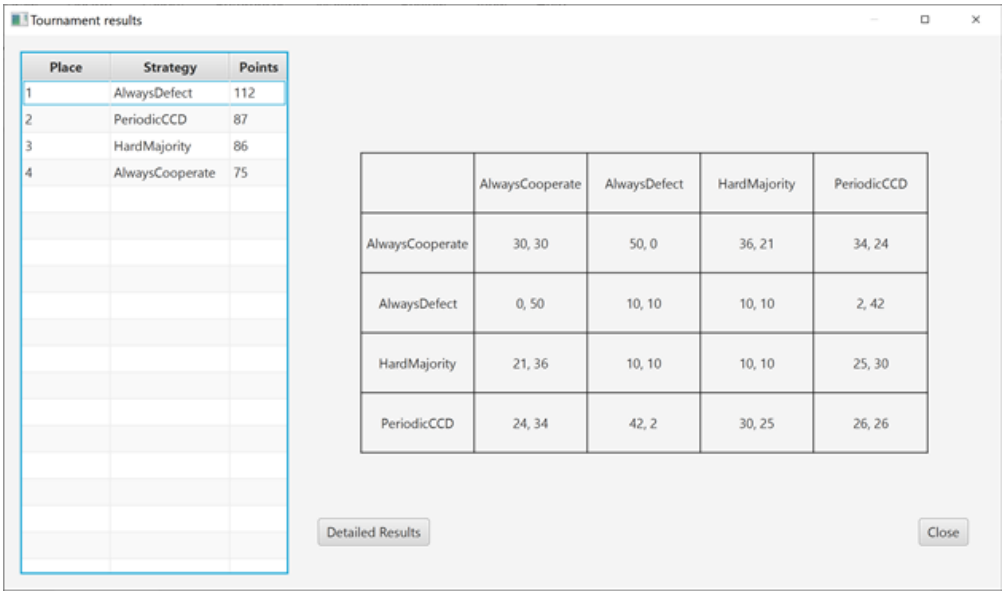


Figure 3: The screen of the program that displays the concise results of the tournament

All of this development was done using test-driven design (TDD) and a version control system. Test-driven design is a software engineering methodology that works by first writing a unit test for the intended functionality then writing code in order to make the program pass that test. The benefit of this is that you can be confident that each small section of code written functions correctly. While this did greatly minimise the number of errors I encountered I did still encounter a small amount when testing the GUI, highlighting the importance of using multiple types of testing. Use of a version control system (in my case SVN), was incredibly important for my project. While in most cases it is only useful for checking when and how code has been altered, I had to use it as a method of recovering my project. A few weeks into the project my laptop became unusable, and without the use of SVN all my work would have been lost, setting my progress back significantly.

