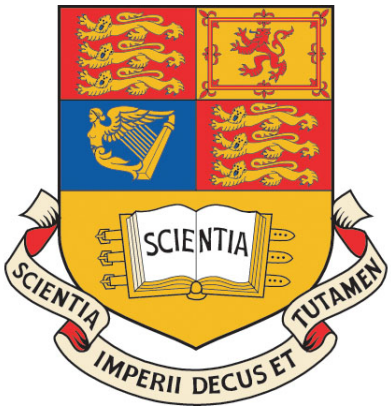
IMPERIAL COLLEGE LONDON
DEPARTMENT OF COMPUTING

Repeated Prisoner's Dilemma

Final Year Individual Project

Interim Report



Lu Xin
(lu.xin08@imperial.ac.uk)

Supervisor: Dr. Daniel Kuhn (dkuhn@imperial.ac.uk)

2nd Marker: TBD

January 2012

Abstract

In the context of Repeated Prisoner's Dilemma, we explore a new paradigm of how mutual cooperation and collusion can be obtained to maximize gains through evolutionary strategies. This project will investigate the use of Genetic Algorithms to create optimal strategies, and extend the current field of research. The aim is to analyse and evolve strategies from strong traits exhibited by existing strategies. These traits include the ability to guard against defectors, and the capacity to capitalize on gains through cooperation with kinder strategies. The project will take advantage of the Exploration-Exploitation methodology approach when developing strategies for an agent interacting in an uncertain environment. The results obtained in this new adapted approach, and the patterns that have emerged are presented in this paper.

Key-Words: - Genetic Algorithms, Prisoner's Dilemma, Game Theory, Explore and Exploit, Evolutionary Computation.

Acknowledgements

I would like to thank Dr. Daniel Kuhn for supervising this fascinating project, as well as providing me with valuable advice and support throughout.

Next, I would like to underscore the importance of work done by Daniela Pucci de Farias and Nimrod Megiddo for their great insight in their paper "How to Combine Expert (or Novice) Advice when Actions Impact the Environment". This helped guide me in my design of an evolutionary algorithm for the purposes of this project.

Contents

Chapter 1

Introduction

The Prisoner's Dilemma is a famous and accepted model, analysed in game theory, used for studying the dynamics of individual interest in the decision-making process. Various strategies arising from Prisoner's Dilemma have been applied extensively as strategies in fields such as economics, politics, psychology, and law. It has thus gained popularity due to its simplicity, relevance and wide range of outcomes that are present.

In a single game of Prisoner's Dilemma, it is trivial to see that picking "Defect" no matter what the opponent picks is always the better choice. However, this does not hold when the game is played repeatedly.

The Repeated Prisoner's Dilemma game has been thoroughly reviewed and many strategies currently exist. The strongest few of the selection are astonishingly simplistic, such as Tit-for-Tat and Pavlov. These strategies have proved effective in multiple tournaments and have withstood the test of time. Nevertheless, it would be compelling if we could use the fundamentals of these strategies as a basis, for further improvement.

The core objective of this project is to create an adaptive agent[1] that will make use of multiple strategies, explore them by applying them, identifying the traits in these strategies, selecting a set, which has proved to be effective in the past, and exploiting it. To do this, the project will also aim to create the necessary support infrastructure to allow these adaptive experts to play the Prison's Dilemma game repeatedly by interacting with other experts, analysing results, and evolving its own strategy in order to optimize its gain.

Furthermore, in this project we will discuss the reasoning behind the well performing adaptive strategies, and give analysis to the outcomes of the different experts. The ultimate goal of the project, and the main challenge is to create a dynamic method of determining and generating the optimal strategy for the given environment when playing repeated games of the Prisoner's Dilemma.

As mentioned above, Prisoner's Dilemma has a wide scope for many real life situations; game theorists in the various fields may therefore apply the results of this project to as a point of inception into analysing and acquiring deeper insight in a situation where a Prisoner's Dilemma is present.

[1] "An agent is a computer system that is capable of flexible autonomous action in dynamic, unpredictable, typically multi-agent domains." [LS05]

Chapter 2

Background

2.1 The Prisoner's Dilemma Game

Prisoner's Dilemma is a game that is analysed in game theory to explain why two individuals may not always choose to cooperate in a given situation, despite the fact that cooperation seemingly emerges as the most profitable choice. The concept was originally coined in 1950. A classic example of Prisoner's Dilemma can be presented as:

> *"Two men are arrested, but the police do not possess enough information for a conviction. Following the separation of the two men, the police offer both a similar deal- if one testifies against his partner (defects/betrays), and the other remains silent (cooperates/assists), the betrayer goes free and the cooperator receives the full one-year sentence. If both remain silent, both are sentenced to only one month in jail for a minor charge. If each 'rats out' the other, each receives a three-month sentence. Each prisoner must choose either to betray or remain silent; the decision of each is kept quiet. What should they do?"*

Particularly, the game is one where two players are faced with a choice, and they each have two options: Cooperate (C) or Defect (D). The players are then awarded a payoff depending on the choice they made compared to the choice of their opponent. Players make decisions without knowledge of the other player's choice, neither can they communicate before or during the game.*[1]

In the above mentioned example, we observe that no matter what the other person chooses to do, it is always a safer bet to go for Defect, even though mutual Cooperation would be the overall choice for both players. This is the Prisoner's Dilemma.

