The role of imitation in the conflict between co-operators and cheaters

2022

Dave Kyere

Supervisors: Matteo Cavaliere, Mohammed AI-Khalidi

A Report submitted to Manchester Metropolitan University for the degree of Computer Science in the faculty of Science and Engineering

Word count: 15,121

Contents

List of Figures

List of tables

Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work. This work has been carried out in accordance with the Manchester Metropolitan University research ethics procedures and has received ethical approval number 43540.

Signed

Abstract

This paper investigates imitation's role in the conflict between cooperators and defectors. By uncovering the role, strategies that help aid a cooperative population can be used for reference in game theory. Results needed for analysis are produced when a cellular automata model tests the various strategies, using the Chicken game and prisoner's dilemma. What was found was that defection is most effective when imitation is based on the neighbour with the highest payoff score. On the other hand, cooperation improves when strategies look at different criteria for selection.

Abbreviations

- Cooperation: **CO**
- Defection: **DE**
- Most successful neighbour: **MSN**
- Prisoner's Dilemma: **PD**

Chapter 1

Introduction

1.1 Background

Evolutionary game theory is how game theory applies to evolving populations in biology. One facet of game theory is the act of cooperation or defection. Some species through cooperation with one another bettered their chances of survival. However not all species choose to be kind for mutual survival; some instead increase their survival chances by cheating on opposing species. The behaviour dynamic is also widely present in economics, where selfishness is often rewarded when played correctly. Game theory takes these fundamentals to further understand correct strategy. Two players unaware of each other moves get the option to be cooperative for mutual payoff or be deceptive for maximum payoff. The prisoner's dilemma is a prominent non-zero-sum game that applies this scenario to two prisoners who are separated and pressured to confess on one another. While being unaware of each players moves, the choice each player makes could be beneficial or to their own detriment.

The issue with games such as the prisoner's dilemma is the inability to adapt and improve due to its finite nature. Strategy becomes a factor when the games are simulated for multiple rounds instead. For example, a player who sticks to an all-defective strategy could see success but also suffer depending on the choices on the opposition move. In a perfect environment, TFT (Tit for tat) was the most dominant strategy. The strategy relied on imitating whatever an opponent chose in the previous round. Cellular automata were used as a model to show the population of cooperative and defective agents. When the model was running each cell would copy the state of their neighbour with the highest payoff score, these scores often resembled the payoff matrix used in the prisoner's dilemma. Through this strategy, the game would eventually reach a state where most agents are cooperative. The discovery was that cooperation is the better choice for survival in a closed population.

1.2 The Problem

In the previous section a basic model for the iterated Prisoner's dilemma using Cellular Automata was discussed. The result of the experiment had players choosing to mutually cooperate for a stable payoff between neighbours. With game theory the aim is to always deeply understand why people make rational decisions. Therefore, it is vital that the best strategy is understood and explain behaviour. The standard model only looks at one possible tactic which doesn't prove peak effectiveness. While choosing the most successful neighbour

from the previous iteration may seem the most rational, winning a particular round equate to consistently high payoff score. To ensure that cooperation prevails against defection, it is important that the best means of imitation is tested. For example, memory and averages could prove to be more beneficial.

A Cooperative society is the best collective result (Momeni, 2018). Society wouldn't be as prevalent as today if it wasn't for mutual trade off. The risk that defection proposes upsets that reality, individually people can receive a high payoff when going against the majority held consensus. By demonstrating that defection doesn't cost well in a Populus promotes cooperation.

1.3 Aims and Objectives

The project aims to test various means of imitation in a cellular automation model to have an understanding on which strategies best aid a cooperative population.

The objectives are:

- To code a spatial game using cellular automata that shows the conflict between co-operators and defectors.
- Produce payoff matrices for the prisoner's dilemma and the Chicken game
- To test an imitation strategy that copies the most successful neighbour.
- To test an imitation strategy where the past payoff scores are considered.
- To test an imitation strategy where a player copies a neighbour with an average payoff score.
- To test a system where the imitation strategy is random.
- Create summary plots of different combinations of imitation to evaluate how they aid in a final cooperative population.

1.4 Report Structure

The report starts with an in-depth review of previous literature for the paper's hypothesis. The review should provide context and understanding for readers. Following the literature review, the next chapter involves the design process for the proposed model. In chapter a, understanding of the approach and methodology will be explained using diagrams of workflow. Additionally, the requirements needed to meet the objectives will be listed. With the design approach clear, the model implementation commences using screenshots and plots to show the success of each test. An evaluation chapter follows to understand the results produced, where context could be applied using discoveries from the literature. The evaluation would assess the solution's success against the aims and objectives. Finally, a conclusion chapter is placed to summarise all the key points discussed throughout the report.

Chapter 2

Literature Review

2.1 Introduction to Review

This literature review aims to undercover previously done work on spatial games involving the conflict between cooperators and cheaters. By grasping the theoretical concepts and examples explored in this chapter, the testing results would then contextualise discoveries and uncover new ones. This review will open and touch upon the research already established.

To fully understand all the findings in the literature, specific terminology needs to be defined:

- Strategy: In this context, Strategy is a set of rules or instructions that would yield the highest reward for a player.
- Zero sum game: One player's gain is equal to another players loss. Therefore, the net benefit is zero.
- Topology: "Studies properties of spaces that are invariant under any continuous deformation" (Pure Mathematics, 2015)
- Ecology: "The study of the relationships and interactions that occur between organisms and environments" (Vlebooks.com, 2022).
- ALL-C: This means all cooperation
- ALL-D: This means all defection
- TFT: "Tit-for-Tat"- A strategy used to best other opponents

2.2 Spatial Game theory

Spatial skills are the ability to recognise one's environment. It is often taught to kids to recognise what is above, below, left, and right to them (Edc.org, 2021). Understanding environmental conditions help determine appropriate behaviour. Biological objects can be seen to alter or use their environment to make decisions. Evolutionary game theory uses the study of spatial awareness as a foundation. Objects are often organised by spreading them across various spatial sites (Proceedings of the Royal Society of London. Series B: Biological Sciences, 2021). The behaviour of Each object is based on its local neighbours; from these local interactions, reasonable strategies can be calculated. Local interactions between two players are how game theory ties itself to offer behaviour explanations seen in social situations found in nature.

Jon von Neuman and Oskar Morgenstern are widely considered the founders of the theory of games (Vlebooks.com, 2022). They were fellow mathematicians whose publication of "The Theory of Games and Economic Behaviour" (see figure 1) in 1944 led the likes of John Nash to further expand on the topic. Their model was based on how players, who could make a range of choices, would interact with one another. The game theory states that "in a broad category of games, it is always possible to find an equilibrium from which neither player should deviate unilaterally (Mérő, 1998). This equilibrium can be applied to specific criteria as followed for every two-player game, such

1. The game is finite. Essentially a certain number of moves must bring the game to an end.
2. The game is zero-sum.
3. All the possible options are known to all the players present. Each player must also be aware that their opponent is presented with the exact same rule set.

Figure 1: Von Neumann & Morgenstern

2.2a Zero-sum games

With this outline for game theory broken down above, certain games become 'interesting' and 'uninteresting'. An example of an uninteresting game is Chess. Chess aligns with the criteria in which the equilibrium can be applied. Chess is finite since the aim is for one player to eliminate another player's king. The game is also zero-sum since a player winning a chess match is also dependent on the opposing player losing. Lastly, Chess also has complete information. Both players are aware of the possible moves each Chess piece can make. With this, there is no room for any random variance in gameplay. With enough computing power, the best possible move can be accurately predicted. In order to observe an "exciting" game,

there needs to be a situation for deceit to occur (Vlebooks.com, 2022). Additionally, the success of a player's move should be altered by the opponent's move.

The matching pennies game is an example of a Zero-sum game (Mookherjee and Sopher, 1994). The basic principle is that two players each have one coin. Both players would place their coins on a table when it was time. The face of the coin each player chooses would depend on who gets to collect a coin and who loses one. A rule set is placed beforehand, stating that player A wins if both coins match. Player B wins, however, if the two coins have opposite results. Both players have no idea what their opponent will choose; therefore, the first round is entirely reliant on luck. There would ultimately be a loser and a winner, and the possibility for a mutual payoff is zero (see figure 2).

		Player *B*'s move	
		heads	**tails**
Player *A*'s move	**heads**	1, -1	-1, 1
	tails	-1, 1	1, -1

Figure 2: Payoff matric for matching pennies game

Game theory is all about choosing the best strategy for the best payoff. The best strategy for the penny game is randomness. A 50/50 option every round the game is played should result in the highest amount gained. If a player decided to predict its opponent's strategy, the opponent could simply adapt and pick up on the tactic. In this scenario, trying to outsmart an opponent is not feasible (Vlebooks.com, 2022).

Game theory branches into economics to a reasonable extent. Buying and selling damaged goods is another example of a zero-sum game. If a buyer sends an item that is damaged, the unlucky customer will lose out by taking in damaged goods. However, the seller would profit from an item that did not hold any value (MasterClass, 2020). Additionally, stock options also apply as a zero-sum game. Receiving stock into a company could prove beneficial for either the company or the employee. If the future stock does well, the employee gains more value for his contract, and the company loses value to its employee. If the stock does poorly, the company gains because the value is less than a high salary. In the next chapter, cooperation will be added to game scenarios. The effect the addition would have on a strategy would be observed.

2.3 Cooperation and Defection

The previous chapter discusses how uninteresting games tend to be transparent. Each move has an absolute best strategy. Interesting games possess the ability for deception to take place. In this section, the research undertaken into cooperation and defection will be reviewed.

2.3a Non-Zero-Sum Games

A non-zero game is when a game has an absolute winner and loser. Non-zero games have the additional option for both players to be rewarded or punished. A social example that loosely follows this theory is cleaning a house with roommates. If both roommates participate in maintaining a clean house, both roommates mutually benefit despite doing work. The second scenario is when one roommate is lazy and does not clean. The lazy roommate benefits from having a clean house without any work, while the other roommate must maintain the house all by itself. After observing how the lazy roommate benefits from a lack of work, the second roommate could join them from a sense of jealousy. Now that both roommates are not cleaning, they suffer from a dirty house. While this scenario applies some of the principles of game theory, there are too many social dynamics that would alter the behaviour of the players. The main objective for a non-zero game is to win.

To properly understand the properties of the game theory, there needs to be a dilemma no matter what option a player chooses at the start. Essentially "no matter what choice a player (Vlebooks.com, 2022). When describing regret, it is worth recognising that it describes objective regret rather than moral. For example, choosing a different strategy could have attained more points. Figure 3 shows a general payoff matrix for cooperation and defection. When both players cooperate, there is a mutual reward for cooperation. When both players defect, there is a mutual punishment. The table highlights how beneficial it is for only one player to defect, with the loser who cooperated being described as the 'sucker's payoff'.

Player B's move

Player A's move		**cooperate**	**defect**
	cooperate	(CC, CC) Reward for mutual cooperation	(CD, DC) Sucker's payoff, and Temptation to defect
	defect	(DC, CD) Temptation to defect, and Sucker's payoff	(DD, DD) Punishment for mutual defection

Figure 3: A payoff matrix for Cooperation and defection

One example of a non-zero game is the Chicken game. The concept involves two players facing each other. It is easy to place both players into a car to get a sense of urgency and danger. When the game starts, both players will head toward each other till collision distance. If both players do not deviate out of the way, they will crash and suffer the consequences (see figure 4). If only one player moves out the way, the other player has one the round as he was not a "chicken". Lastly, if both players change paths, neither player wins (Economics, 2022). If the game is visioned from one player's perspective, the best scenario from best to worst is DC > CC > CD > DD. The best scenario would be for Player A to drive ahead without deviation with player B swerving. The second-best scenario is for both Player A and B to swerve since there is no loss on either side. The third best is for Player A to swerve while

Player B continuous forward. Even though the move counts as a loss, no collision allows player A not to be penalised too harshly. The most severe penalty is attributed to the game when there is a collision.

	Swerve	**Continue Straight**
Swerve	0, 0	-1, +1
Continue Straight	+1, -1	-2, -2

Figure 4: Payoff matrix for the chicken game

One of the most notorious games for game theory is the Prisoners Dilemma, devised by Merrill Flood and Melvin Dresher in 1950 (Kuhn, 2020). The scenario involves two partner criminals arrested for crimes they know they are guilty of committing. Luckily to their advantage, the prosecutors lack sufficient evidence to prosecute to the full extent on their own. Both prisoners are interrogated separately, knowing that their partner receives similar treatment. Each prisoner has two options; either snitch on their partner (defect) or remain silent (cooperate). If both prisoners cooperate, they will get charged a minor sentence of 1 year each (Investopedia, 2022). If one prisoner decides to defect, he gets to be released while their partner receives five years of jail time. If both prisoners decide to defect, they will receive two years of jail time.

The prisoner's dilemma could be viewed as not having a dilemma. Due to the possible penalty of cooperating, the safer option would be to defect regardless of the partner's decision. In the worst case for both situations, cooperating results in 5 years jail time, while defecting results in 2 years jail time. The order of scenarios for one prisoner is, DC > CC > DD > CD. The standard version of the prisoner's dilemma does not lead to other draw outs that could be applied to the real world. It is limited in its approach since there is the absolute best strategy when played for the first time. In the next sub-section, a look into various strategies when games such as the prisoner's dilemma are iterated.

2.3b Various Strategies through Iteration

The iterated prisoner's dilemma applies the same rules as the original but with the addition of numerous rounds with the same players. The players would know the past moves of their opponent. This game version can be played with a sentient being or a computer (Vlebooks.com, 2022).

- ALL-D (“Always Defect”)- For every iteration of the game played, the plan is always to choose defection. This strategy has its benefits because the player is incapable of being exploited. When faced with other more cooperative strategies, it has the advantage. Its weakness comes from facing itself.

- ALL-C ("Always Cooperate")- For every iteration of the game played, the plan for this strategy is always to cooperate. When faced with other strategies that aim to cooperate, it can benefit. However, exploitation is rampant when challenging a more ruthless strategy such as ALL-D.
- RAND ("Randomly choose between cooperation and defection")- mixing up the choice between the offers RAND roughly a median score. It bests the All-C strategy but would also struggle against an All-D strategy. The plus side is that it fairs well against itself.
- TFT ("Tit for Tat")- The strategy is to be cooperative initially. Unlike the other strategies, TFT can memorise previous rounds. From the second round and onwards, TFT replicates whatever its opponent chooses. Against All-C, TFT would be identical, ending with a tie. TFT only loses in the first round when facing All-D as it adapts.
- PAV("Pavlov')- The initial choice is to remain cooperative. PAV works by being reactive to punishment. If all is well, the current state remains, but it alternates to the other state if it is losing. This strategy plays better against All-D than All-C due to oscillating between the two states rather than remaining.

In the table (see figure 5), the strategies ALL-C, RAND and ALL-D are scored against each other. All-C scores the highest of 3.0 when played against itself. Nevertheless, a 1.5 and 0 from RAND and ALL-D bring the average score to 1.5. The RAND strategy, as predicted, scores a higher average than All-C. This is due to it being more competitive against All-D and All-C. Finally, All-D scores the highest average of a score of 3.0. Thanks to its uncompromising strategy, high scores such as 5.0 can be obtained from competing against ALL-C.

	ALL-C	RAND	ALL-D	Average
ALL-C	3.0	1.5	0.0	1.5
RAND	4.0	2.0	0.5	$2.16\bar{6}$
ALL-D	5.0	3.0	1.0	3.0

Figure 5: Table showing strategy averages

The TFT strategy was victorious in Axelrod's tournament. Axelrod's tournament saw multiple strategies competing against each other in multiple rounds. The victory made Axelrod compromise a list of what he believed a successful strategy looked like (Vlebooks.com, 2022). The four conclusions were:

- Be nice: In the second round of the tournament, many other strategies tried to emulate TFT. However, they would occasionally try to defect to catch everyone else out. The harshness of trying to be the first defection never proved successful among the 'nice' strategies.