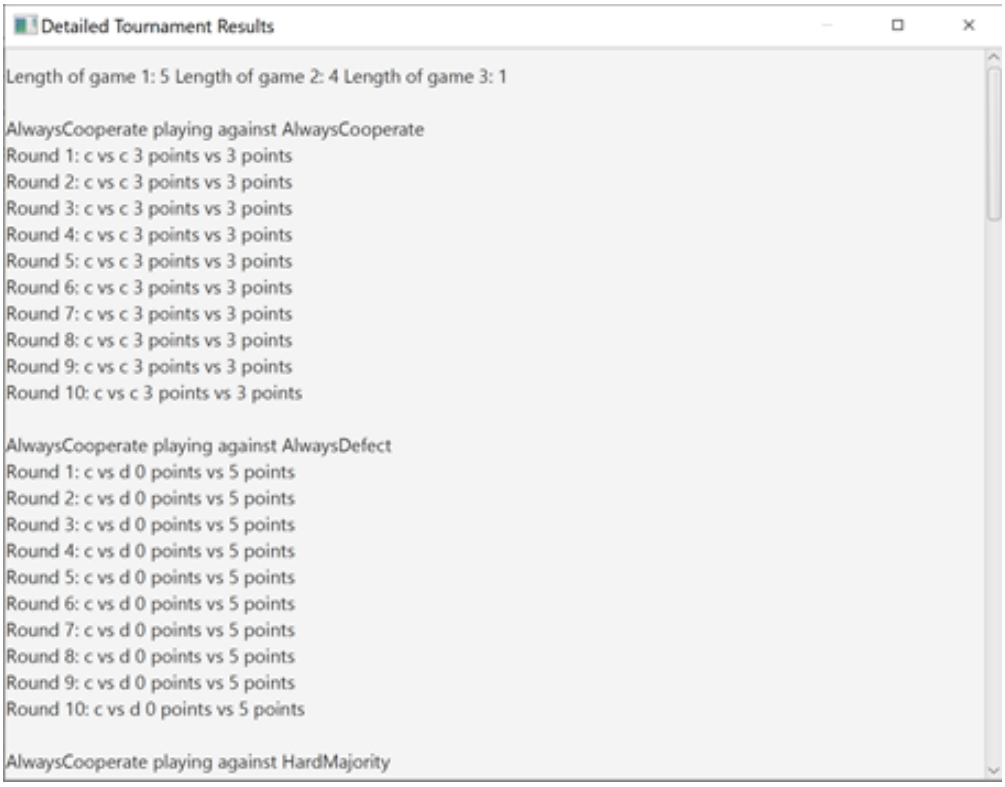
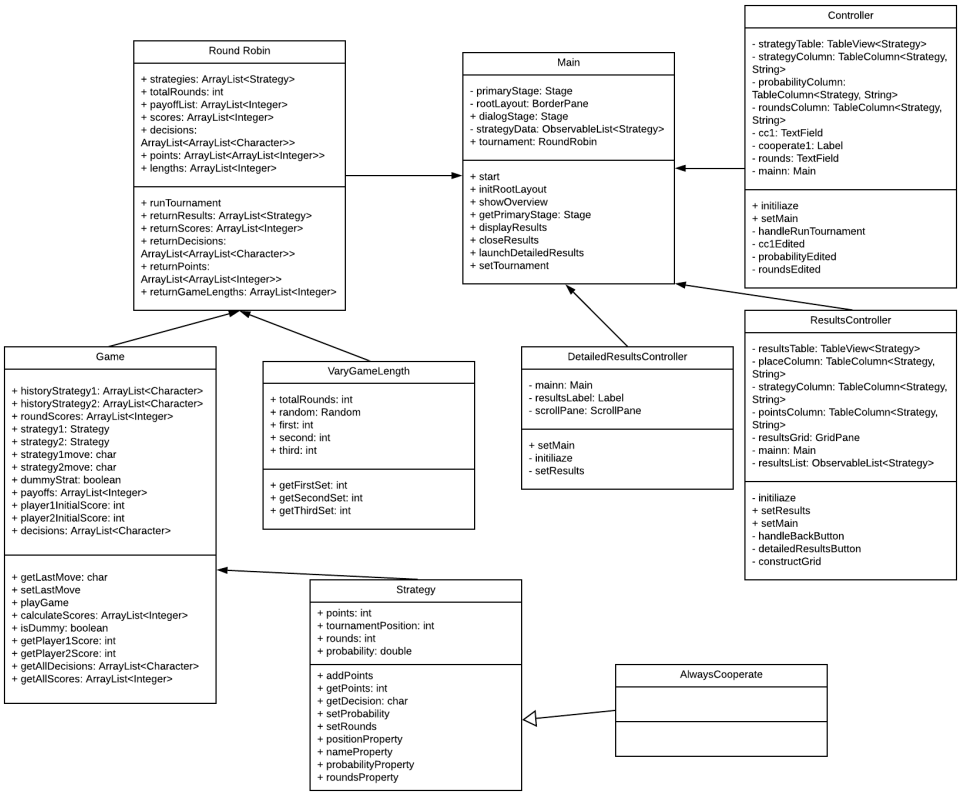


Figure 4: The screen of the program that displays the verbose results of the tournament, which is scrollable due to its length



The UML diagram above shows the structure of the final program, including the classes used to create and control the GUI. I decided to exclude some detail from it in order to make it more readable and useful. Specifically, I excluded all but one of the strategy classes as they are identical in regards to UML and some labels and textfields in the controller class as there were a large amount that performed similar functions.

The following link shows a demonstration of the program's functionality: https://youtu.be/BXPYN4mFWsU

3.4 Planning and Timescale

The structure of my project has changed from the initial structure and timescale that I set out in my project plan. While I did achieve everything I intended to this term (implementing strategies, a singular round of the Prisoner's Dilemma, a game of the Iterative Prisoner's Dilemma and a round robin tournament), I also began implementing a simple GUI for my program. The main objective of this was to effectively show the functionality of my program in my viva and report. This means that my plan for next term will need to be altered as the first two technical milestones, implementing a GUI and outputting statistics, have been mostly achieved.