Whilst the game seems simple and the best choice for both players is easily identifiable, the theory behind it can be applied to an array of different real life situations.

[1] Other forms of the Prisoner's Dilemma game also exists where one player moves first, and the second player then decides on a move depending on what the first player chose. The first player gets a 'first mover's advantage' in some respect, but the second player has more info to make his move. However in this project we will not delve into these other forms of Prisoner's Dilemma, we will only discuss the basic original version.

2.2 Payoffs and Equilibrium

The outcomes to a single round of Prisoner's Dilemma is as follows:

2.2.1 Outcomes

- *Both Cooperate* - both players receive a reward R,
- *Both Defect* - both players receive a punishment P
- *One Cooperate, One Defect* - the player defecting a large payoff T (temptation), while the player cooperating gets a small payoff S (sucker).

These payoff can be typically described in the matrix below (define the default values as [R,S,T,P] = [3,0,5,1]):

2.2.2 Payoffs Table

		Player 2	
	Decision	Cooperate	Defect
Player 1	Cooperate	R=3 R=3	S=0 T=5
	Defect	T=5 S=0	P=1 P=1

R: REWARD S: SUCKER T: TEMPTATION P: PENALTY

$$T > R > P > S,\ (T+S)/2 < R$$

As we can observe from the table above, it is always better for a player to defect regardless of what the choice of the other player might be. Intuitively, a dilemma exists where both players defect to either gain the large payoff for temptation, or to defend themselves from the opponent defecting, thus both scoring worse than if both had trusted each other and cooperated. We can describe this outcome by studying the concept of Nash Equilibrium.

2.2.3 Nash Equilibrium

A Nash equilibrium is a solution where both players maximize their score based on their correct assumptions of the other player's choice, thus neither player has anything to gain from changing their choice and an equilibrium is achieved. [OR94]

As we can see from the given payoff table, if player 2 assumes player 1 will choose to cooperate, it should defect in order to get the 5 points for Temptation rather than the 3 points for Reward, on the other hand, assuming that player 1 chooses not to cooperate, player 2 should not cooperate to in order to both gain extra points (1 point for Penalty rather than 0 for Sucker) and not allow the opponent gain more points. The same is true for player 1. Thus considering the outcomes, and seeing that the players will not face each other again so they can play as selfish as they'd like, the payoff is always better when the player defects regardless of what the opponent chooses.

Thus under the assumption that both players are logical, they will both notice this and both choose to defect, earning 1 point each. This results in a Nash Equilibrium where both players defect in a single round of Prisoner's Dilemma[2]. However this is lower than the Reward payoff of 3 points if they both chose to cooperate, this is where the dilemma lies. [RO94]

On the other hand, we can examine the game from a different perspective, by borrowing a concept used in Economics, namely Pareto Optimality.

2.2.4 Pareto Optimality

Pareto Optimality, or commonly know as Pareto Efficiency, is the allocation of a set amount of goods amongst a set of members, such that there is no change of allocation that can make at least one member better off without making any other member worse off. [FT93]

From this concept, we can conclude that, if players chose to cooperate, they would both earn 3 points, and this is the Pareto Optimal solution.

2.3 Achieving Collusion

Collusion, in economics, is an agreement between two or more persons, sometimes illegal and therefore secretive, to limit open competition by deceiving, misleading, or defrauding others of their legal rights, or to obtain an objective forbidden by law typically by defrauding or gaining an unfair advantage [Sul03]. However, in our case, it will merely mean the mutual cooperation of two players in order to maximize both their gains since cooperation is not illegal, but rather desired.

Will it ever be possible for a single game of Prisoner's Dilemma to result in mutual collaboration where both players are logical? The answer is **yes**, but not with some other extra information. We will discuss this in more detail in the next section.