- Don’t be envious: TFT is designed not to get greedy. It simply follows the path of others around it. The best it can do in a particular face-off is a draw. In summary, TFT seeks a win-win scenario for both strategies.
- Don’t be too clever: TFT is very transparent. Its intentions are clear but are still hard to beat. When Advanced algorithms try to outsmart TFT, the outcome looks familiar to randomness.
- Reciprocate: Cooperation is encouraged when coming up against TFT. Mean strategies are often punished for defection. It also has a forgiving nature to it. TFT and another strategy are both defecting; TFT would choose cooperation if the opponent returns to it.

2.3c Cooperation and Defection in Nature

Cooperation and defection are often studied in ecology between different organisms. Many organisms benefit from cooperation. For example, ants fiercely defend plants from other animals (mainly herbivores). To protect a plant from possible death, the ants receive sugar and protein (ScienceDaily, 2019). When the sugars on the plants become more scares, the ants would tend to fight off other animals more aggressively. If the ant has plenty of recourses for nutrition, the plant would be in danger of being consumed more quickly. Mammals have been shown to groom and hunt together in order to survive. The key motivator for cooperation between animals tends to be survival reasons rather than moral ones. There must be a mutual benefit.

The case of Wrasse is an example where defection is also a factor in behaviour. Wrasse fish follow and attach themselves to much larger fish. When food particles and parasites try to attack the larger fish, the Wrasse will eat it instead. The big fish gets to remain clean and protected from potential miniature threats. The Wrasse gets food and protection by assimilating with larger fish (the, 2019). The relationship is seen as mutualistic. Another species of fish called a Blenny complicates the otherwise healthy relationship. The Blenny looks very similar to the Wrasse (see figure 6) and has studied most of its mannerisms. The reason for this is to be able to trick a larger fish into mistaking it. When the Blenny enters proximity to a larger fish, rather than cleaning and protecting, it takes a bite from the tail fin of the larger fish. The Blenny successfully uses defection to seek personal benefit in nature.

Figure 6: Wrasse fish

In nature, the animals with the best strategy are often the group that survives. So, with game theory, simulations were done to see which strategy is the best for long term survival. Graph A shows (see figure 7) the population dynamics starting at an equal proportion rate of 0.2 without any noise. Graph B is the same but with 0.5 noise. In Graph an ALL-D has an initial rise in population, but by step 15, it starts to decline rapidly. As discussed, 'mean' strategies often start to fail when competing against fellow 'mean' players. ALL-C and RAND immediately decrease and do not see a rise as the number of generations continues. Both TFT and PAV fluctuate towards the end, with the winner being TFT. TFT came out the winner due to its better ability to handle defectors than PAV. PAV oscillates back to cooperation after copying a defection, which costs more points.

A noisy environment in this simulation is when a random fluctuation changes the choice of particular strategies for a round. In the presence of noise, the most adaptive strategy is more likely to succeed. Graph B mimics A. ALL-C, ALL-D and RAND end dying around where the generation gets to 25. The PAV strategy ends up taking up most of the population over TFT. Noise in the simulation meant that TFT could copy other opponents, which had a randomised outcome. Copying opponents now proved to be more costly because there is no strategic selection to an opponent selecting a particular option. PAV, however, chooses whether to cooperate or defect depending on the highest payoff choice. This ability to adapt to the best choice of survival gives the winning strategy more flexibility. In conclusion, TFT is the best strategy for survival when conditions are perfect, but PAV is a more viable solution in a more realistic world.

(a)

(B)

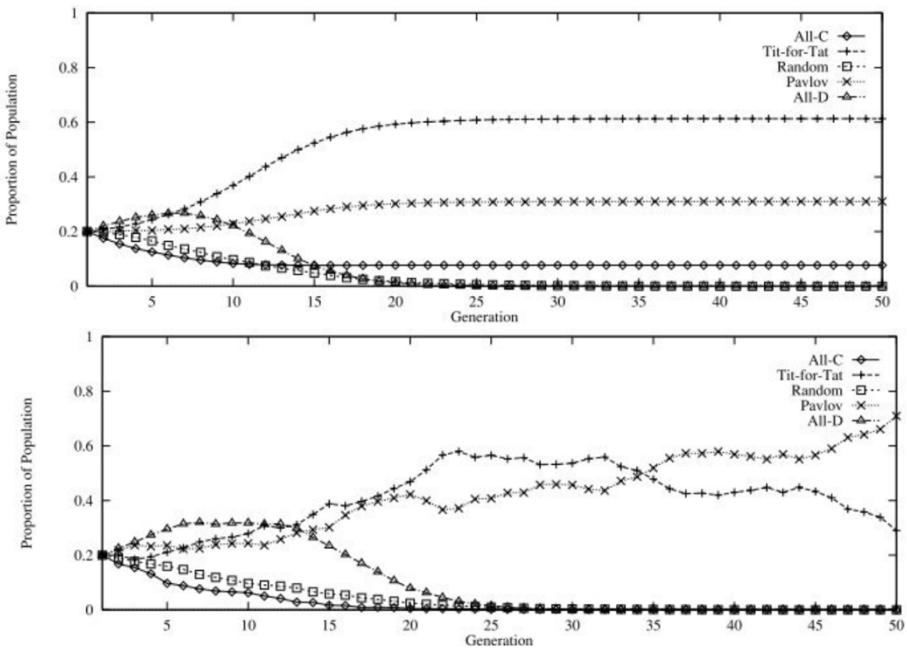

Figure 7: Population of different strategies

2.4 Simulating Evolutionary Game Theory

As previously mentioned in the literature, Robert Axelrod conducted a robin round tournament to determine which strategy would prove dominant. The tournament went for 200 rounds in which TFT succeeded. These theories were placed in simulations to determine which tactic dominated the population. Spatial games offered a viable solution to judging the population dynamic to understand which is more dominant between cooperation and defection. Cellular automata have the conditions for a population sample to be simulated for multiple rounds. Each cell having eight neigh rest neighbours gives an individual range to make a sensible choice as to which state is best for maximum payoff.
Studies have been conducted using Cellular automata (CA) to observe the evolution of cooperation in a controlled population. There have been variations that tweak the strategies and the rule set to observe the affect it has on the population. How memory helps determine strategy choice is an example. Additionally, one study investigated how cooperation would do under environmental conditions (Alonso, Fernandez and Fort, 2005). Novak and May proposed CA to solve evolutionary game theory with specific properties. Complicated strategies such as TFT, while disregarded due to CA only using unconditional strategies, helped shape the strategic choice of imitation. Each state is either cooperative or defective, and its future strategy is solely dependent on its neigh rest neighbours. They first tried a simplified version of the prisoner's dilemma. The punishment value was equated to the sucker's payoff. Since a high score signals success, the worst punishment would have the lowest score. Additions were added to the model to include randomness at various game stages. An issue arrived were games that adopted the simple PD strategy ultimately resulted in all cells adopting DE for maximum payoff.

4.4a Simulating non-Zero Games

The population dynamic between cooperative and defective agents constantly evolves. One study examined how the dynamic shifted when a fixed structured population (García-Victoria et al., 2022). Two games were simulated for 20,000 cycles. The first game was the Snowdrift game. While unique in theory, the snowdrift game shares similar core principles to the Chicken game discussed earlier in the literature. A population of 20 cell split of 50% was run using the software MeGoSim. Every simulation run also had a mutation probability of 0.01. Figure 8 shows the results once the simulation is over. Both CO and DE were successful in invading each other.

In conclusion, both states can coexist. Results change dramatically when the prisoner's dilemma is used instead. As shown in figure 9, DE remains dominant throughout the entire simulation. CO is unable to invade. With a considerable number of rounds being tested, it is safe to say the payoff matrix given to each game plays a huge role in the conflict between both states.

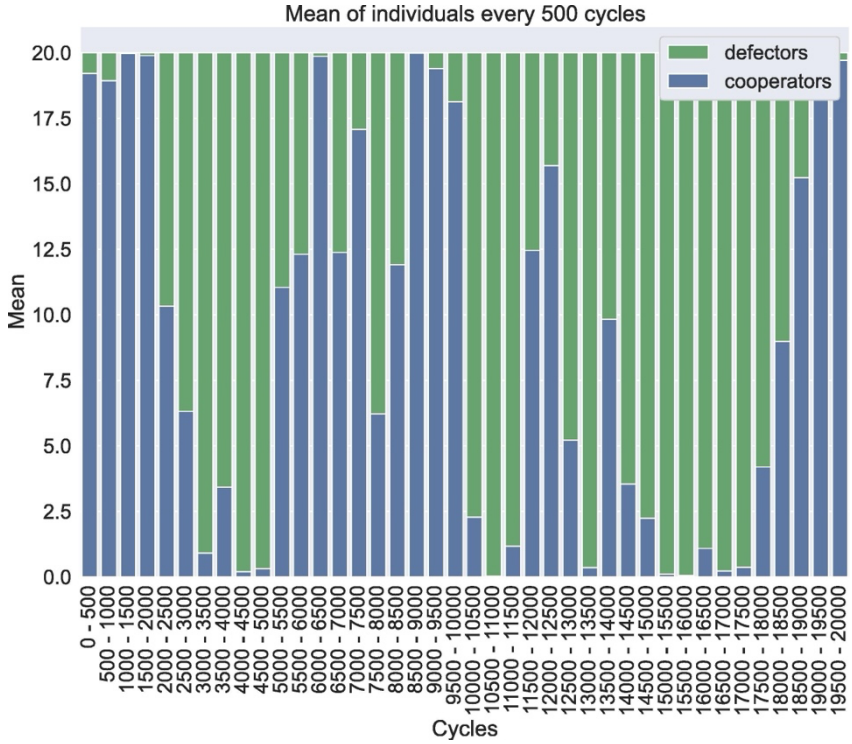

Figure 8: Mean cooperation and defection for the snowdrift game for 20,000 cycles

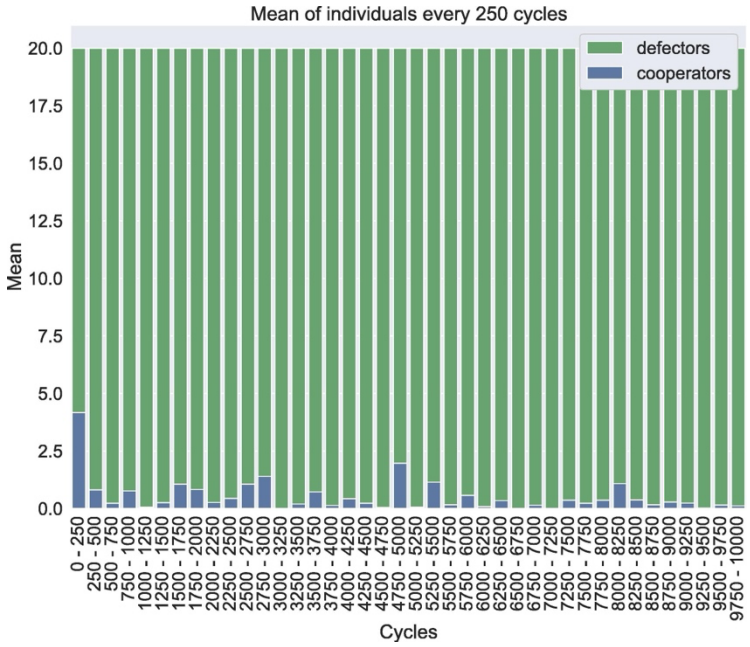

Figure 9: Mean cooperation and defection for the prisoner's dilemma for 20,000 cycles

2.4b The Prisoners Dilemma Game with Memory

One study looked at the possibility of reducing the number of strategies that would need to be computed individually for every round for the iterated Prisoner's Dilemma (IPDG) using Cellular Automata. The theory they tested to solve the problem was to base a agent's strategy on opponent's memory (Nakayama, 2013). All strategies were divided from being deterministic and un-deterministic. Deterministic strategies such as TFT require the memory of other opponents to decide its correct action. Discussed earlier in the literature, TFT only considers the most recent play, thus lacking the full information to accurately imitate. After three iterations, the last three moves should always reveal opponent's next move. Initial transient instructions are what influence strategies move until the rounds are completed. Any behaviour past the initial three moves is referred to as asymptotic behaviour.

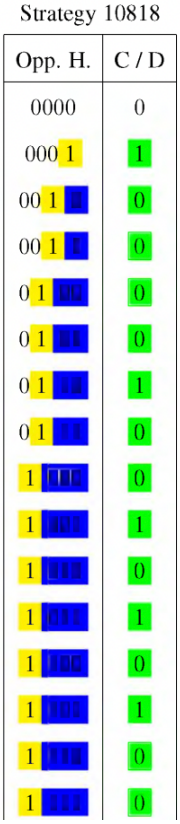

Figure 10 shows a table of an oppositions past moves corresponded with the state a player chooses. This table includes both asymptotic behaviour and transient. The able represents all the possible moves for every round. "0001" represents round 1 where "001X" represents round 2. The yellow indicator marks what number the current player should judge from. It could observe that as the rounds continue, the more possible combinations can be produced.

The table would continually expand until there were 15 bits in total. This unique bit number could then be used to describe a particular strategy when converting a number to binary. For example, the binary combination 0010101001000010 can be converted to the number 10818. Being able to produce a unique identifier meant that colours could be assigned to each strategy in the cellular automation. Since each bit could either be 0 or 1, there are 32,786 different possible strategies by calculating 2^{15}. The study had various strategies compete TFT. Due to the nature of TFT, the strategies still received low payoff scores.

Figure 10: Asymptotic and Transient memory strategy

A 2d lattice was simulated to represent how copying neighbouring strategies based on memory would lead to one dominant strategy. Figure 11 shows two simulations with an initial state, one round state and the result after 10 rounds. A random allocation of initial strategies was given to each cell. Applying Moore's neighbourhood, the payoff score over history is compared to determine the most appropriate strategy to copy. The second simulation circled in black represents ALL-D or id 0. The 'mean' approach can be seen to overrun over strategies after round 0ne. The initial white colour is the strategy that dominates in the closing state.

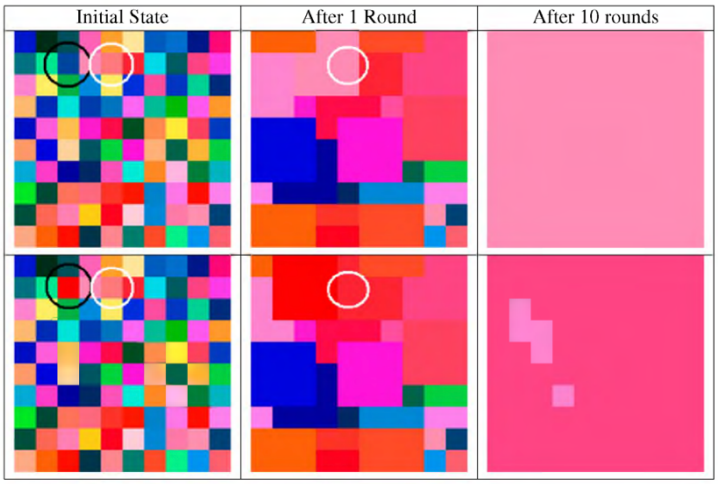

Figure 11: A demonstration of how neighbours affect neighbours

2.4c Evolution under Environmental Pressure

The new model used both the von Neumann neighbourhood of four and Moor's eight neighbourhood. The grid size ranged from 50 x 50 to 500 x 500. The rationale of this model is that adopting the most successful neighbours state or strategy is not enough in nature. Better does not mean good enough; a better survivalist may still lack the resources to survive in nature. Therefore, copying a prosperous neighbour may still be futile. Introducing a minimum score before death is more feasible from an evolutionary viewpoint. Two variations of the model were considered:

- Simplest variant: A cell copies its most successful neighbour state. If that neighbour's score is below the minimum threshold, the cell(player) gets assigned a "probability p of adopting the opposite state" (Alonso, Fernandez and Fort, 2005).
- Variant 1 and the death of organisms: The rules are the same as the first variation except when the most successful neighbour's score is below the minimum. In this instance, the player dies. A probability is assigned of 1-p, leaving a blank cell. In the next iteration, that empty cell copies the most successful neighbour with 1-p. Alternatively, if a cell is surrounded by fellow empty cells, it remains empty.