My aims for next term will be:

- Improve existing GUI (styling, ensure it resizes properly, explanations of strategies, help section)
- Improve output of detailed results (styling, think about a more readable format)
- Add an option to output results to a file
- Add another kind of tournament as an extension, likely an evolutionary tournament

While I did initially use the headings set out in my plan for the interim reports, I found that some were too similar and therefore caused too much repeated information in my reports. Because of this, I decided to change the structure and contents of some of my reports by removing these repeated headings.

I also deviated significantly from the timescales set out in my project plan. While I mostly planned to write reports and programs simultaneously, I found that I needed to prioritise programming for most of the term in order to achieve my aims. Additionally, I found that writing the reports took less time than anticipated but programming took more, particularly as I achieved more than I initially set out to this term. Overall I feel that my changes to my planned timescale were beneficial, as I have achieved all my aims for this term and am in a good position to not only achieve my aims for next term but also choose and implement some sort of interesting extension as well.

3.4.1 Diary Summary

I chose to make diary entries that corresponded to a week each. In each entry I felt that the most important information was what I have achieved that week, the questions I had for my advisor (and his responses) and everything else discussed in the weekly meeting. The summary of my completed work that week is important to document because it shows the evolution of my program each week and shows that I have achieved everything that I wanted to this term. The list

of questions and answers provides an easily accessible reference for me (if my paper notes are not accessible) as well as showing the decisions I have made and options I have considered. The summary of the discussion with my advisor is important as he often makes suggestions or gives feedback on what I have worked on that week.

References

Robert Axelrod. Effective choice in the prisoner's dilemma. *The Journal of Conflict Resolution*, 24(1): 3–25, 1980a. ISSN 00220027, 15528766. URL http://www.jstor.org/stable/173932.

Robert Axelrod. More effective choice in the prisoner's dilemma. *The Journal of Conflict Resolution*, 24(3):379–403, 1980b. ISSN 00220027, 15528766. URL http://www.jstor.org/stable/173638.

Robert Axelrod. *The evolution of cooperation*. Basic Books, 2006.

Bruno Beaufils and Delahaye Jean-Paul. Our meeting with gradual: A good strategy for the iterated prisoner's dilemma. 5, 04 1996.

Morton D Davis and Steven J Brams. The prisoner's dilemma. *Encyclopædia Britannica*, Jul 1999. URL https://www.britannica.com/science/game-theory/The-prisoners-dilemma.

Jiawei Li and Graham Kendall. The effect of memory size on the evolutionary stability of strategies in iterated prisoner's dilemma. *IEEE Transactions on Evolutionary Computation*, 18, 10 2013. doi:10.1109/TEVC.2013.2286492.

Philippe Mathieu and Jean-Paul Delahaye. New winning strategies for the iterated prisoner's dilemma. *Journal of Artificial Societies and Social Simulation*, 20(4):12, 2017. ISSN 1460-7425. doi:10.18564/jasss.3517. URL http://jasss.soc.surrey.ac.uk/20/4/12.html.

Philippe Mathieu, Bruno Beaufils, and Jean-Paul Delahaye. Studies on dynamics in the classical iterated prisoner's dilemma with few strategies. pages 177–190, 2000.

Martin Nowak and Robert May. Evolutionary games and spatial chaos. *Nature*, 359:826–829, 10 1992. doi:10.1038/359826a0.

Anatol Rapoport, Albert M. Chammah, and Carol J. Orwant. *Prisoner's dilemma: a study in conflict and cooperation*. Univ. of Michigan Press, 1970.

A. Rogers, R.K. Dash, S.D. Ramchurn, P. Vytelingum, and N.R. Jennings. Coordinating team players within a noisy iterated prisoner's dilemma tournament. *Theoretical Computer Science*, 377(1):243 – 259, 2007. ISSN 0304-3975. doi:https://doi.org/10.1016/j.tcs.2007.03.015. URL http://www.sciencedirect.com/science/article/pii/S0304397507001831.

C Wedekind and M Milinski. Human cooperation in the simultaneous and the alternating prisoner's dilemma: Pavlov versus generous tit-for-tat. *Proceedings of the National Academy of Sciences*, 93(7):2686–2689, 1996. ISSN 0027-8424. doi:10.1073/pnas.93.7.2686. URL https://www.pnas.org/content/93/7/2686.