To demonstrate, we borrow an example from the documentary 'Nice Guys Finish First'

[2]This always holds true as long as the payoffs adopt the rule of T > R > P > S, and R > (S + T) / 2.

by Richard Dawkins:

> *The popular sport of football as we know it, has a tournament known as the Premier League. Before the creation of it in 2004, First Division was the highest division of English Football. It was considered to be highly prestigious to be in the First Division. In addition, it is also highly profitable for the clubs, due to the fact that it ensured large audiences at all games, as well as fans.*
> *At the end of each seasons, the bottom 3 clubs in the First Division were relegated to the Second Division for the next season. Seeing as the Second Division did not attract as many viewers, and also clubs' reputations would be damaged, relegation was worth going to great efforts to avoid.*
> *On 1977 May 18th, the last day of that season's football, two out of the three relegation teams were already determined. But the third relegation was still in dispute. The table looked like so:*

LEAGUE DIVISION ONE			
		Pld	Pts
18	SUNDERLAND	41	34
19	BRISTOL CITY	41	34
20	COVENTRY	41	34
21	STOKE	42	34
22	TOTTENHAM	42	33

> *It was definitely going to be either Sunderland, Bristol City, or Coventry. Bristol City and Coventry happened to be playing against each other, whilst Sunderland played a fourth team. It was common knowledge that if Sunderland had lost their game (gaining 0 points), then Bristol and Coventry needed only to draw with each other (gaining 1 point each) to secure their place in the First Division. However if Sunderland had won, then losing team would be relegated. The games were played concurrently due to the arrangements of the tournament.*
> *Interestingly enough, the Bristol-Coventry game ran five minutes late, and because of this, the result of the Sunderland game was known before the end of the Bristol-Coventry game. The score of the Bristol-Coventry game was 2-2 at the time, and once the news of the Sunderland game came through, the Coventry manager immediately had the results of the Sunderland game flash up on the screen. The players then knew that they didn't need to bother playing any more, and a draw with the current score would suffice.*
> *As a result, the players of both teams aimlessly kicked the ball around with no challenge to the person in possession because both teams were committed to secure a draw. Thus the teams ended in mutual cooperation.*

From the above incident, we can see that the teams were willing to cooperate once they obtain extra information about the situation. The information here was that defection (playing to win) and cooperation (playing for a draw) both yielded good outcomes (staying in the First Division), and seeing as there is a cost and risk of defecting, the teams chose to cooperate.

As a result, we notice that mutual cooperation is possible with extra information. So in terms of our simple Prisoner's Dilemma scenario, nice guys can finish first. If the game was played repeatedly, we will be able to gain extra information too through the reactions the opponent yield, we will now delve deeper into this model in section 2.4.

2.4 Repeated Prisoner's Dilemma

As mentioned earlier, the dominant strategy, when only a single round of Prisoner's Dilemma is played, is to defect as you will get the best outcome no matter what the opponent chooses. However, what if the game was played repeatedly and players knew that their actions now maybe affect the opponent's actions later?

Iterated Prisoner's Dilemma (IPD) is a fascinating variant of the traditional Prisoner's Dilemma, where two players play an infinite or unknown number of rounds of the game against each other, the payoffs are accumulated in a Round-Robin tournament[3], and the winner is the one with the highest accumulated number of points.

An important factor that IPD introduces to the game is that the two players will be playing each other again and again, therefore the actions of a player may affect the choice of the opponents in subsequent rounds (this is known as 'The Shadow of the Future', a phrase coined by Axelrod). Hence agents can develop strategies relying on what the opponent did in previous stages of the game to predict what they may do in the following rounds.

Now that player's moves influence how the opponent behaves, it in turn influences the expected future payoffs of the player. By repeating the game, with multiple agents in a Round-Robin tournament, it abolishes the always defect strategy as the strictly dominant strategy; it is merely a Nash Equilibrium. This is because some of the more intelligent strategies may take advantage of circumstances where cooperation will gain more points such as in the mutual cooperation case, thus certain strategies may form a collusion (much like in economics) when playing against each other and therefore maximizing their gain in those games. Some of these strategies will be discussed in more detail in section 2.5.2.

[3] A tournament where each player plays all the other opponents once

2.4.1 Repeated Prisoner's Dilemma and Number of Rounds

It is important to note that in Iterated Prisoner's Dilemma, the number of rounds in the games should not be disclosed to the participants. The reasoning behind this is we can analyse the most optimal choices at each stage of the game. Therefore taking the last round for example, this round becomes a single round prisoner's dilemma game where choosing to defect will always yield the best payoffs in either situation, therefore both agents will choose to defect it they wanted to maximize their utility. The same rule can then be applied for the second last round since it does not influence any further decisions; therefore it becomes a one round game again, which will result in a mutual defect too. Thus by induction, we can conclude that the dominant strategy here is again the always defect strategy[BW06].

Round	n	n-1	...	n-2	2	1
P1 Choice	D	D	...	D	D	D
P2 Choice	D	D	...	D	D	D

Therefore, we can successfully conclude that in order for collusion occur in a game of Repeated Prisoner's Dilemma, the game will need to be such that the player's do not know how many rounds there are in each game, or the game is played infinitely.