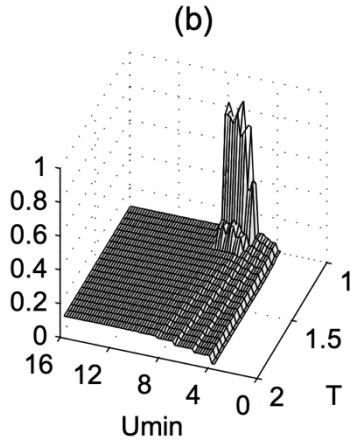

Figure 12: Asymptotic frequency of co-operators for the simplest model, p=0.1, z=8 neighbours

The system variations got tested, and both saw all the states reach a cooperative state after a short period. The system was tested using different grid sizes and a different number of neighbours. To produce a final conclusive set, the study decided to take an average of all the different possible conditions. The standard deviation only resulted in 7 per cent making the results quite reliable. Figure 12 shows an asymptotic plot of the first variant to show how the percentage of cooperation in the population changes as the minimum threshold varies along with the temptation parameter. CO reaches low numbers when the minimum threshold is equal to the punishment of 0.5. Representation is shown with a stepladder on the right border. When the threshold is around 5-8 and the temptation parameter is high, there is a high peak of cooperation. After that point, a higher minimum threshold results in a plateau of around 0.2 for CO. The study tested other variations and models, all with relatively similar findings.

Evolutionary base survival is one of many attempts at testing cooperation and defection using cellular automata. Through surveying the literature, there is a gap in memory use. The common practice involves recording the most recent state of opposing neighbours to adapt in the next iteration. Some strategies only reveal themselves after a couple of rounds to spot a pattern; therefore, only considering the most current state leaves players unsuspecting about the play. This thesis will look into how tracking the states of previous generations would affect the population dynamic between co-operators and defectors after multiple rounds. Additionally, most experiments use the most successful neigh rest neighbour as a shining as the player to mimic. It is undiscovered if adopting strategies such as imitating a player with an average score could assist in the success of cooperative agents.

Chapter 3

Design

With the literature review surveyed, the design process of building the model could commence. This chapter will show how certain concepts from the literature could translate into development. There would be a transparent process of how the prototype being produced would help achieve the aims and objectives. Additionally, any potential hurdles should be accounted for before starting the implementation; that way, an idea of which objectives would be successfully met.

3.1 Requirements and Workflow

The table below shows specific deliverables that should be met when undertaking the development. Some of the requirements are optional and only serve to enrich the product presentation. The essential requirements with high priority need to be completed for the model to have the capabilities of meeting the aims and objectives.

ID	Requirements	Priority
F1	Code a cellular automaton model that shows two states of CO and DE	High
F2	The grid dimensions should be adjustable to select the most appropriate one.	Low
F3	The game should run for a set of rounds before coming to an end state. The end state should be reached when the simulation reaches a stable state.	High.
F4	The GUI should display a counter of the population levels and the number of rounds.	Medium
F5	A standard algorithm is implemented to select the most successful neighbour.	High
F6	An algorithm where imitation depends on the neighbour closest to the average payoff score.	High
F7	Imitation algorithm implemented where an agent considers the payoff score from a collection of past iterations in deciding the best to imitate.	High

F8	Including an imitation algorithm that selects a random neighbour to imitate regardless of the payoff score.	High
F9	Produce a plot showing how the population evolves over time after the game reaches a stop state.	High
F10	Introduce buttons to pause and play the simulation. There should also be a speed controller.	Low
F11	The simulation should have a mutation probability that chooses the opposite of the chosen strategy.	Medium

Table 1 Functional requirements of model

3.2 Designing the Model

Constructing the lattice

Conditions must be selected and consistent for the proposed model to complete its function. In this thesis, the chosen model is Cellular Automata (CA). A one-dimensional grid would lack the properties for local environmental interactions. In a 2d lattice, the most common type of neighbourhood is either Von Neuman's or Moor's Neighbourhood. Moor's neighbourhood is selected, with eight local neighbours present (see figure 13). This decision was made due to more density being achieved, which would result in more accurate imitations based on the local environment. In nature, a subject is responsible for 360° of its surroundings.

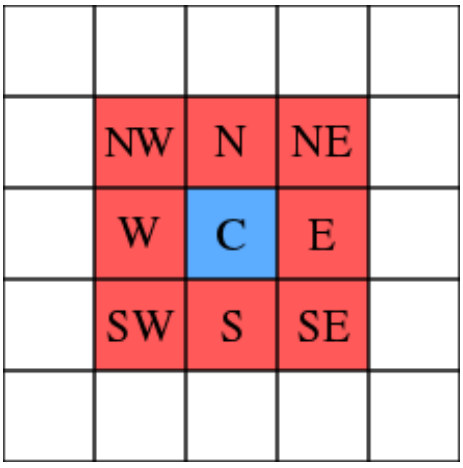

Figure 13: Moore Neighbourhood

The model would have a finite parameter size. If the grid were infinite, there would not be a fixed population size to analyse the population evolution. While the grid dimensions must be fixed, the neighbourhood regions can be wrapped. Wrapping the 2d grid ensures there is no disadvantage to either state on starting position. The model would also include a mutation

probability to replicate a noisy environment. A perfect simulation would be more challenging to contextualise for the broader meaning.

The model process

Figure 14 shows a prototype of the various stages of the model. Bias can be eliminated by the initial states being randomly assigned. A manual assignment would increase the likelihood for patterns to be formed increases. There could also be a population that does not represent a 0.5 split, given a margin of error. Such individualism for a particular simulation makes spotting trends and evaluation less reliable. The next stage involves local neighbours competing in either a prisoner's dilemma game or the Chicken game. The payoff matrix should produce a sum score for every agent. The algorithm used to decide the states each agent would imitate in the next iteration is the main testing point of this thesis. For functionality in the system, the first imitation method should be to imitate the most successful neighbour. There is previous research which validates the following results in the stated conditions. The model should then simulate for a set number of rounds until the population reaches a stable state. Finally, plots could be produced showing the history of the cooperative population.

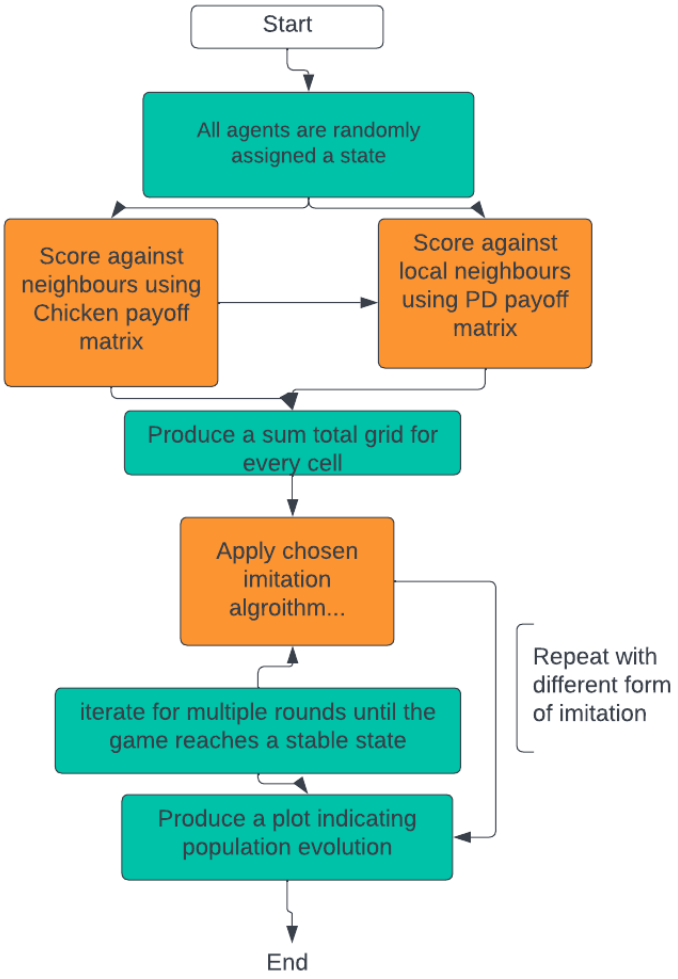

Figure 14: model process

3.3 Payoff Matrixes

For the conflict between cooperators and defectors to be observed, there first needs to be a simulated game between local neighbours to calculate an agent's total score. Two games from the literature would be used. The first is an iterated version of the chicken game, and the second is an iterated version of the Prisoner's Dilemma. Each version has previous test results that could be used as a successful test for the model being undertaken. The aim of choosing two games is to make it easier to evaluate if the choice of imitation plays a role.

The theory of the Chicken game following scenario order is DC > CC > CD > DD. The worst-case scenario is mutual defection. Since the penalty for choosing cooperation is less severe in this game than in the Prisoner's Dilemma, an iterated version is expected to favour cooperative agents over defectors. After observing previous experiment figures, an appropriate payoff matrix is produced in Table 2 (García-Victoria et al., 2022). The scores assigned to each game scenario are as follows: DC=4, CC=3, CD=1 and DD=0. The justification for having DC=4 and CC=3 is that cooperation is a solid strategic choice. Not crashing into an opponent is the more rational choice for a player who values survival. While the player does not win, there is not too much loss allowing a second try. Defecting by not swerving is still rewarded but only by a small deficit.

	CO	DE
CO	(3,3)	(1,4)
DE	(4,1)	(0,0)

Table 2: Payoff matrix for the Chicken game

The prisoner's dilemma scenario order is DC > CC > DD> CD. Unlike the Chicken game, the worst-case scenario is Cooperating while an opponent is defecting. Amongst the basic strategies, it was proven that ALL-D would be the most dominant strategy choice in an iterated game version. From the theory, it could be assembled that this model would favour defective agents. A payoff matrix shown in Table 3 is produced. The scores assigned to each scenario are as follows, DC=5, CC=3, DD=1 and CD=0. A gap of two points is given between DC and CC. In the context of the prisoner's dilemma, having the option to be released immediately should be represented with a reward strongly ahead of CC. Cooperation has less impact on the score compared to the Chicken game.

	CO	DE
CO	(3,3)	(0,5)
DE	(5,0)	(1,1)

Table 3: Payoff matrix for the Prisoners dilemma

3.4 Imitation Algorithms

Memory

One of the imitation techniques that is going to be observed is how the inclusion of memory affects the conflict between the two states. The completion of this imitation technique should ensure that F8 is completed. The idea is for a player to select a neighbour to imitate based on the highest scoring neighbour. The score for each neighbour is going to be based on previous iterations as well as the present.

Figure 15: The memory imitation algorithm

Figure 15 shows the algorithm that would be attempted in the actual model. The pseudo-code could be adapted into any language of choice. The theory from the literature suggests that three moves are all that are required to be able to predict an opponent's strategy. This principle was adapted for this algorithm to use three iterations of scores to decide on the best neighbour to imitate. The algorithm is broken into three different scenarios. Round one and

two have a different set of instructions compared to round one due to there being no existence of a second and third datasets. The set of instructions presented in the figure from the third round onwards is what the simulation would consistently use. After every round, the oldest set of payoff data is disregarded to shift the scores from the most current three results.

Average

A second imitation approach being tested is to imitate based on the neighbour closest to the average score amongst all local participants. Traditionally in cellular automation models for spatial games, copying the neighbour with the highest score offers the best chance for maximum payoff. However, in most cases, a player's success is directly correlated to the states of their neighbours. The success tends to drop when neighbours imitate the same strategy. For example, the ALL-D strategy works best against ALL-C, but the score suffers when faced with itself.

Selecting a neighbour that maintains a score closest to the average amongst the local populace could prove insightful. Average often represents stability. Due to the push for stability, cooperative agents could benefit when the main goal is survival across the board compared to selfishly seeking the highest possible score. Figure 16 describes the algorithm needed to select the average neighbour. Consistency amongst all the imitation types can be achieved when an agent's score is not included in the calculations.

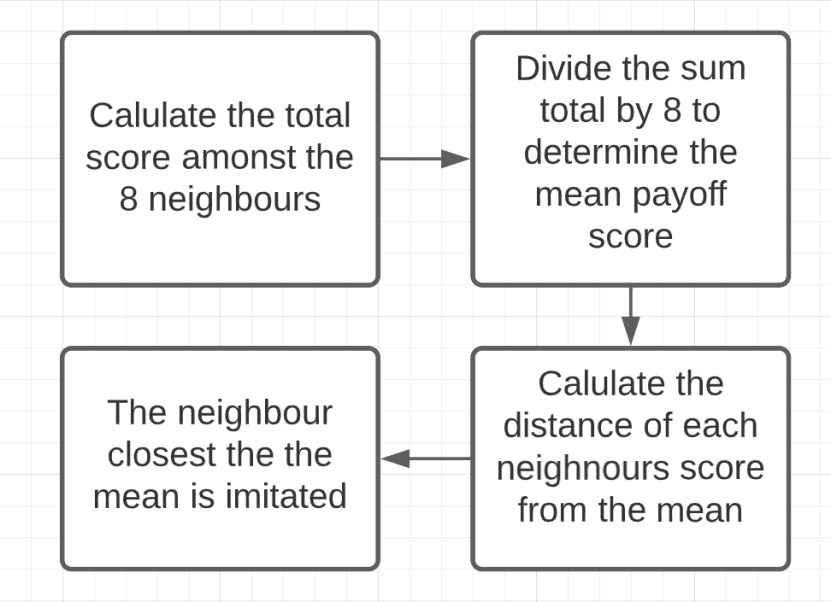

Figure 16: Average algorithm

Randomness

The last imitation tactic being explored is randomness. The aim is for a cell to select a neighbour regardless of its payoff score. The key to testing this imitation strategy is adjusting the population proportions. A population initially dominated by cooperators may remain cooperative given the higher probability of similar neighbours. Alternatively, the population could reach a state of mutual coexistence regardless of initial properties.

3.5 Data and Ethical issues

The completion of the simulation would produce a visualisation of the population in various stages. To properly analyse the affect imitation has on the conflict between C and D, numerical data must be extracted to be analysed. This Quantitative approach allows plots to be produced on software such as mahlab and R. Plots such as histograms and line graph equipped with the correct data should aim to answer the research question. Possible data points to be extracted are:

- The round number after every iteration.
- The mutation rate.
- The initial invasion percentages.
- The number of Co-operators present in each round.
- The number of Defectors present in each round.

To fully understand the effect of imitation, selecting the highest scoring neighbour should be used in reference. The data from this experiment is already available in past papers. While the conditions for the experiments are listed, there is a high chance of variance due to different model builds. All data and results would be primarily collected in order to maintain scientific integrity through consistency. All possible variations would be retested using the new model.

Chapter 4

Implementation

In this chapter, the implementation of the proposed model will be broken down and described. The software used to code this project will be justified in this section. Code would be used to show how certain functions were created in building the initial configurations of a cellular automaton model. The chapter's primary focus is integrating the imitation types discussed in the previous chapter. Successful completion of each imitation strategy should show the results of a completed simulation to help deliver the objectives of this thesis.

4.1 Setting up the Model

The chosen language of choice to construct the model was Java. The choice was made due to its well-constructed design patterns. Java was also chosen due to multithreading; it was decided early in construction to try to create two distinct stages of the model. The first being simulation and the second being graphing. Eclipse IDE was the chosen software to build the project. With Eclipse, debugging code becomes less of a hindrance to workflow. There are plenty of pre-installed libraries to access features such as Jframes. The maven plugin also made installing any additional libraries required seamless.