2.4.2 Properties of Iterated Prisoner's Dilemma

Some of the key characteristics of a game of Iterated Prisoner's Dilemma are summarized as follows:

- Non Zero-Sum game (A Zero-Sum game is mathematical representation of a situation where one player's gain is another player's loss. Therefore the total sum of points amongst all players remains at 0)
- Both the players can simultaneously 'win' or 'lose'
- There is no universal paramount strategy
- The optimal strategy depends upon the opponents the agent is playing against

We will now take a look at the naive strategies that are available in section 2.5.

2.5 Existing Naïve Repeated Prisoner's Dilemma Strategies

There are numerous existing strategies for the game of Repeated Prisoner's Dilemma, and some of the most successful strategies are naïve strategies, which work on simple logic, yet proved to be extremely successful in previous tournaments.

2.5.1 Axelrod's Tournaments

In the late 1970s and early 1980s, Robert Axelrod, a professor of political science lecturing at the University of Michigan, held tournaments of the Repeated Prisoner's Dilemma game. He invited numerous famous game theorists at that time to submit their strategies, to be run on computers. The programs then played all other strategies, including itself, repeatedly. The tournament was held in a Round-Robin format.

The winner of Axelrod's tournament was the Tit-for-Tat strategy. This is a very simple strategy in which the agents always cooperates first, then mirrors the opponent's previous move. The strategy can both benefit from cooperating agents, as well as defend against the defecting opponents. In addition, when matched against itself, it will always cooperate.

A downfall of the Tit-for-Tat strategy is that it makes the assumption that the opponent is always trying to maximize its gain. Thus when it is playing against oblivious agents such as the Random agent (pick moves randomly), the Tit-for-Tat strategy stoops to its opponent's level. It is because of this reason that Tit-for-Tat is not the best performing strategy against newer more exotic strategies. We will then analyse the key characteristics of the stronger strategies. [Axe87]

2.5.2 Existing Strategies

We shall now see some of the strategies that exist today, including the original strategies from Axelrod's tournaments.

	Strategy Name	**Strategy Description**	**Comments**
1	Always Co-operate	Always cooperates with the opponent regardless	Naïve strategy that may get taken advantage of by other strategies which defects to for selfish gains.
2	Always Defect	Always defects against the opponent regardless	This strategy is the dominant strategy for single round of PD. And it will do at least as well as any single opponent.
3	Alternate	Swaps between cooperating and defecting	No real tactics, just cooperates and defects in turn.
4	Random	Picks a random choice, 50%	No strategy, just takes a random choice for a move.
5	Grudger	Cooperate until the opponent defects, then defects forever	Effective strategy against those strategies that try take advantage through defecting.
6	Soft Grudger	Co-operates until the opponent defects, in such case opponent is punished with 4 defects, then offers co-operation with 2 co-operates	A kinder version of the grudger, and may escape the case of death spiral of mutual defects.
7	Gradual	Co-operates until the opponent defects, in such case defects the total number of times the opponent has defected during the game. Followed up by two co-operations	The gradual strategy remembers how nasty the opponent has been in previous rounds, and punishes accordingly. Then tries to offer peace again.

8	Tit-for-Tat	Cooperates in first round, then mirrors opponents previous move	Simple, yet effective strategy, winner of the Axelrod's Tournament.
9	Tit-for-two-Tat	Similar to Tit-for-Tat, but allows the opponent to defect twice before it retaliates	This avoids a death spiral if the opponents defected by chance, or is a pessimistic strategy that starts with defecting.
10	Naïve Peace Maker	Repeat opponent's last choice (i.e. Tit For Tat), but sometimes make peace by co-operating in lieu of defecting	Modified Tit-for-Tat, sometimes tries to make peace when it should defect.
11	True Peace Maker	Co-operate unless opponent defects twice in a row, then defect once, but sometimes make peace by co-operating in lieu of defecting	Modified Tit-for-two-Tat, sometimes tries to make peace when it should defect (even kinder).
12	Naïve Prober	Repeat opponent's last choice (i.e. Tit For Tat), but sometimes probe by defecting in lieu of co-operating.	Modified Tit-for-Tat, sometimes tries to make a gain by defecting when it should cooperate.
13	Remorseful Prober	Like Naïve Prober. But if the opponent defects in response to probing, show remorse by co-operating once	Modified Tit-for-Tat, sometimes tries to make a gain by defecting when it should cooperate, though if opponent retaliates, it backs off.
14	Suspicious Tit-for-Tat	Same as Tit-for-Tat, except starts with defecting	Similar performance to Tit-for-Tat. However the pessimistic defect first can cause issues with many strategies that start by being trusting.
15	Pavlov	If 5 or 3 points scored in the last round then repeat last choice	Very powerful strategy, and often out performs Tit-for-Tat. Logic is simple too, repeats action if it proved profitable last round.

2.5.3 Strong Strategies Analysis

From tests ran with these strategies in a Round-Robin fashion, the result was that Pavlov and Tit-for-Tat were the clear winners. The hypothesis of the results is that the stronger and superior strategies had two simple characteristics in common.

Firstly, the strategies are kind and therefore they can exploit the rewards of mutual co-operation against other kind strategies. Secondly, the strategies can protect themselves from any selfish defecting strategies.

Tit-for-Tat (including its variations) and Pavlov have both achieved this, Tit-for-Tat will defect the next turn if a strategy defects against it this turn, and will only cooperate with it again if the opponent shows signs of cooperation, and continue cooperation until the opponents changes its mind again. If Pavlov is suckered, i.e. getting a gain of 0, it will not continue its original choice (cooperation) and change to defection, therefore protecting itself from any further exploits from selfish opponents.

In this project, we will use these two traits as a foundation, and take advantage of genetic algorithms to sculpt new strategies that will find the best balance between these traits, and any other beneficial traits that may exist. This is be discussed further in section 2.7.

2.6 Relevance to Genetic Algorithms

In a repeated game of Prisoner's Dilemma, we could observe from previous tournaments that the stronger algorithms are not necessarily the most selfish ones, such as always defect, but rather the ones that are kind and willing to cooperate.

In Richard Dawkins' book, *'The Selfish Gene'*, he underscores that through evolution of all organisms on earth, it is not survival of the fittest, strongest, most competitive, most selfish, but rather survival of the genes that doesn't preclude cooperation, survival of those that can engage in social cooperation. [Daw76]

To demonstrate this, we will take a look at a few examples from nature.

2.6.1 Example 1: Worker Bees

Worker bees are highly admired for their willingness to sacrifice their own lives in a battle to the death with an intruding enemy bee from a different colony to protect its own. Seeing as the genes of the bees within a colony are closely connected and almost identical, this ultimate act of altruism makes sense as it aids the genes of the current bees to live on in future generations. [Daw86]

2.6.2 Example 2: Birds and Parasites

Consider a population of birds, which are parasitized by infested ticks. Birds can remove ticks for other birds, but removing ticks aren't free as it costs energy and time. Mutual grooming is rewarding, but getting ticks removed and not repaying for it is cheating. The birds who groom others are called suckers, and those who get groomed but don't groom back are called cheats.

Thus, if we consider a scenario where only cheats and suckers exist. Cheats will drive suckers extinct in the long run as no one will groom the suckers and the cheats will get groomed. However, their victory is short-lived, as once all the suckers are gone, no cheat will groom each other, and the parasites will drive the cheats extinct in time. Therefore neither of these genes will persist through the process of natural selection in evolution. We need a better strategy that makes cooperation sustainable.

Thus consider a new type of bird, called the Grudger. Grudgers will always cooperate first, and if the other bird cheats it, it will remember that bird and never cooperate with it again. Therefore it is easy to see that Grudgers will drive both cheats and suckers extinct in time, and they become the dominant gene.[Daw86]

Conclusion
From these two real life examples, we understand that the strategy for survival is to be both cooperative and can identify deception from defectors. It is much like a game of chess, where the players carry around an arsenal of offensive and defensive strategies that they can deploy when the time is right.

Borrowing this concept that we have gathered from Richard Dawkins' examples, we fathom that the key to creating a strong agent in playing a game of Iterated Prisoner's Dilemma is using a few simple naïve strategies as an initial population, and then through genetic algorithm, select the advantageous traits of these strategies and evolve them into a stronger strategy, this is possible because genetic algorithms emulate the natural process of evolution. This will be the main aim of the project.

2.7 Current State of the Art and Employing Genetic Algorithms

Genetic Algorithms starts off with an initial population of chromosomes, which in this case will be the naïve and existing strategies/agents, these will each contain a candidate solution to the optimization to the Iterated Prisoner's Dilemma problem. The evolution process will happen in generations. In each generation the fittest individuals (which are stochastically selected) will breed and be modified to create a new population of individuals. The new population is then used in the next iteration of this algorithm. The process will cease once an acceptable fitness level has been achieved or the maximum number of generations has been attained.

We will now discuss some methods of how genetic algorithms should be applied that have already been suggested in other papers and look at some key components to this kind of solution that they have highlighted.

2.7.1 Evolutionary Algorithm

One simple way of using machine learning in the game of IPD that was suggested by Shashi Mittal of IIT is as follows. He suggested an extension to the works of Axelrod's method. Axelrod's method makes the next move at a given stage in the game by examining the behaviour of both the parties during the previous 3 moves. [Axe87]

There are 4 possibilities for each move (assuming we are player 1):

- CC: R for Reward
- CD: S for Sucker
- DC: T for Temptation
- DD: P for Penalty

Thus, we can encode the behavioural sequences as a 3-letter string. For example, RRR represents a history of where both players cooperated for the last 3 moves. SSP suggests the first player was played for a sucker for 2 moves, then realized and defected on the third move.

We can then use the 3-letter sequences and generate a number between 0 and 63 by translating each outcome to a number in base 4.