The project's first stage was to code a functioning grid that represented a visualisation of a 2d array. A Java GUI was used to display a window on run time. The size of the array was made adjustable to make testing more straightforward. Requirement F1 meant that each cell should have two possible states, CO and DE. In representing the two states, a 2d array type was created to colour where green represented CO and red represented DE. No manual input was required for this model since every cell randomly gets assigned to either state. The population proportion was also adjusted to test the invasion theory. Figure 17 shows how the grid loads up with the initial states assigned.

Figure 17: Initial grid

The application needed buttons and counters to meet F4. At the bottom of the grid, features such as a round counter and cooperator counter were placed. Play and Pause features were a valuable component to include to produce screenshots. The final component added was a speed slider. This allowed the simulation to run at a pace where every step was visible. When slid to the other side, the simulation could complete in a short period. These two options made testing different imitation types more manageable.

Global configurations

The role imitation plays in the conflict between CO and DE is the variable that should be tested. All the other factors that produce the model needed consistency for systematic evaluation. The first condition was the grid dimensions. The aim was to find a balance, so the cells were not too small to witness individual sections. However, a small population could narrow the results to a lack of broader context. The total population was set to 18,260 agents. The size chosen made the simulation be viewed as multiple clusters rather than individual cells. This was made possible by setting the resolution variable to 6. Figure 17 shows how a resolution of 6 scales in the given GUI.

```
private int resolution = 6;
```

In the previous chapter, it was decided that the model must include a mutation probability. In the literature review, typical mutation values are set at 0.01 or 1%. While a noisy environment is representative of nature, studying each imitation effect on the conflict becomes more unreliable due to the randomness. The mutation rate for this model was set to 0.0025 or 0.25%. Figure 18 shows the code used to perform a mutation. The rand generator chooses a number from 0-400. If the number 0 is selected, a mutation will occur by switching the chosen imitated state.

```
int mutation = rand.nextInt(400);
if(mutation == 0) {
    if(newColour == green) {
        newColour = red;
    }
    else if (newColour == red) {
        newColour = green;
    }

}
//System.out.println("x: "+x+" y:"+

return newColour;
```

Figure 18: mutation code

After initial testing, it was concluded that a reasonable number of rounds would be 500. When the simulation ran for a shorter period, the states did not fully evolve to reach a stable state resulting in missing data. At the other end of the spectrum, simulations that ran for 1000 rounds or more reached stable states between the populations far too soon. Five hundred rounds were the suited middle ground between the two.

Games and testing strategy

Following the outline created in the design chapter, the two games being played by local neighbours were the iterated Chicken game and prisoners dilemma. A scoring method had to be created within the GUI class to implement the two games. The purpose of the class was to loop through every neighbour and determine the states each had. For example, the selected cell may be green, and a neighbour may be red.; this combination would register as CD resulting in a score associated with that scenario. There are two combinations for each variable to represent each game's interchangeability. Figure 19 shows the algorithm used on every cell for every round. The payoff matrix figures were used from the design process. To ensure every cell has exactly eight neighbours. Cells on edge used a modulus equation to ensure a cell on the left edge registers a cell on the right side as its neighbour.

```
private int score(Color[][] grid, int x, int y) {
    //standard PD matrix
    //int DC=5, CC=3, DD=1, CD=0;
    //Chicken game
    int DC=4, CC=3, DD=0, CD=1;
    int score = 0;

    for (int i = -1; i<2; i++) {
        for(int j=-1; j<2; j++) {
            if(!(i==0 & j==0)) {
                int cols = (x+i + col) % col;
                int rows = (y+j + row) % row;

                //CC
                if(grid[x][y] == green & grid[cols][rows]== green) {
                    score= score + CC;
                }
                //CD
                else if(grid[x][y] == green & grid[cols][rows]== red) {
                    score= score + CD;
                }
                //DD
                else if(grid[x][y] == red & grid[cols][rows]== red) {
                    score= score + DD;
                }
                //DC
                else if(grid[x][y] == red & grid[cols][rows]== green) {
                    score= score + DC;
                }
            }
        }
    }
    return score;
}
```

Figure 19: Scoring algorithm

Before implementing the different imitation algorithms, the starting populations had to be decided and fixed. The ratio between states must represent the theoretical principles of each game. For the prisoner's dilemma (PD), DE is the dominant strategy backed by Axelrod's round-robin tournament. Therefore, the initial population is 95% CO and 5% DE. The exact ratio would not be precise due to randomisation being used but consistently around the mark. The test would be to observe if defectors could successfully invade the initial dominant cooperative region. The test for CO would be its ability to resist the invasion attempt.

The chicken game has different behaviours to the PD. The literature proves that cooperation and defection can both invade each other in different instances. Even with the existence of mutual coexistence, the favoured strategy is CO. To test the success of each state; the initial population would be set to 1/3 CO and 2/3 DE. When observing the results, the aim of the tests would check to see if cooperation were able to succeed and dominate the defective agents. Now that all the conditions were set and fixed. Each imitation method would be changed individually so it is clear the effect each method has on the population.

Test number	What to test
1	Imitating the most successful neighbour using the Chicken game
2	Imitating the most successful neighbour using the prisoner's dilemma
3	Imitating based on memory using the Chicken game
4	Imitating based on memory using the prisoner's dilemma
5	Imitating the average neighbour using the Chicken game
6	Imitating the average using the prisoner's dilemma
7	Imitating a random neighbour using the Chicken game
8	Imitating a random neighbour using the prisoner's dilemma

Table 4 Different tests to implement

4.2 Imitating the Most Successful Neighbour

The starting imitation strategy to implement involved copying the most successful neighbour (MSN). Since the strategy has been an approach used in literature, it was essential to test it out as the standard. After an entire grid had played a round and collected scores, the next step was for each grid to go through the 'standardImitate' method. Once a cell had passed through the method, a new cell colour for the next iteration would be produced.

Figure 20 shows the code used inside the 'standardimitate' method. Every time the method is called, an empty 'newColour' variable is created, followed by the largest score (LS) being set to 0. Each neighbour then gets compared against the current LS. If the neighbour has a score higher than LS, its colour gets set as the 'newColour' variable. After all the neighbours have been looped, a new colour should be submitted to the array. The implementation of this algorithm had to be tested by producing print statements of random cell states before and after rounds. The simulation ran for both games when the method was considered complete.

```
private Color standardImitate(Color[][] grid, int x, int y) {
    Color newColour = grid[x][y];
    //Largest score
    int LS = 0;
    int mutation = rand.nextInt(400);//mutation seed
    //iterate through each neighbour and set the current highest score as the new colour
    for (int i = -1; i<2; i++) {
         for(int j=-1; j<2; j++) {
             if(!(i ==0 && j==0)) {
                 int cols = (x+i + col) % col;
                 int rows = (y+j + row) % row;

                 if(activeGridScore[cols][rows] > LS) {
                     LS = activeGridScore[cols][rows];
                     newColour = grid[cols][rows];
                 }
             }
         }
    }
    //if mutation occurs
    if(mutation == 0) {
        if(newColour == green) {
            newColour = red;
        }
        else if (newColour == red) {
            newColour = green;
        }

    }

    //System.out.println("x: "+x+" y:"+y+"highest score"+LS+" Colour:"+newColour);
    return newColour;
}
```

Figure 20: Implementation of MSN imitation algorithm

Figures 21 and 22 show the successful completion of test 1. In the histogram, the cooperative population starts at 60% for the first cycle. The percentage rises, however, for the remaining cycles with values around 75%. This consistency produces a long-term average of 74% for CO. While it is conclusive that CO is the more dominant strategy in the population, the screenshots show how coexistence between the two states occurs. defective agents can invade cooperative regions, and cooperative regions can invade defective regions. These regions are evenly spread across the grid.

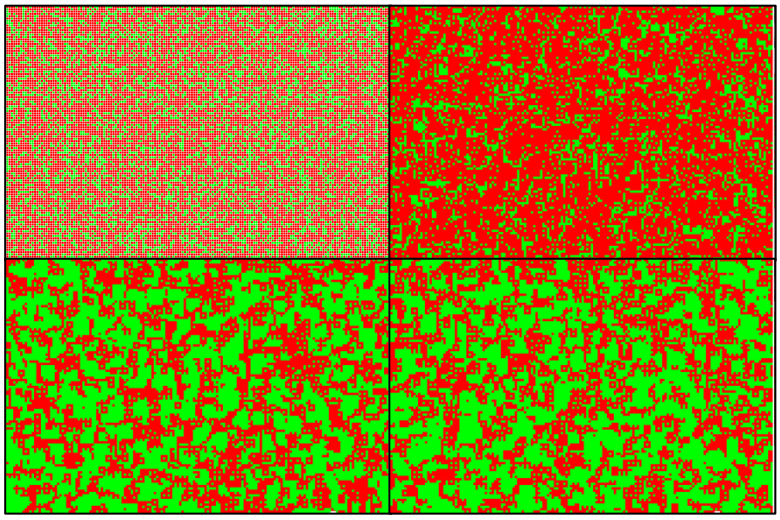

Figure 21: (Test 1) Simulation of MSN imitation for the Chicken game showing rounds 0,1,250 and 500. u=0.0025, initial population CO=1/3 DE=2/3

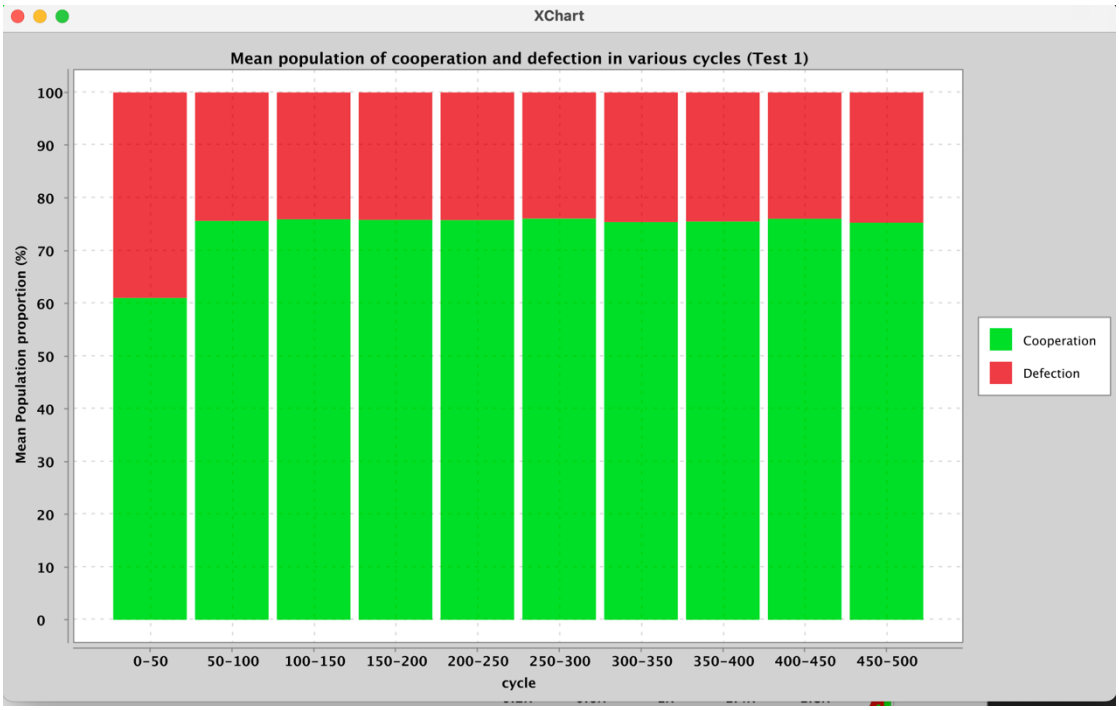

Figure 22: Histogram plot of test 1 showing the mean population dynamic per 50 cycles for 500 rounds. The first 2 rounds are removed to avoid the transient effects of initial configuration resulting in a long-term average of **74**% for CO.

Figures 23 and 24 show the successful implementation of test two. In cycles 0-50 in the histogram, the CO holds around 3% of the population. For the remaining cycles, the population remains DE dominant with CO below 1%. A long-term average of 0.9% represents this trend. In this experiment, it could be concluded that cooperative agents cannot resist defective agents from invading. The screenshots were gathered to confirm the data produced. Round 250 and 500 show that the CO is effectively wiped out; the only existence of CO comes from mutations in the Populus.

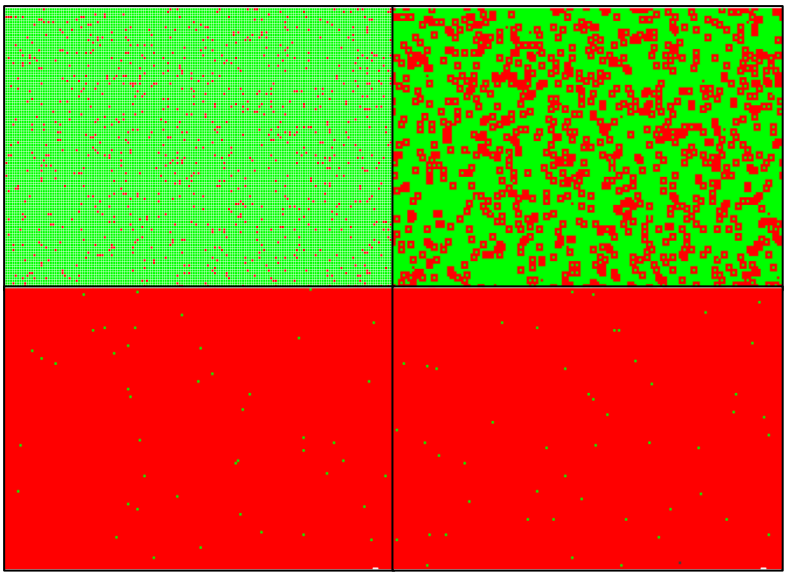

Figure 24: Histogram plot of test 2 showing the mean population dynamic per 50 cycles for 500 rounds. The first 2 rounds are removed to avoid the transient effects of initial configuration resulting in a long-term average of **0.9**% for CO.