- CC: R for Reward = 0
- CD: S for Sucker = 2
- DC: T for Temptation = 1
- DD: P for Penalty = 3

Therefore RRR will decode as 0, and SSP will decode as 43 ($2 \times 16 + 2 \times 4 + 3 \times 1$). The strategy string is a 64-bit binary string of C's and D's where the ith bit corresponds to the ith behavioural sequence.

However the first 3 moves of the game are undefined in the scheme, so we add an additional 6 bits to specify a strategy's premises, i.e. assumption about the pre-game behaviour. Therefore together, each of the 70-bit strings thus represents a different strategy.

We will need to use Genetic Algorithms seeing as the total number of all possible strategies is very high – 2^{70}. In addition, the fitness function is non-continuous, so classical methods would not work. And GA also imitates the process of evolution.

We will use a Multi-Object GA. The reasoning behind this is because IPD is a non zero sum game (player's scores may not be directly related), and also there are 2 objectives, maximize your own score and minimize your opponents.

In Axelrod's approach, only maximization of your own score is done via Single Objective GA. Whereas in this suggested approach, maximization of your own score and the minimization of the opponents score are done separately using Single Objective GA. Thus, the search for optimal strategies is to optimize both objectives simultaneously and can be done using NSGA-II[4]. [DPAM00]

When the performance was ran for Single Objective GA (maximizing own score), the results were that these strategies always surfaced as the winner, defeating strong tactics such as Tit-for-Tat.

On the other hand, when using Multi Objective GA, the Pareto optimal front was obtained, there was a slight trade off between the own score and the opponent's score. The observations were they still emerged as winners, and often a winner by a significant margin. In addition, they beat strategy's that were obtained via Single Objective GA too. From this we can conclude that using Multi Objective GA is better suited in finding the optimal strategies in IPD.

Other researchers have also suggested other methods of applying GA, such as using Minimum Regret, which we will now discuss.

2.7.2 Genetic Algorithms to Combine Experts Advice

Expert Algorithms

Expert Algorithms are a collection of methods in machine learning. The aim of these methods is to learn from past experiences on how to combine different experts' advice to make successive decisions in a changing environment. The general idea is to let the algorithm choose repeatedly from a given set of strategies, and the reward at each stage will be a function of the chosen action and the choice of the opponent. Each strategy in the set is known as an "expert". Each stage the chosen expert will make the choice for which action to take, and the algorithm will monitor the payoff of that choice. The Expert Algorithm will conduct which expert to use next, this decision is dependent on the yields of those experts in the past.

Minimum Regret (MR)

A standard measure in the analysis and model of expert algorithms is called Minimum Regret, which is described as follows. Regret is defined as the difference of the reward

[4]NGSA-II is a famous multi-objective optimization algorithm. See citation[DPAM00].

that achieved against the reward that could have been achieved given the observed choice of the opponent. An expert selection rule minimizes regret if it returns an average reward that is at least as high as any single expert, against any number of fixed actions selected by the opponent. [MC05]

One downfall of MR is that since experts are compared on a sequence-by-sequence basis, MR doesn't take into account the possibility that different experts' choices now may have induced different sequences of the choices by the opponent later. Therefore, MR only really works well in an environment where the opponent's choices are independent of the other player's choices.

The Algorithm is as follows:

We denote $M_e(s-1)$ to be the average reward achieved by expert e prior to stage s of the game. Then we will choose to follow expert e's advice with a probability that is proportional to some monotone function of $M_e(s-1)$. Specifically, when the probability is proportional to $\exp\{\eta_s M_e(s-1)\}$, for a certain η_s, this algorithm is know to minimize regret. [APCBNS95][FS99]

This can be shown in the equation:

$$\frac{1}{s}\sum_{s'=1}^{s} E[R(i,j_s) : i \sim \sigma_X(h_s)] \geq \sup_s \frac{1}{s}\sum_{s'=1}^{s} E[R(i,j_s) : i \sim \sigma_e(h_s)] - o(s). \tag{2.7.1}$$

In the above formula, the player's actions that they can choose from are $i \in I$, and the opponents actions are $j \in J$. Reward matrix is R. The player at each stage may consider experts $\{1,\ldots,r\}$. σ_e denotes the strategy proposed by expert e, $\sigma_e(h_s)$ denotes the probability distribution over actions in stage s, given history h_s.

However, it is important to be mindful of the deficiency of MR that we have mentioned earlier. It fails to take into account the fact that player's actions now may influence the future choices of the opponent. The above equation (2.7.1) only holds for a fixed sequence of actions j_s from the opponent, it doesn't cater for the possibility that the opponent may have chosen different actions given that the agent provoked different actions.

This tact is also absent in the above mentioned experts algorithm. There is no concept for learning how an expert's actions affect the opponent's choices. For example, the algorithm will be unable to determine the connection between a Tit-for-Tat opponent's cooperative moves and its own.

Thus to overcome this shortcoming, we refine the algorithm by taking into account how the opponent reacts to each expert. We will now look at this approach in more detail.

Exploration-Exploitation Experts Method (EEE)

Another expert method is Exploration-Exploitation Experts Method, this method is especially designed for agents competing in a reactive environment. A reactive environment is one where your actions and choices now may affect the state of environment in the future. The EEE algorithm is built on with MR as a foundation. EEE algorithm employs different experts sensibly as it aims to boost the long-term average reward. It differs from the previous approaches because not only does EEE follow strategies in phases, but also it takes into account only the rewards that we actually achieved, rather than the rewards that could have been achieved. [MC06]

EEE is also a simple algorithm as the ones previously mentioned, yet it enjoys powerful performance in a game of IPD. The EEE algorithm takes advantage of the "exploration-exploitation" technique. This algorithm's objective is two-fold, it chooses between either aiming to explore and acquire knowledge, or it chooses to exploit and capitalize on the knowledge it has acquired to maximize the payoffs. It is important to realize that there is a trade-off between the exploration and exploitation phases of the algorithm.

Intuitively, we recognize that in order for the algorithm to learn and be effective in a given environment, some kind of blending needs to be present to ensure that any earlier 'mistakes' of the learning algorithm does no leave a lasting effect.

There has been a lot of work done around learning and adaptation for problems of consecutive decision making with uncertainty. Most of the work done has focused on environments that can be modelled as Markov Decision Processes (MDPs) with a known state. This method uses the MDP approach as a basis but extends it to consider adversarial environments. We will now analyse the EEE method.

The idea is instead of choosing an expert at each stage of the game, the number of stages each expert is followed each time it is selected, increases progressively. We will refer to these sequence stages "phases".

For the purpose of demonstrating how the algorithm works, the phase number will be expressed as i. The number of phases during which expert e has followed is N_e. Total number of stages in which expert e is followed is denoted by S_e. Average payoff from the phases where expert e was followed expressed as M_e. The general algorithm is as follows:

- *Exploration*: phase that picks a random expert e (i.e., from the uniform distribution of $\{1, ..., r\}$), and following e's advice for a certain number of stages depending on the variant of the method.
- *Exploitation*: phase that picks an expert e with maximum M_e, breaking ties by using randomness, and following e's advice for a number of stages depending on the variant of the method.

Exploration-Exploitation Experts Method:

1. Initialize $M_e = N_e = S_e = 0$ (where $e = 1, ..., r$) and $i = 1$

2. With probability p_i, perform an exploration phase, and with probability $1 - p_i$ perform an exploitation phase; denote by e_i the expert chosen to be followed and by n_i the number of stages chosen for the current phase.

3. Follow expert e_i's instructions for the next n_i stages. Increment $N_{e_i} = N_{e_i} + 1$ and update $S_{e_i} = S_{e_i} + n_i$. Denote by $\tilde{R}$ the average reward accumulated during the current phase of n_i stages and update:

$$M_{e_i} = M_{e_i} + \frac{n_i}{S_{e_i}}(\tilde{R} - M_{e_i})$$

4. Increment $i = i + 1$ and to to step 2.

Note that the two sets of parameters must be chosen to specify the experts algorithm are the exploration probabilities p_i, and the phase lengths n_i, for $i = 1, 2, \ldots$.

This new improved algorithm yields better results than the original simple MR strategy and provides a better basis of how genetic algorithms can be applied in a reactive environment.

With these different perspectives and standpoints of how genetic algorithms should be applied which we have gathered from different sources in hand, we can now build on top of their fundamentals and proceed appropriately.

Chapter 3

Project Plan

3.1 Objectives

The objectives of the project are as follows:

3.1.1 Coding the Environment

Creating a generic environment that will easily allow different strategies to join a Round-Robin game or IPD. The same environment should be extensible to allow for genetic algorithms to train and observe outcomes.

3.1.2 Coding Naïve and Existing Strategies

Initially populate strategies with the current array of proposed ones. In addition, they will also act as a benchmark for the trained algorithms to compete against to evaluate their performance

3.1.3 Using Genetic Algorithms

The main object of the project is to create an evolutionary algorithm that will be able to use an initial population of strategies, and evolve them to a stronger strategy that is a weighted combination and mutation of the basic strategies. It will take advantage of tactics of evolution mentioned previously, such as Explore and Exploit and Minimum Regret, when training.

3.1.4 Conclusion

Once the project is completed, the report will underline key findings of the project. It will consider the achievements as well as the failures and reason about the outcomes of the project. In addition, it will also propose where the project should head towards and further prospects if continued.