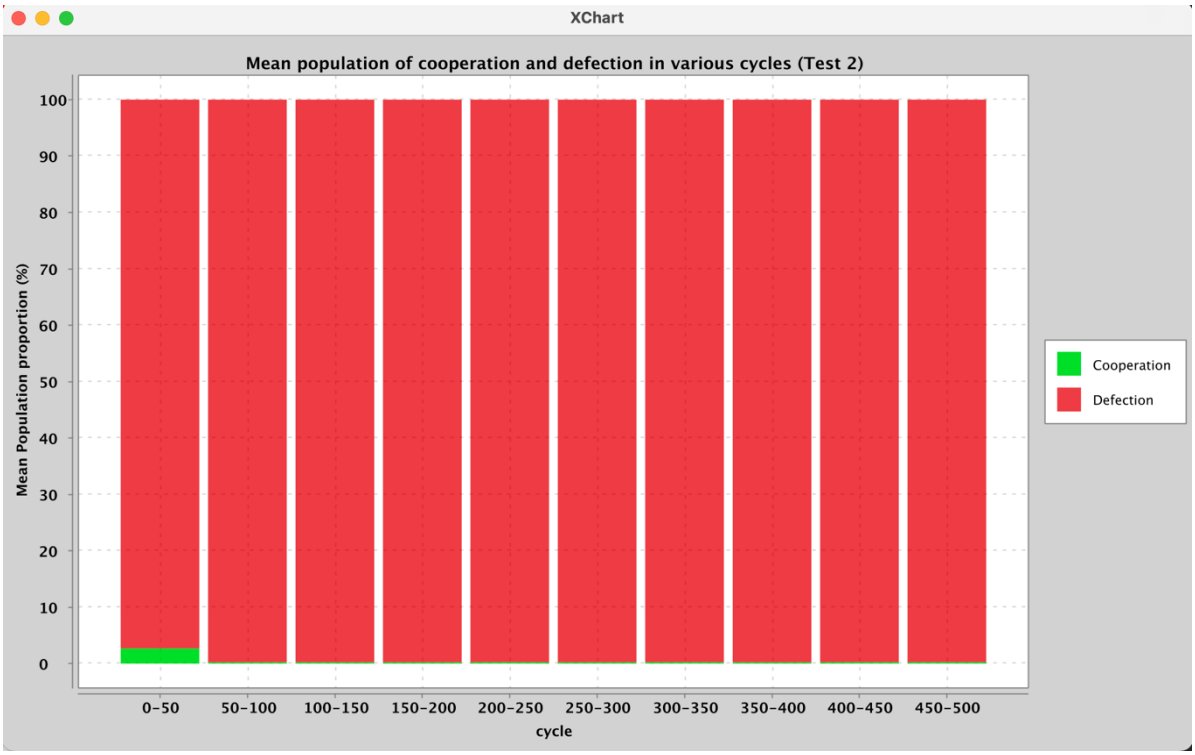

Figure 23: (Test 2) MSN imitation for the prisoner's dilemma showing rounds 0,1,250 and 500, u=0.0025, initial population CO=95%, DE=5%

4.3 Imitating based on Memory

Imitating the most successful neighbour is only one type of method. Planned out in the design chapter, considering the last three payoff scores could result in different population dynamics. The algorithm used in chapter 4.2 was able to be used again. This method would still incorporate searching for the neighbour with the highest payoff score to imitate. The challenge was creating a 2d array of scores that summed up the three iterations.

The first stage was to create multiple array lists of type integers. The names of each array were called 'activeGridScore', 'oneStepGridScore' and 'twoStepGridScore'. The values were each set to 0 before the simulation started. Figure 25 shows a section of code that makes up the algorithm to complete the imitation. The first step was to create a new array called 'sumGridScore'. The purpose of this variable is to be the 2d array that contains the total sum for every cell. Null pointer errors were avoided by splitting the algorithm into three different sections, representing round one, round two and round three onwards.

The chosen game is played in round one, and each cell produces a total payoff score. Since there has only been one round, the array ' activeGridScore' is set to the 'sumGridScore'. This process follows the exact instructions used in a standard imitation. For round two, the scores from round one need to be shifted to the 'oneStepGridScore'. Once the line of code is complete, the 'activeGridScore' can be replaced with a round two set of scores. Both arrays get added together to form a sum. The process repeats itself for round three. The only change is the addition of shifting the 'onesStepGridScore' to 'twoStepGridScore'.
The implementation of this algorithm was successful from a theoretical standpoint. The pseudo-code model from the design chapter was easily transferrable. Print statements were used to test whether the payoff scores were representative of three iterations. The code was then ready to be tested with the two games.

```
int sumGridScore[][] = new int[col][row];
//Compute the scores of all the cells
//For Round 1
if(rounds == 1) {
    for(int i=0; i<col; i++) {
        for(int j=0; j<row; j++) {
            int sumScore = score(grid,i,j);
            activeGridScore[i][j] = sumScore;
            sumGridScore[i][j] = activeGridScore[i][j];//Grid score to compare
            //System.out.println("i:"+i+"j:"+j+" "+sumGridScore[i][j]);
        }
    }
}//For Round 2
else if(rounds == 2) {
    for(int i=0; i<col; i++) {
        for(int j=0; j<row; j++) {
            int sumScore = score(grid,i,j);
            oneStepGridScore[i][j] = activeGridScore[i][j];
            activeGridScore[i][j] = sumScore;
            sumGridScore[i][j] = activeGridScore[i][j] + oneStepGridScore[i][j];
            //System.out.println("i:"+i+"j:"+j+" "+sumGridScore[i][j]);
        }
    }
}//For round 3 and onwards
else {
    for(int i=0; i<col; i++) {
        for(int j=0; j<row; j++) {
            int sumScore = score(grid,i,j);
            twoStepGridScore[i][j] = oneStepGridScore[i][j];
            oneStepGridScore[i][j] = activeGridScore[i][j];
            activeGridScore[i][j] = sumScore;
            sumGridScore[i][j] = activeGridScore[i][j] + oneStepGridScore[i][j] + twoStepGridScore[i][j];
            //System.out.println("i:"+i+"j:"+j+" "+sumGridScore[i][j]);
        }
    }
}
```

Figure 25: Memory imitation algorithm

Figures 26 and 27 show the successful implantation of Test 3. In the histogram, CO holds around 75% of the population. Once the simulation reaches 50 rounds, the average population share rises to a high 90% value. Stability continues throughout the cycle, with DE unable to push on cooperative domination. In the end, the long-term average of CO stood at 96%. While the overall population does not shift, constant invasions are attempted by defective agents. The screenshots show that small sections of defective agents are newly created during simulation. While both can invade each other, DE never seeks to gain a majority.

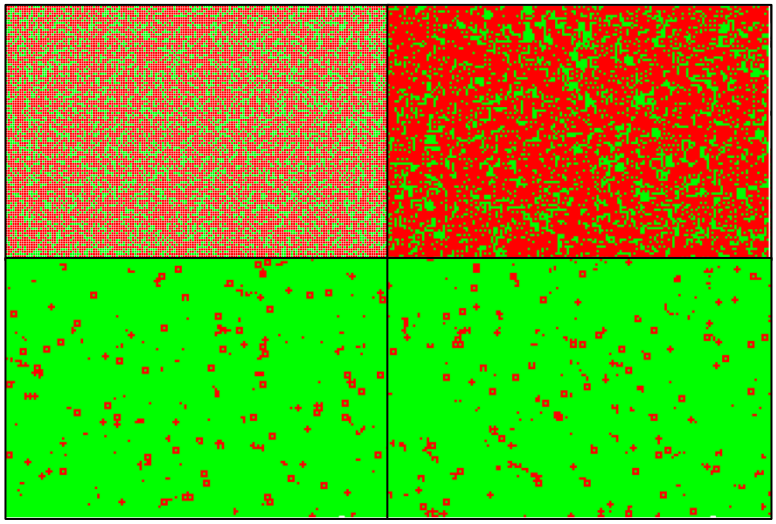

Figure 26: (Test 3) Simulation of imitation using memory for the Chicken game showing rounds 0,1,250 and 500. u=0.0025, initial population CO=1/3 DE=2/3

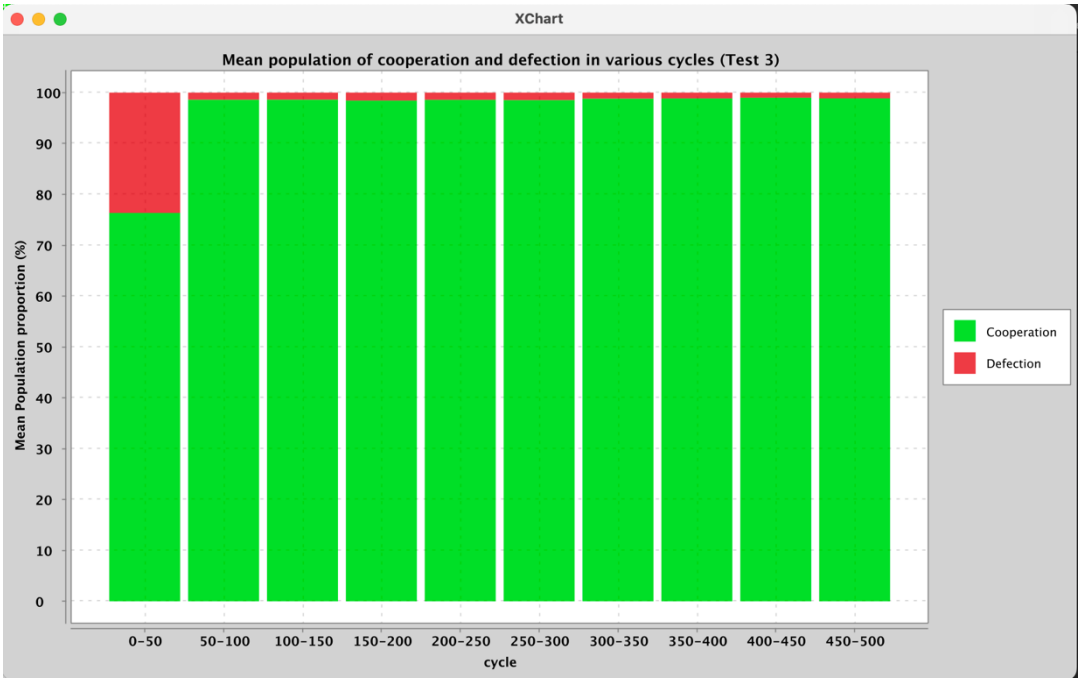

Figure 27: Histogram plot of test 3 showing the mean population dynamic per 50 cycles for 500 rounds. The first 2 rounds are removed to avoid the transient effects of initial configuration resulting in a long-term average of **96**% for CO.

Figures 28 and 29 show the successful implementation of test 4. The histogram shows that this test has coexistence between the two states. The first cycle has the CO at 10%. Throughout the cycles, the percentage share slowly increases in favour of DE. By cycles 450-500, the amount of CO reaches below 5%. It can only be assumed that more cooperative regions would be invaded if the simulation continued. The screenshots show the forever small amounts of cooperative agents surviving and invading defective regions. By round 500, more of the population is defective, despite small clusters of CO being formed.

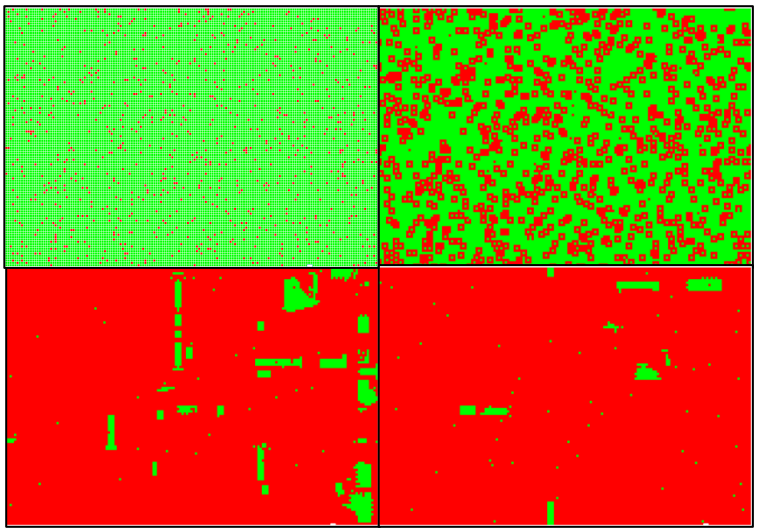

Figure 28: (Test 4) Memory imitation for the prisoner's dilemma showing rounds 0,1,250 and 500, u=0.0025, initial population CO=95%, DE=5%

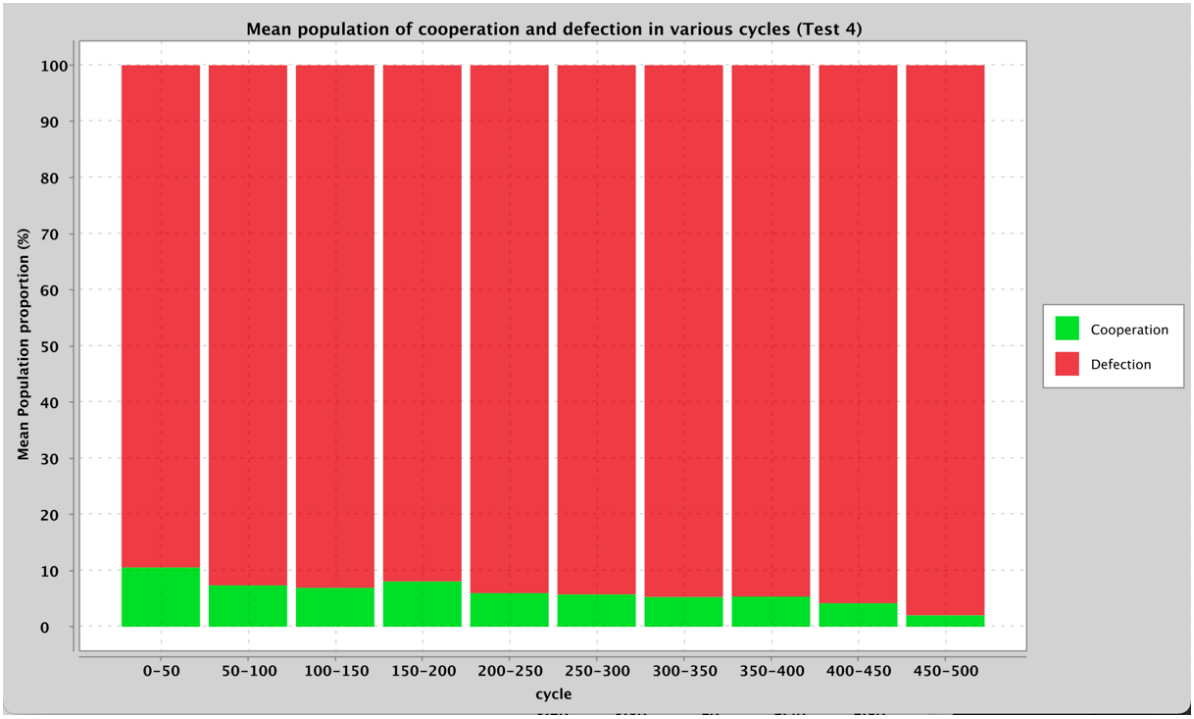

Figure 29: Histogram plot of test 4 showing the mean population dynamic per 50 cycles for 500 rounds. The first 2 rounds are removed to avoid the transient effects of initial configuration resulting in a long-term average of **6**% for CO.

4.4 Imitating the Average Neighbour

The following imitation tested the effect of copying the average. After every round, all cells scan all the neighbours. The neighbours that are the closest to the average payoff score get imitated.

Figure 30 shows the attempted mean algorithm implemented. Three variables were needed in the method. The TS variable means the total score, where all the neighbours' scores added and placed. MS means the mean score, which gets calculated later in the code. The final variable is 'smallestDistance', which initially gets set to 100 so every neighbour can replace it with a new value. With the variables declared, the first step was calculating the total payoff score. The neighbour's score was iterated and added to produce a final TS.

The mean score got calculated by dividing the TS by eight. The following calculation involved reiterating through every neighbour again. The distance between its payoff score

and the average for each neighbour was calculated. If the distance was less than the lowest distance currently set, that neighbour gets set as the neighbour to imitate. Once every neighbour is iterated, a new imitator is selected for the current cell. The algorithm could then be tested using the two games.

```
int TS = 0;
//Mean score
int MS = 0;
int smallestDistance= 100;
//mutation
int mutation = rand.nextInt(400);

for (int i = -1; i<2; i++) {
     for(int j=-1; j<2; j++) {
          if(!(i ==0 && j==0)) {
               int cols = (x+i + col) % col;
               int rows = (y+j + row) % row;

               TS = TS + activeGridScore[cols][rows];
          }

     }
}

MS = TS / 8;
//Find the score closest to the average
for (int i = -1; i<2; i++) {
     for(int j=-1; j<2; j++) {
          if(!(i ==0 && j==0)) {
               int cols = (x+i + col) % col;
               int rows = (y+j + row) % row;
               int distance = Math.abs(MS - activeGridScore[cols][rows]);

               if(distance< smallestDistance){
                    smallestDistance = distance;
                    newColour = grid[cols][rows];
               }
          }
     }
}
```

Figure 30 average algorithm

Figures 31 and 32 show the successful implementation of test 5. In the simulation, the long-term average of CO in the population stands at 84%. The histogram shows the possible figure by showing a steady uptrend from cycles 0-50 to cycles 150-200. Halfway through the population, the percentage DE starts to stabilise at around 15%. The data suggests mutual coexistence is possible, but with favour towards a cooperative population. The screenshots help figure out what occurs during the simulation run. By round 250, most of the grid is coloured green, but there remains a large cluster of red shifting from left to right. The possibility for this cluster to grow or be invaded by cooperative agents is still open.