3.2 Timetable

Milestone No.	Description	Due Date
1	Coding environment and Engine	31st December 2011
2	Coding of Naïve existing strategies	31st December 2011
3	Testing of work done so far	31st January 2012
4	Project Review Deadline	8th February 2012
5	Employing genetic algorithms	1st March 2012
6	Testing GA, bug fixing	1st April 2012
7	Analysing outcomes, exploring other possibilities of work done	1th June 2012
8	Write up of full report	19th June 2012
9	Presentation	25th June 2012

3.3 Extensions and Fallback

If time permits, possible extensions of the project include exploring and extending to other proposed genetic algorithms. In addition to this, another possible extension is to try out other viable machine learning techniques and observing the results.

We could also revise the game format slightly, such as adjusting the different values of Sucker, Temptation, Reward, and Penalty, so that the ratios between them are smaller (or bigger) and correlating how that difference changes the optimal choice of the algorithms. In addition, we could analyse the results of when the tournament is changed from a Round-Robin tournament to any other type of tournament, such as double elimination.

Analyses of how cooperation is initiated when players 'communicate' through their actions. Exploring the two types of player's such – patient and impatient, and how they react with one another. Explore how altruistic and selfish behaviour traits pan out in the long run.

Another possible line of thought is: instead of always looking into the past of how different experts' strategies have played out, we could rather look into the future and try predict the possible decision tree of the opponents, then using that to seek for the best possible strategy.

The fallback of the project if time is limited will be encoding just the simple genetic algorithm and trying to outperform Tit-for-Tat or Pavlov algorithms.

Chapter 4

Evaluation Plan

4.1 Environment

Evaluation of the project will be firstly proving that the environment created is a suitable and easily extensible one that will allow the effortless injection of any new agents. The engine will allow any number of rounds of games to be played in a Round-Robin fashion, and amongst any number of competitors.

4.2 Genetic Algorithms

Secondly, we need to show a proof of concept that Genetic Algorithms is a suitable method of training more superior agents, showing that they do pick the traits of current strong naïve strategies and evolve them to stronger strategies.

4.3 Benchmark

We will use simple strategies such as Tit-for-Tat and Pavlov as benchmarks, and run experiments in the traditional Round-Robin tournament. In addition, if possible, we will test the trained genetic algorithms against other genetic algorithms proposed by other researchers of this field. This will show the successful extension of current state of the art strategies.

We will also adopt another method of measuring performance of players by a method suggested by Shashi Mittal of Indian Institute of Technology[MD05]. The benchmarking is done as follows:

- Benchmark Score: the maximum score – i.e. the score a player would have achieved against an opponent if both players always cooperated
- Then divide the score of the player by the number of players in the tournament, and express it as a percentage of the Benchmark Score.

Example: Suppose there are 15 players, each player plays against each other as well as itself, and each game is of 200 moves.

- Benchmark Score: $3 \times 200 = 600$
- A player scores 7500, then he has scored (500/600), i.e. 83% of the Benchmark Score in this case.

Bibliography

[APCBNS95] Freund Y. Auer P. Cesa-Bianchi N and Schapire. Gambling in a rigged casino. the adversarial multi-armed bandit problem. *In Proc. 36th Annual IEEE Symp. on Foundations of Computer Science, CA IEEE Computer Society Press*, pages 322–331, 1995.

[Axe87] R. Axelrod. *The evolution of strategies in the iterated prisoner's dilemma*. CA-Morgan Kaufmann, 1987.

[BW06] D. Begg and D. Ward. *Economics for business*. McGraw-Hill Education, 2006.

[Daw76] Richard Dawkins. *The Selfish Gene*. Oxford University Press, New York, 1976.

[Daw86] Richard Dawkins. Nice guys finish first (bbc horizon documentary), 1986.

[DPAM00] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2000.

[FS99] Y. Freund and Schapire. *Adaptive game playing using multiplicative weights. Games and Economic Behavior*. 1999.

[FT93] D. Fudenberg and J. Tirole. *Game theory*. MIT Press, 1993.

[LS05] McBurney Luck and Willmott Shehory. *Agent Technology. Computing as Interaction. A Roadmap for Agent Based Computing*. 2005.

[MC05] Daniela Pucci De Farias (MIT) and Nimrod Megiddo (IBM Almaden Research Center). How to combine expert (or novice) advice when actions impact the environment. *ACM*, 2005.

[MC06] Daniela Pucci De Farias (MIT) and Nimrod Megiddo (IBM Almaden Research Center). Combining expert advice in reactive environments. *ACM*, pages 762–799, 2006.

[MD05] Shashi Mittal and Kalyanmoy Deb. Optimal strategies of the iterated prisoner's dilemma problem for multiple conflicting objectives. *MIT Press*, 2005.

[OR94] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, MA, 1994.

[RO94] M. Rubenstein and M. Osborne. *A Course in Game Theory*. MIT Press, 1994.

[Sul03] Arthur Sullivan. *Economics : principles in action*. Prentice Hall, 2003.