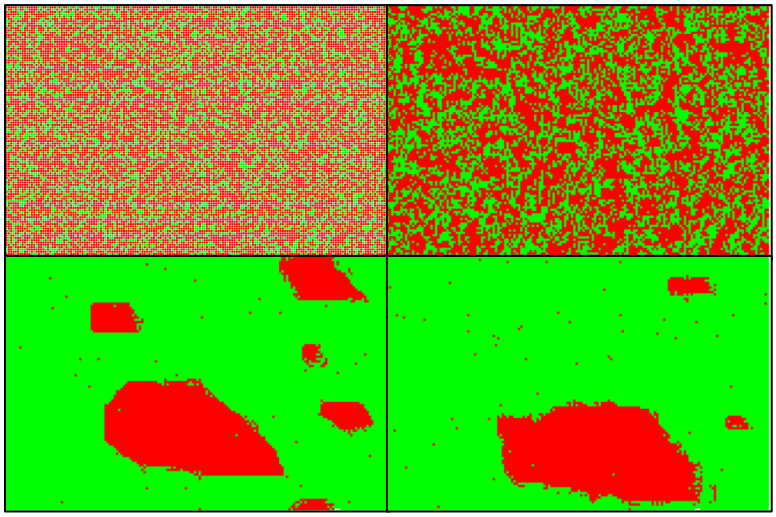

Figure 31: (Test 5) Simulation of Average imitation for the Chicken game showing rounds 0,1,250 and 500. u=0.0025, initial population CO=1/3 DE=2/3

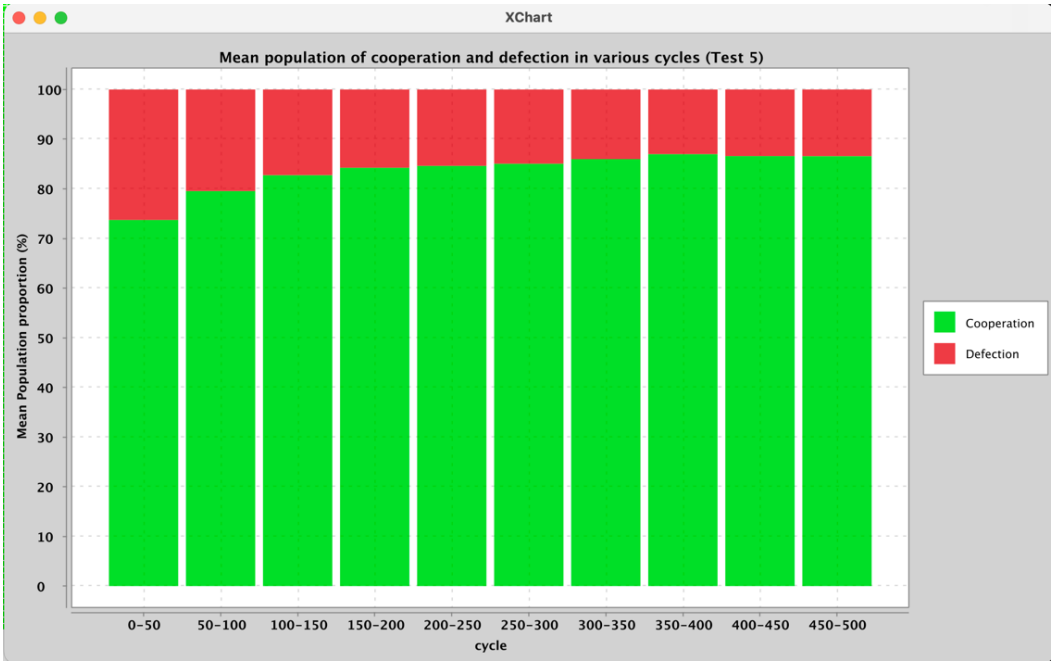

Figure 32: Histogram plot of test 5 showing the mean population dynamic per 50 cycles for 500 rounds. The first 2 rounds are removed to avoid the transient effects of initial configuration resulting in a long-term average of **84%** for CO

Figures 33 and 34 show the successful implementation of test 6. In both figures its shows that DE is wiped out from the population. Unlike other versions of the prisoner's dilemma DE is unable to invade a population of CO. There remains a small percentage of DE through mutations which prevents the long-term average from being 100%.

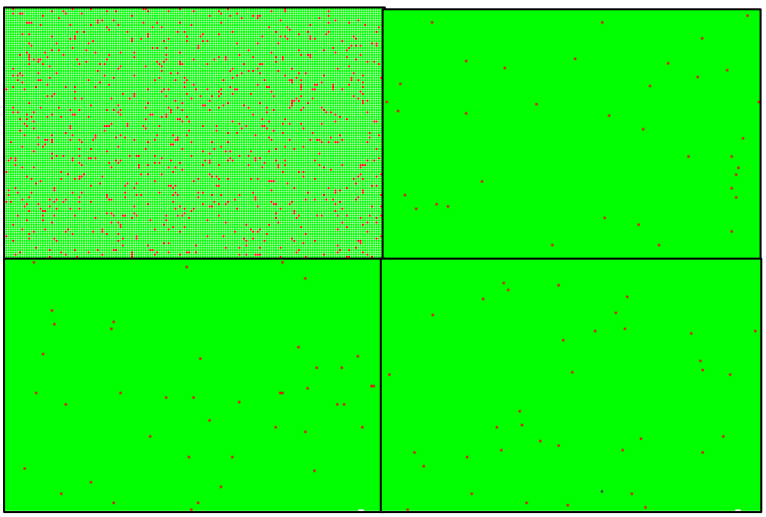

Figure 33: (Test 6) Average imitation for the prisoner's dilemma showing rounds 0,1,250 and 500, u=0.0025, initial population CO=95%, DE=5%

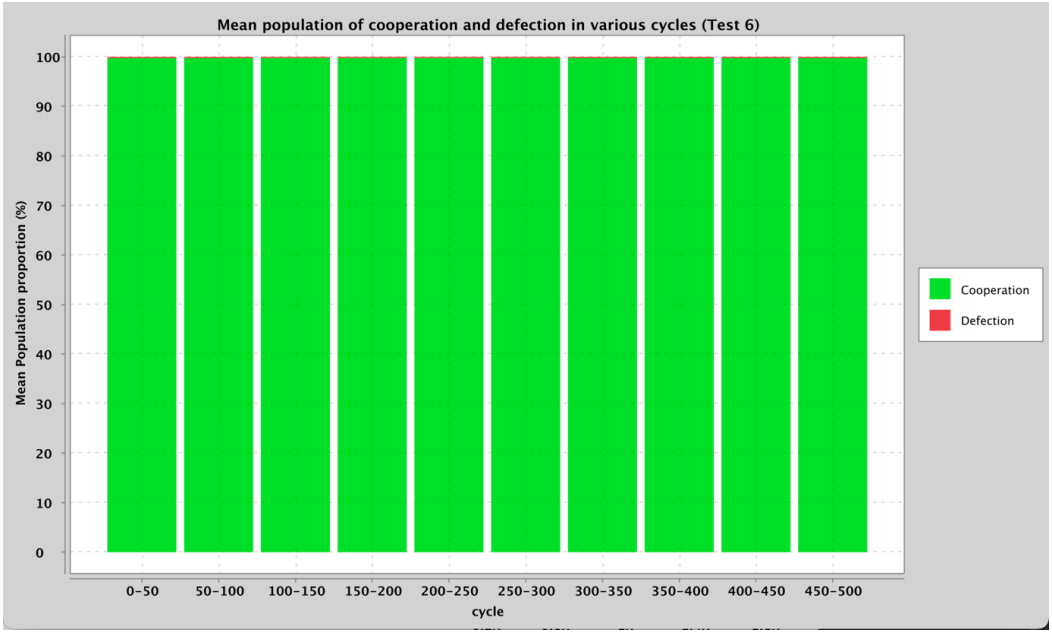

Figure 34: Histogram plot of test 6 showing the mean population dynamic per 50 cycles for 500 rounds. The first 2 rounds are removed to avoid the transient effects of initial configuration resulting in a long-term average of **99.7%** for CO

4.5 Random Imitation

The last imitation method to implement was based on copying based on randomness. In the design process, it was discussed that a cell would pick a neighbour to imitate based on equal probability amongst them. The payoff score becomes irrelevant in this instance. To keep the testing strategy consistent, two versions of this simulation needed to be created following the starting conditions of the Chicken game and Prisoners dilemma.

Figure 35 shows the code used to instruct a random choice of neighbour. The key variables were 'count' and 'randomiser'. The count variable was set to 0, and the randomiser chose a number between 0-7 after every instantiation of the method. Every neighbour gets iterated; if the randomiser value and the count equate, that neighbour is selected to be imitated.

```
int count = 0;
int randomiser = rand.nextInt(8);

int mutation = rand.nextInt(400);

for (int i = -1; i<2; i++) {
     for(int j=-1; j<2; j++) {
         if(!(i ==0 && j==0)) {
             int cols = (x+i + col) % col;
             int rows = (y+j + row) % row;
             //If the randomiser matches the count select that neighbour to copy
             if(randomiser == count) {
                 newColour = grid[cols][rows];

             }
             count++;

         }

     }
}
```

Figure 35: Random algorithm

Figures 36 and 37 show the successful implementation of test 7. The histogram shows that CO continued to grow from the initial starting conditions of 33.33% to highs of 52%. The resulting average due to the upscale got to 45%. When observing the screenshots from the simulation, it is evident that random imitation is occurring. Both defective agents and cooperative agents invade each other without a clear side dominating.

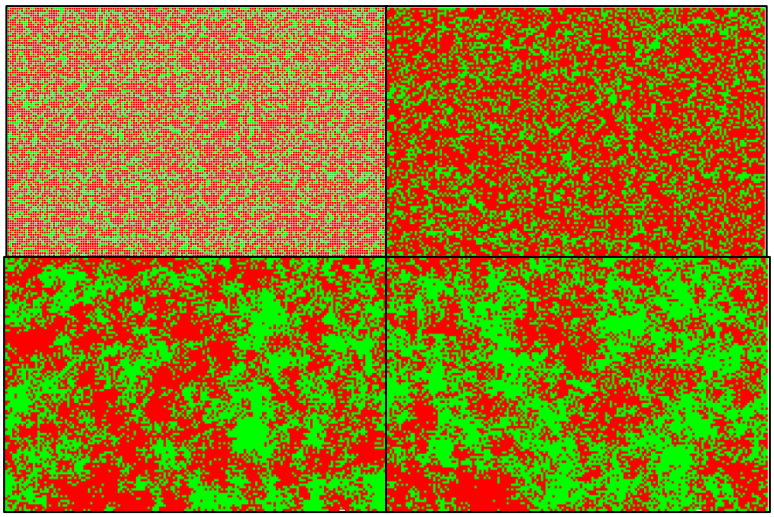

Figure 36: (Test 7) Random imitation simulation showing rounds 0,1,250 and 500. u=0.0025, initial population CO=1/3 DE=2/3

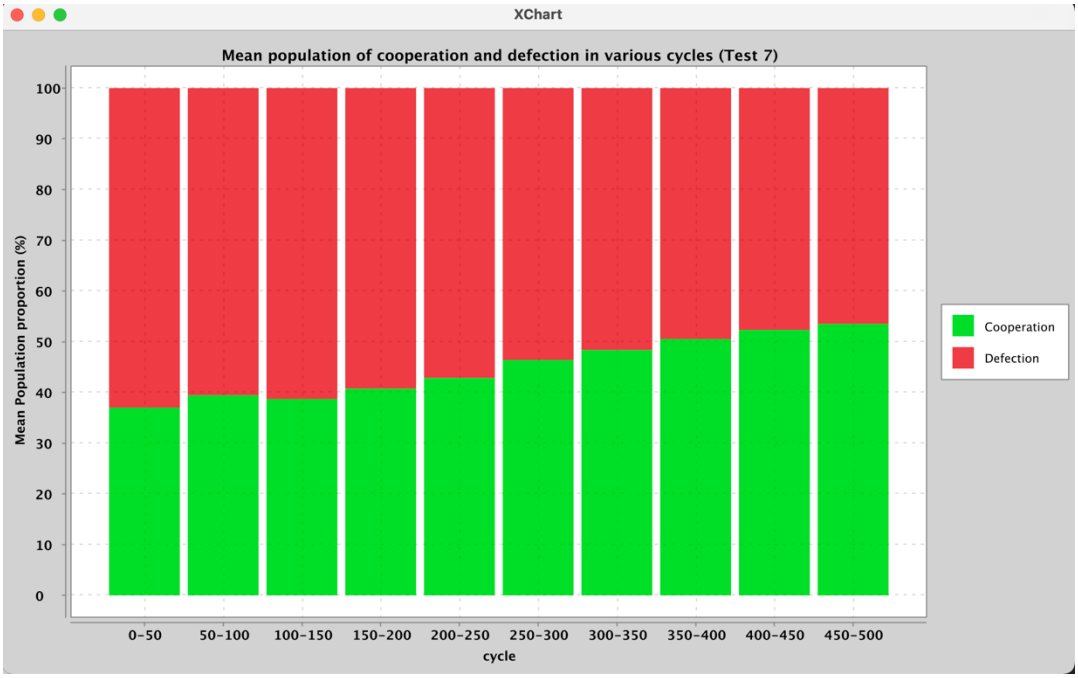

Figure 37: Histogram plot of test 7 showing the mean population dynamic per 50 cycles for 500 rounds. The first 2 rounds are removed to avoid the transient effects of initial configuration resulting in a long-term average of **45%** for CO.

Figures 38 and 39 show the successful implementation of test 8. The simulation starts with a cooperative majority. Cycle 0-50 represents this with a percentage share of 90%. The mean number of CO continues to decrease after every cycle. By the time the last cycle is reached, the population dynamics are close to 50%. This trend can also be spotted viewing the screenshots in figure 38; DE can invade cooperative regions. While the amount of CO is decreasing in the population, the spread of its agents suggests that it is also successful at invading defective regions.

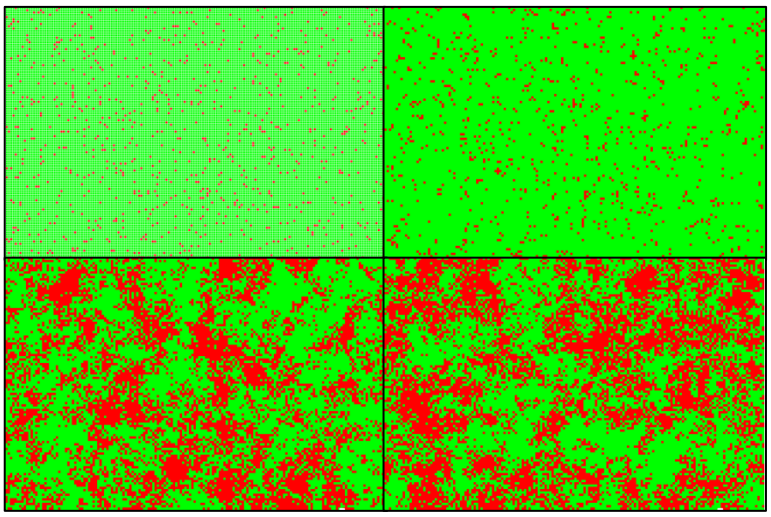

Figure 38: (Test 8) Random imitation simulation showing rounds 0,1,250 and 500, u=0.0025, initial population CO=95%, DE=5%

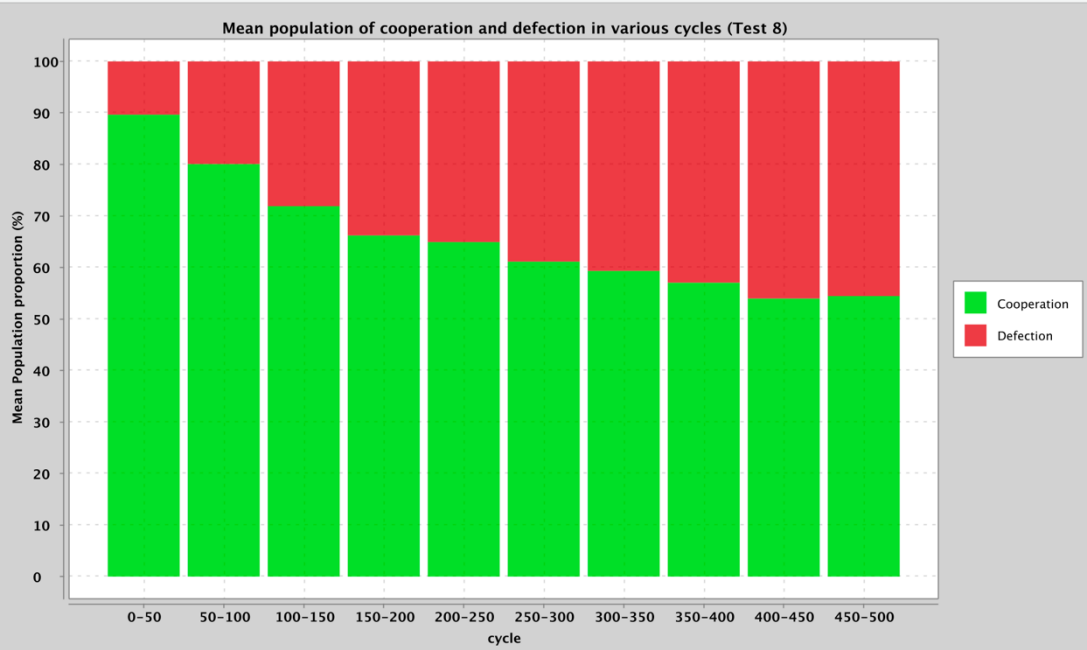

Figure 39: Histogram plot of test 8 showing the mean population dynamic per 50 cycles for 500 rounds. The first 2 rounds are removed to avoid the transient effects of initial configuration resulting in a long-term average of **66%** for CO.

Chapter 5

Evaluation

In this chapter, an extensive evaluation of the results produced from the project will be carried out using statistical analysis and graphs. The further meaning of the work produced will be discussed using literature references for context. The standard of work would be critically analysed to understand what was successful and what improvements could be made if attempted again.

5.1 Analysing Results from the Chicken game

The role of imitation in the conflict between CO and DE was carried out by observing CO during 500 rounds. The Chicken game for each test started with a cooperative population of 1/3 and a defective population of 2/3. Immediately DE has an advantage over CO due to CO being the favoured strategy when copying the most successful neighbour. By using fixed starting conditions, the actual results do not matter individually. The critical factor that matters is how the results compare against one another. Before comparison is drawn, the validity of each test must be carried out by completing ten repetitions per imitation strategy.

NO	**Long term average (%)**
1	56.31
2	55.23
3	55.15
4	55.30
5	58.92
6	68.69
7	60.00
8	55.40
9	64.55
10	70.32
M	**65**
σ	**5.82**

Table 5: 10 repetitions of test 1 to produce the long term average of CO. M= mean, σ=standard deviation

In table 5, the average percentage of CO is 65%. Being above 50% given the starting conditions suggests that CO is the preferred strategy when imitating the most successful neighbour. The histogram (see figure 22) showed that the value would not significantly increase with DE resisting further invasion. The results produced from the tests align with the

theory of the Chicken game. It was expected that mutual coexistence would exist, with neither state entirely dominating the other. The accuracy of the results can be met with scepticism with a standard deviation of 5.82. The lowest CO value is 55.40%, and the highest is 70.32%, which shows how cooperative agents invade defective regions at a different rate based on initial conditions.

NO	Long term average (%)
1	92.16
2	93.75
3	93.8
4	93.81
5	93.88
6	93.90
7	93.71
8	93.78
9	97.21
10	93.78
M	**94**
σ	**1.25**

Table 6: 10 repetitions of test 3 to produce the long term average of CO. M= mean, σ=standard deviation

Table 6 shows the results when the Chicken uses imitation based on memory. Imitating on memory consistency produced a long-term average of 94% for ten iterations. A standard deviation of 1.25 makes this set of results more consistent than table 5. When the past three rounds are considered, CO is more preferred to DE agent by a considerable amount. When a single example of test 3 was produced, it was shown that DE can still exist in the population but only at a tiny percentage.

NO	Long term average (%)
1	62.02
2	81.36
3	67.24
4	63.03
5	80.36
6	66.69
7	68.16
8	78.81
9	59.99

10	63.56
M	**69**
σ	**8.04**

Table 7: 10 repetitions of test 5 to produce the long term average of CO. M= mean, σ=standard deviation

Test 5 looks at the amount of CO in a population when copying the average neighbour. These tests produced the most variance out of all the tests, with a standard deviation of 8.04. With a mean of 69%, CO occupies a more considerable portion of the population. Even though the results were not consistent, all results end up with clusters of defective regions surrounded by CO. This suggests that average imitation separates the two states from mixing in the population.

NO	**Long term average (%)**
1	43.25
2	43.50
3	48.07
4	45.94
5	43.55
6	42.25
7	42.44
8	44.89
9	46.60
10	46.45
M	**46**
σ	**2.11**

Table 8: 10 repetitions of test 7 to produce the long term average of CO. M= mean, σ=standard deviation

Table 8 shows the results for random imitation. All the ten results had minor variance producing a standard deviation of 2.11. The mean result of 46% shows that CO can invade and gain ground on DE. In the given number of rounds, the population is still DE dominant, but this proportion is close to even. The results produced to align with the views before attempting random imitation. The probability may be heavily weighted at the beginning of the simulation, but given enough time, the probability of selecting either CO or DE becomes 0.5.

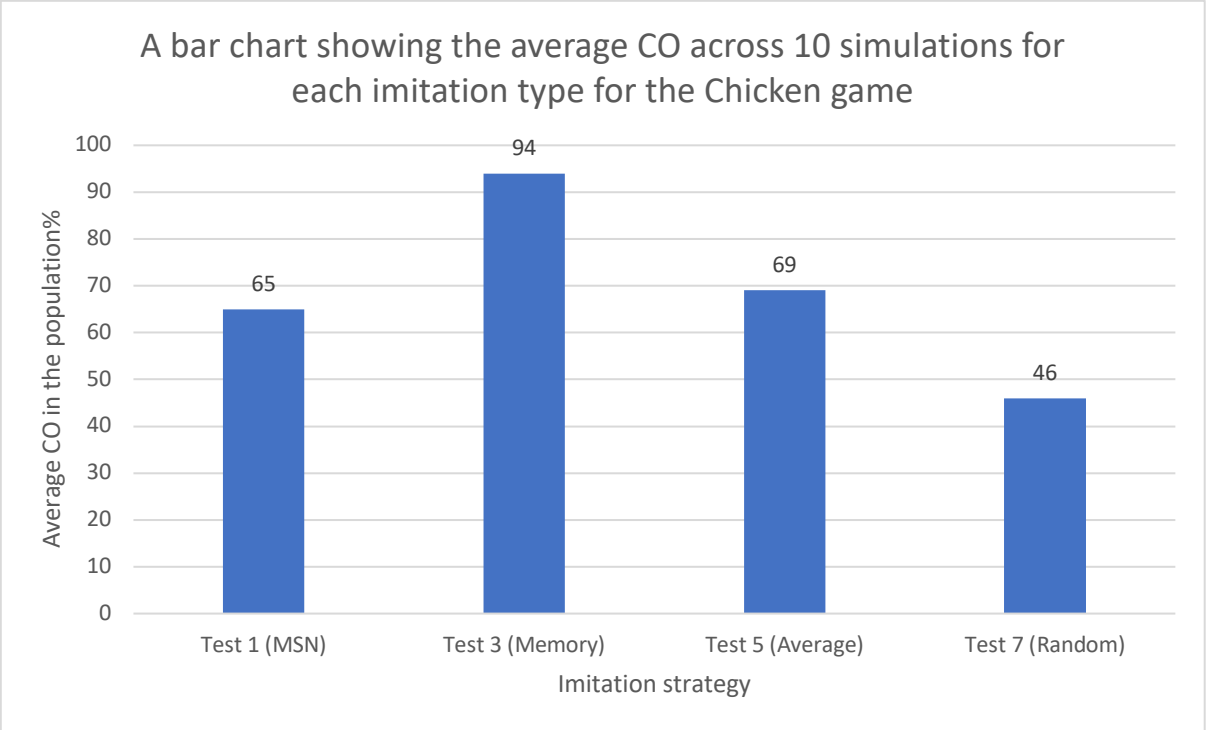

Figure 40: Bar chart comparing imitation strategies for the Chicken game

Figure 40 combines the averages of each test for a long-term average of CO across different imitation types. All 4 of the different imitation strategies produce different results for the CO population. The variance of results proves that imitation is a decisive determining factor in the population dynamics. Each strategy is evaluated by its success in aiding the invasion of DE by CO. From the bar chart, the order of succession is **Memory > Average > MSN > Random.**

Random imitation aids defectors the most. The removal of the payoff matrix means there is no motivation for agents in a particular round to choose to imitate a successful strategy. While defectors do not take over, it resists the invasion attempt the most. In this experiment imitating the most successful neighbour is the standard to compare the rest; while the strategy is successful in aiding CO, it has been proven to be beaten as the most dominant imitation type. The next most successful strategy is imitating the average. When implemented, DE loses more population compared to the MSN. A possible reason for this is down to CC and CD being the two scenarios in the centre of the payoff matrix. DC is not rewarded to the same extent as that would require a cell to seek the most successful neighbour.

When a cell considers the payoff score of the MSN over past iterations, DE fails at resisting the invasion of CO the most. In a standard simulation of the Chicken game, a single round for a defective agent surrounded by cooperative agents would see it score the highest since DC is the highest payoff score scenario. In the next round, a cooperative agent may seek to copy that defective agent, which results in a DD scenario. It is established that two defections are treated harshly in the Chicken game. With that logic, it makes sense why a cell with access to past iterations would choose to imitate a cooperative agent to reduce the risk of "crashing".

5.2 Analysing Results from the Prisoner's Dilemma

The experiments were repeated for every imitation strategy using the prisoner's dilemma instead of the Chicken game. To further understand the general trends of each imitation, the tests done in the implementation chapter were repeated ten times to produce an average figure of the amount of CO in the population. The starting conditions remained the same for each repeated test, with a population of 95% CO and 5% DE. The aim for the PD is the ability of CO to resist invasion, considering DE is the standard, more dominant strategy choice.

NO	Long term average (%)
1	0.87
2	0.89
3	0.87
4	0.88
5	0.88
6	0.86
7	0.86
8	0.88
9	0.87
10	0.86
M	**0.9**
σ	**0.01**

Table 9: 10 repetitions of test 2 to produce the long term average of CO. M= mean, σ=standard deviation

After ten repetitions, the mean percentage of CO when imitating the most successful neighbour is 0.9% (see table 9). The reliability of the small sample is high due to a standard deviation of 0.01. From the literature, the domination of DE was expected. The extent, however, to how small the size of CO was surprising. The existence of mutation could have played a role in the lack of cooperative regions across the grid.

NO	Long term average (%)
1	4.73
2	3.27
3	5.34
4	6.39
5	7.32
6	8.65
7	8.87
8	4.65
9	9.52

10	6.15
M	**6.5**
σ	**2.07**

Table 10: 10 repetitions of test 4 to produce the long term average of CO. M= mean, σ=standard deviation

Test 4 looked at the role imitation, when memory is used in the internal conflict of CO and DE in the PD game. Table 10 shows the long-term average of CO, where the percentage is 6.5%. The standard deviation was 2.07, showing values ranging from 2% -10%. These values proved mutual existence can exist.

NO	**Long term average (%)**
1	99.73
2	99.74
3	99.74
4	99.73
5	99.73
6	99.73
7	99.73
8	99.73
9	99.73
10	99.73
M	**99.7**
σ	**0.0048**

Table 11: 10 repetitions of test 6 to produce the long term average of CO. M= mean, σ=standard deviation

Test 6 simulated the effect of copying the neighbour closest to the average on the population proportion. Table 11 shows that when repeating the test 10 times, the given mean is 99.7%. The test proved that DE cannot exist under these conditions and is mainly invaded by CO. The results prove to be consistent with a standard deviation of 0.0048.

NO	**Long term average (%)**
1	64.29
2	64.94
3	67.63
4	65.40
5	68.96
6	64.81
7	65.62
8	65.46

9	64.50
10	65.86
M	**65.8**
σ	**1.46**

Table 12: 10 repetitions of test 8 to produce the long term average of CO. M= mean, σ=standard deviation

Test 8 involved simulating a population where imitation is based on randomness. The payoff matrix used for the PD became irrelevant, but the starting conditions remained. After ten repetitions, the average long-term mean was 65.8%. A standard deviation of 1.46 showed that values consistency existed in the 60% region. The mean value proves that mutual coexistence becomes possible with randomness. A 30% drop-off suggests that DE was successful at invading a once dominant cooperative population.

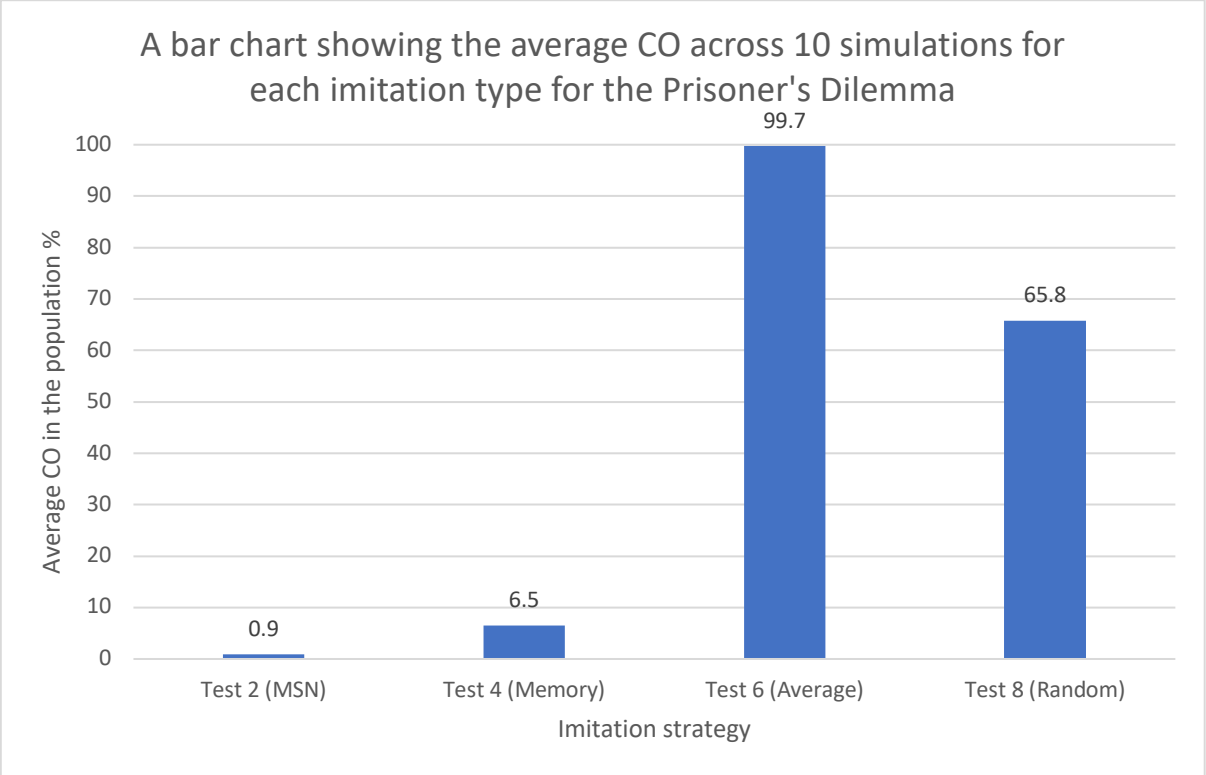

Figure 41: Bar chart comparing imitation strategies for the Prisoner's Dilemma

Figure 41 compares the average scores of each imitation strategy to understand the role imitation plays on the PD. Once again, each strategy produced has completely different results proving that imitation plays a significant role in the conflict. From the bar chart, it can be concluded that the strategies that best aid CO to resist defective invasion are **Average > Random > Memory > MSN.**

CO can resist DE from invading easier when copying the MSN based on the last three iterations. Instead of being wiped out, a few defectors constantly switch to CO, keeping the total population at around 6%. It was discussed in the literature that three steps allow a player to predict an opposing player's strategy. These predictions would allow some defective agents to predict the lack of benefit from remaining in a DD situation. Switching to CC would temporarily increase the payoff score. Eventually, being surrounded by too many defectors causes a switch back. The result is a forever shifting grid of small cooperative regions being created and destroyed.

Random imitation results in more CO resisting DE compared to memory. As discussed earlier, the dominant strategy of ALL-D has no effect when the strategy is used. Even though randomness is the 2nd best, the downward trend of CO suggests that DE would invade more given enough time. The most successful strategy is Average imitation. Its CO value of 99.7% is the opposite of the MSN cooperative population. The initial high amount of CO meant that the high payoff scores attained by DC scenarios were unable to persuade other agents to switch. In this system, Cheating is no longer seen as a desirable choice.

5.3 Performance Overview

At the start of this report, a set of objectives were aligned to complete the main aim of uncovering the role imitation plays between CO and DE. In this section, the effectiveness of the completed model will be investigated to gather if it was successful or if more work needed to be done.

The first criteria to check were the requirements discussed in the design chapter. These requirements covered the main components that would tailor the attempted CA model to the objectives (see table 1, page 23).

- F1: Overall, this requirement was a success. The model resembled a CA model with a 2d grid. Only two states are made possible, resembling CO and DE in the colours green and red, respectively.
- F2: The grid dimensions were adjusted to a certain degree. All the code variables, such as column to row count, could be adjusted through one variable. The lack of control, however, of the dimensions leads to rows and columns not being equal.
- F3: The implementation of stopping the simulation proved successful. In Java, the use of multithreading made it possible for the thread used to complete the simulation to come to an end without shutting down the entire simulation. This made room for screenshots and plots to be produced.
- F4: In the working GUI, the display information correctly outlines the data for each round. In the application, the round, CO, and DE counter are all updated after every iteration.

- F5: The attempt to implement an algorithm for the MSN proved successful. When completing the code, no errors crashed the simulation; the use of print statements also made testing if the correct imitation occurred possible.
- F6: Coding an algorithm to imitate the average proved more complicated than the MSN strategy. There required two steps to make the desired outcome possible. More testing could have been done to guarantee that a cell correctly copied the correct neighbour.
- F7: The design outlined for incorporating memory into an MSN algorithm was seamlessly able to be transferred into code. The payoff scores after round 3 tripled compared to results first tested in F5.
- F8: The random algorithm was the last and most straightforward to implement. Java's built-in seed generator made producing random values for every cell possible. The algorithm's success was quickly assured by observing the population evolution once run.
- F9: The raw data from the simulation could be extracted to produce arrays that could later be analysed using data points.
- F10: The requirement of play and pause buttons was beneficial to transfer the visual simulation to fixed screenshots at specific times. If the simulation ran to a standstill, behaviour about the interactions between the two states would be lost.
- F11: lastly, a mutation probability was also successfully implemented; when the experiments were carried out, observations could be made of states switching sides.

Completing the model requirements helps guide the work to complete the main objectives. A model was created using an imitation strategy involving memory, average and randomness. The standard imitation strategy in the MSN was also used as a baseline of comparison. However, certain design choices call into question whether the aim can be confidently answered or not.

Configuration choices

Choosing a population size of 18,260 proved to have its benefits and weaknesses. At the size chosen, the individual changes between each agent became negligible. If one state could not wholly invade another region, the population would appear to stay stable despite constant changes in local interactions. This choice was reflected poorly in the histogram plots produced. Gains in territory barely shifted the percentage share. Without screenshots, it would be difficult to understand what occurred during a simulation run. In the literature, the tests for the Snowdrift game and PD only used 20 cells. The small population size allowed for local interactions to have more influence on the overall conflict.

Simulating the model for 500 rounds was a choice that never consistently felt correct. The nature of the different tests left the desire for more rounds or less. In cases like test 6, which involved imitation using averages for PD, a graph showing 50 rounds would have fleshed out the trends more. In cases such as random imitation, a simulation with 1000 rounds could have brought the average value of CO to 50%. In the end, a simulation showing all the possible

periods would have been preferred to understand each role better. In hindsight, the choice to remove only the first two rounds out of data collection was a mistake. The transient effects still played a role and influenced the final average. It can often be seen that the first cycle's histogram results are different to the rest. A more accurate long-term mean will have been achieved if the cycle's started from round 50 instead of 2.

The inclusion of a mutation probability was an area that took many tests to find the most suitable value for these experiments. In most models, the mutation rate stood at 1%. While there was intent to uphold these conditions, the mutation rate proved too impactful on the model. For instance, test 2 (PD using MSN) held a higher CO population with no mutation than with one. In those conditions, cooperative agents' existence depends on structures preventing defection. When a mutation attacked one of the structures, it often broke down any existing agents to stay cooperative. Setting the value to 0.25% felt like the perfect balance of noise inclusion without playing too dominant a role in the tests.

Imitation choices

Copying the MSN was a given starter strategy to implement. The inclusion confirmed that the model functioned correctly due to past literature results. When applied to the Chicken game, the behaviour was expected, and coexistence allowed the understanding of the data to be straightforward. The transfer of this strategy over to the PD proved to be disappointing. No matter what initial starting population was used, coexistence proved impossible; DE constantly invaded the entire population. Setting the initial cooperative value to 95% proved to create opposite results depending on strategy.

Including memory in the implementation was one of the most anticipated strategies due to its complexity. In the end, both games had a noticeable difference, causing improved resistance and increased invasion capacity for CO. The one method used produced a desire for more to be discovered. For example, a cell that monitored only two iterations could produce different results compared to the three iterations which were used. Additionally, the pattern of imitation could have been another strategy to test. In the method used, a cell only looks at the payoff score of the neighbour before copying. A payoff score is no indication of the pattern used by the cell. Another memory version could observe the states that appear more to copy.

How interesting the average strategy was depended on the game used. It is well established that in the PD, no exciting observations could be drawn due to the fixed starting conditions. If tested again, the initial conditions could favour DE to observe if CO can still invade with the pace shown in test 6. The chicken game proved that copying the average could produce varied results. The high standard deviation and forever changing population leaves this test with further exploration to be conducted.

What role does imitation play

The deployed strategies proved to have a uniquely significant role—different invasion tactics needed to be used to gain confidence in each strategy's effect. If the same behaviours and population shifts occurred, the findings could be more conclusive. Nevertheless, in the

Chicken game, the strategy that best aided the invasion of CO was imitation with memory. The strategy that best helped DE resist invasion was random imitation. While memory also sees a boost in the PD, the best strategy that helps CO resist DE is average imitation. When it comes to invasion, the best strategy in helping DE is the imitation of the MSN.

Chapter 6

Conclusions

In conclusion, the solution proposed aimed at producing a CA model that would test the role imitation played between the conflict between CO and DE. To understand the role imitation plays a couple examples had to be tested to observe the changes. The strategies were MSN (most successful neighbour), imitation using memory (the incorporation of MSN but with the payoff scores of the past three moves, imitating the average and random imitation.

It would be impossible to test strategies without a game to play it with. Two games were chosen to see if the chosen strategy effect on conflict transferred. The two games were the Chicken game and the prisoner's dilemma. When both games were incorporated the results were staggery different. The chicken game when incorporating MSN consistently produces environments where both CO and DE can invade. The equal playing ground meant any other strategy could prove aid DE and DE in an obvious way. The PD when using the MSN as a strategy typically results in DE being dominant across the population. Since the PD was more one sided, any of strategy was likely to improve the resistance of CO.

After testing all the strategies a few conclusions about game theory can be made. When a player has access to past iterations, the choice of CO becomes more favourable for both games. In nature, players who recognise the consequences of greed are less likely to replicate that. When there are only recent events to compare to, the choice to cheat becomes more favourable due to short slightness. An average strategy also improves CO resistance and invasion capability across both games. The results really bring to light how populations can all seek to benefit when no one seems to be greedy. While individual success suffers, a collective improvement in well-being and survival is the more desired approach.

Randomness also speaks volumes how behaviour and personal motivation leads to in balance of behaviour choices. There is simplicity in the idea that no state can overcome the other no matter the original starting conditions. To summarise, imitation plays a crucial role in game theory. Cooperative societies, which help shape a functioning society only works, when players are cautious of the risk of cheating and or not tempted by greed. When everyone seeks maximum benefit, a defective population becomes apparent.

In future work more strategies can be undertaken hopefully to better understand the role being taken. One such strategy is trying different variations of memory. A study trying to uncover the mathematical principles on how certain strategies behave is also interesting. Additionally, some tests failed at co-existence. When experiments are conducted again, the right conditions and scores could prevent it.

References

1. Vlebooks.com. (2022). Full details and actions for The computational beauty of nature: computer explorations of fractals, chaos, complex systems, and adaptation. [online] Available at: https://www.vlebooks.com/Product/Index/2060996?page=0 [Accessed 17 Apr. 2022].
2. Proceedings of the Royal Society of London. Series B: Biological Sciences. (2021). Spatial evolutionary game theory: Hawks and Doves revisited | Proceedings of the Royal Society of London. Series B: Biological Sciences. [online] Available at: https://royalsocietypublishing.org/doi/abs/10.1098/rspb.1996.0166 [Accessed 27 Apr. 2022].

3. Edc.org. (2021). Spatial Games. [online] Available at: https://youngmathematicians.edc.org/math-game/spatial-games/ [Accessed 27 Apr. 2022].

4. Mérő, L. (1998). John von Neumann's Game Theory. Moral Calculations, [online] pp.83–102. Available at: https://link.springer.com/chapter/10.1007/978-1-4612-1654-4_6#:~:text=The%20fundamental%20theorem%20of%20John,neither%20player%20should%20deviate%20unilaterally. [Accessed 27 Apr. 2022].

5. Pure Mathematics. (2015). What is Topology? | Pure Mathematics. [online] Available at: https://uwaterloo.ca/pure-mathematics/about-pure-math/what-is-pure-math/what-is-topology#:~:text=Topology%20studies%20properties%20of%20spaces,but%20a%20figure%208%20cannot. [Accessed 28 Apr. 2022].

6. Historyofinformation.com. (2022). Von Neumann & Morgenstern Issue 'The Theory of Games and Economic Behavior' : History of Information. [online] Available at: https://www.historyofinformation.com/detail.php?id=1803 [Accessed 29 Apr. 2022].

7. Mookherjee, D. and Sopher, B. (1994). Learning Behavior in an Experimental Matching Pennies Game. Games and Economic Behavior, [online] 7(1), pp.62–91. Available at: https://www.sciencedirect.com/science/article/pii/S0899825684710372 [Accessed 29 Apr. 2022].

8. Economics. (2022). Chicken game. [online] Available at: https://economics.fandom.com/wiki/Chicken_game#:~:text=A%20chicken%20game%20is%20a,%2C%20implicitly%20braver%20player%2C%20wins. [Accessed 9 May 2022].

9. Kuhn, S. (2020). Prisoner's Dilemma (Stanford Encyclopedia of Philosophy/Winter 2007 Edition). [online] Sydney.edu.au. Available at: https://stanford.library.sydney.edu.au/archives/win2007/entries/prisoner-dilemma/ [Accessed 10 May 2022].

10. Investopedia. (2022). What Is the Prisoner's Dilemma? [online] Available at: https://www.investopedia.com/terms/p/prisoners-dilemma.asp#:~:text=A%20prisoner's%20dilemma%20is%20a,many%20aspects%20of%20the%20economy. [Accessed 10 May 2022].

11. Wolfram.com. (2022). Cellular Automaton. [online] Available at: https://mathworld.wolfram.com/CellularAutomaton.html#:~:text=A%20cellular%20automaton%20is%20a,many%20time%20steps%20as%20desired. [Accessed 12 May 2022].

12. the (2019). Cleaner Wrasse 'Clean' a Grouper. [online] Aquariumofpacific.org. Available at: https://www.aquariumofpacific.org/blogs/comments/cleaner_wrasse_clean_a_grouper#:~:text=There%20the%20wrasse%20will%20eat,benefits%20for%20the%20larger%20fish. [Accessed 12 May 2022].

13. Wikipedia Contributors (2022). Wrasse. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Wrasse#/media/File:Thalassoma_lunare_1.jpg [Accessed 12 May 2022].

14. ScienceDaily. (2019). Ants that defend plants receive sugar and protein. [online] Available at: https://www.sciencedaily.com/releases/2019/07/190717230337.htm [Accessed 12 May 2022].

15. unknown (2003). Figure 1: An example of a one dimensional cellular automaton, the Rule... [online] ResearchGate. Available at: https://www.researchgate.net/figure/An-example-of-a-one-dimensional-cellular-automaton-the-Rule-110-from-Wolfram-21-The_fig1_4030410 [Accessed 16 May 2022].

16. Bays, C. (2010). Introduction to Cellular Automata and Conway's Game of Life. Game of Life Cellular Automata, [online] pp.1–7. doi:10.1007/978-1-84996-217-9_1.

17. Lipa, C. (2022). Conway's Game of Life'. [online] Cornell.edu. Available at: http://pi.math.cornell.edu/~lipa/mec/lesson6.html [Accessed 16 May 2022].

18. Alonso, J., Fernandez, A. and Fort, H. (2005). Prisoner's Dilemma cellular automata revisited: Evolution of cooperation under environmental pressure. [online] ResearchGate. Available at: https://www.researchgate.net/publication/2174761_Prisoner's_Dilemma_cellular_automata_revisited_Evolution_of_cooperation_under_environmental_pressure [Accessed 19 May 2022].

19. Momeni, N. (2018). How Can We Promote Cooperation in an Uncooperative Society? [online] Scientific American Blog Network. Available at: https://blogs.scientificamerican.com/observations/how-can-we-promote-cooperation-in-an-uncooperative-society/ [Accessed 27 May 2022].

20. Nakayama, B. (2013). Universal Computation in the Prisoner' s Dilemma Game. [online] Available at: https://epublications.regis.edu/cgi/viewcontent.cgi?article=1596&context=theses [Accessed 7 Jun. 2022].

21. García-Victoria, P., Cavaliere, M., Gutiérrez-Naranjo, M.A. and Cárdenas-Montes, M. (2022). Evolutionary game theory in a cell: A membrane computing approach. Information Sciences, [online] 589, pp.580–594. doi:10.1016/j.ins.2021.12.109.
22. Wikipedia Contributors (2022). Moore neighborhood. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Moore_neighborhood [Accessed 16 Jun. 2022].

Appendix one

Link to application: https://github.com/Dave196