1 Literature Review / Background

1.1 Introduction

This project is an attempt to combine multiple facets of game theory and artificial intelligence, in hope that the end result will be educational and/or useful to the fields. There are multiple stepping stones within the fields of game theory, mathematics, etc. which need to be examined in order to gain a further understanding of the project, which we will look at in this section.

1.2 Brief History of Game Theory

Game theory is the study of behaviours and mathematical models which result from the decisions and strategies of two or more economically rational players in either cooperative or non-cooperative strategy games. Applications of game theory have manifested in social science, psychology, mathematics and many more fields of study; however, the root interactions lie in strategic games such as the prisoner's dilemma or tit-for-tat. Game theory was introduced and popularised by mathematician John von Neumann, who first proved an optimal strategy for zero-sum games with perfect information such as chess or go called the minimax theorem in 1928. This theorem indicates that in such games, there is a pair of strategies for each player which allows them to minimise their maximum losses, while considering all responsive moves of the opponent.
After von Neumann published his initial paper on game theory, he published a book co-authored by economist Oskar Morgenstein entitled, "Theory of Games and Economic Behaviour". Within this book, von Neumann fixates mainly on non-cooperative games and/or zero-sum games; but most importantly, identified a method of finding consistent solutions and strategies for both players in two-person zero-sum games. This work became a milestone for game theory as it established a foundation for becoming a unique discipline.
Following this, numerous advancements in game theory occurred during the 1950s - mathematicians Merrill Flood and Melvin Dresher experimented mathematical and game versions of the prisoner's dilemma for the American think tank corporation, RAND (Research and Development). In the same year, John Forbes Nash Jr published his dissertation on non-cooperative games which contained the first definitions of the Nash equilibrium - an important milestone for adaptive strategy in game theory. He proved that in every n-player non-zero sum game, a Nash equilibrium existed, assuming the game had a finite number of actions. This was a continuation of the work from von Neumann and Morgenstein in their 1944 book, which only covered two person zero-sum games, and was restrained by the implications of 'rational' behaviour.
In 1980, political scientist Robert Axelrod set up a multi-agent tournament for the iterated/repeated prisoner's dilemma. Multiple well-known game theorists from different professions such as psychology, political science, economics, mathematics and more submitted 14 FORTRAN (Formula Translation) programs for the agents to follow as implicit strategies. In this tournament, agents would play against each other for 200 rounds - mutual cooperation would yield 3 points, mutual defection 1 point, single defection 5 points and single cooperation 0 points. The winning strategy was a simple tit-for-tat program which cooperated on the first turn, then repeated the opponents previous move for each subsequent turn. This strategy ended the tournament with an average of 504.5 points of a maximum 1000.

1.3 Prisoner's Dilemma

The prisoner's dilemma is one of the fundamental games of game theory which shows the payoffs and consequences of two 'players' acting in their own self interests. This summary, cited from britannica.com, is a model version of the prisoner's dilemma:

"*Two prisoners are accused of a crime. If one confesses and the other does not, the one who confesses will be released immediately and the other will spend 20 years in prison. If neither confesses, each will be held only a few months. If both confess, they will each be jailed 15 years. They cannot communicate with one another. Given that neither prisoner knows whether the other has confessed, it is in the self-interest of each to confess himself. Paradoxically, when each prisoner pursues his self-interest, both end up worse off than they would have been had they acted otherwise.*"

The first examples of the prisoner's dilemma being used in the context of game theory date back to the 1950s, by Merrill Flood and Melvin Dresher who devised puzzles and experiments using the structure of the dilemma - mainly an attempt to verify the usefulness of a non-cooperative Nash equilibrium. In this experiment, Flood and Dresher ran 100 games between two human players - in which player 1 (economist Armen Alchian) cooperated 68 times, while player 2 (mathematician John Williams) cooperated 78 times. In game theory, if a strategic game exists with the possibility for a various number of possible outcomes, a payoff matrix can be used to visually represent the benefits and consequences of each outcome. For the prisoner's dilemma, a typical payoff matrix would look as such:

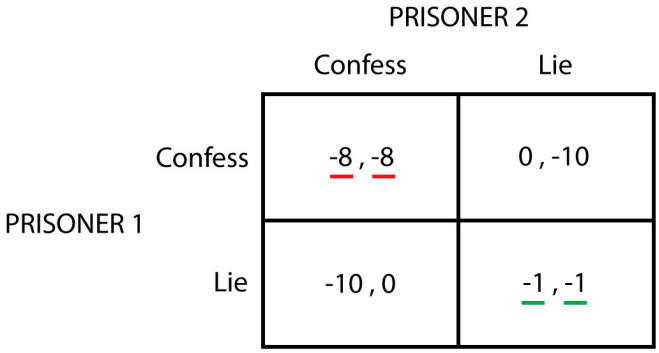

Figure 1: Prisoner's Dilemma Payoff Matrix

As you can see, the prisoner's would achieve the best possible equal payoff if they consistently chose to confess, but a prisoner could achieve a higher payoff if they were to follow their own self interests. However, the payoff matrix in this experiment looked like this:

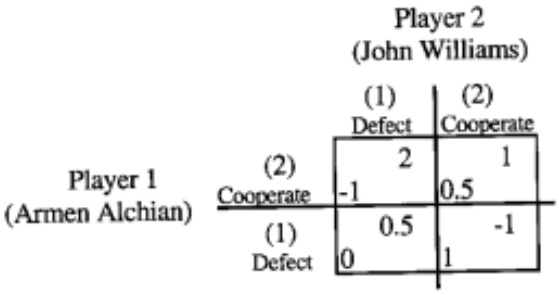

Figure 2: Flood-Dresher Experiment Payoff Matrix (de Herdt, 2003, p. 184)

In the Flood-Dresher experiment, the restraints can be thought of as 'unfair' as human players have a level of empathy and other emotion which may sway their decision for reasons an A.I. program would never follow. Such an example would be the comments which player 1 made in their log of comments. Alchian, or player 1, wrote comments such as "He does not want to trick me. He is satisfied. I must teach him to share", while player 2 Williams wrote comments such as "A shiftless individual - opportunist, knave" (de Herdt, 2003, p. 189) just a turn apart from each other. Many economists, game theorists and mathematicians believe that the results of this experiment may have been swayed slightly due to each player being empathetic or vindictive at numerous points in the game.

1.4 Nash Equilibrium

In 1950, John Forbes Nash Jr. published his dissertation entitled, "Non-cooperative Games". Within this dissertation was proof which indicated that within a two person zero-sum game, there exists an 'equilibrium point' for both players. This equilibrium was described in the paper as such, "Thus an equilibrium point is an n-tuple such that each player's mixed strategy maximizes his pay-off if the strategies of the others are held fixed. Thus each player's strategy is optimal against those of the others" (Nash, 1950, p. 3). Simply put, Nash was illustrating that within a two person game in which one player's benefit is a direct loss for the opponent, there lies a state in which neither player has any incentive to switch strategies, as it will not benefit their payoff - thus, the game sits at an equilibrium. The simplest, and most likely quickest way to prove the existence of a Nash equilibrium would be as follows:

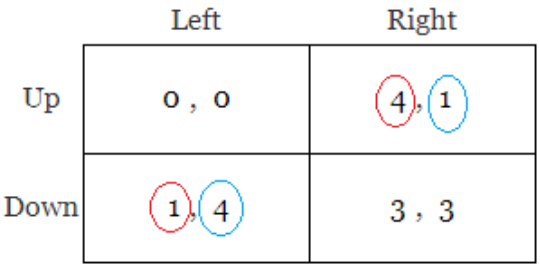

Figure 3: Example Matrix

Most proofs of equilibria exist if a certain number of conditions are met. Given a model payoff matrix, figure 3, our conditions for a pure/mixed strategy Nash equilibrium are as follows:

- The first player's best response is the same against any potential move of the opponent (red circle).
- The second player's best response is the same against any potential move of the opponent (blue circle).
- Nash equilibrium = (Up, Right),(Down, Left).

There are a few ways of proving the existence of a Nash equilibrium within games in a more detailed way - within Nash's dissertation, he chose to speak about the 'generalised' Kakutani fixed point theorem, and the Brouwer fixed point theorem. The Kakutani theorem is a more generalised proof of the Brouwer fixed point theorem, but is used to prove a Nash equilibrium in a very similar way. The conditions used in both theorems can be modified in such a way that you would prove the existence of an equilibrium state rather than a fixed point, within a set of strategies rather than tuples.

1.4.1 Applications of Game Theory

Beside its obvious magnitude in the fields of mathematics, Nash's work has had effects on fields such as computing, social science, psychology, and many more. Economists have used examples of the Nash equilibrium to calculate the prices of rival companies, predict prices of future products and calculate the best prices for supply and demand. A dissertation/report was published by the federal reserve bank of Minneapolis looking into why car insurance was so expensive in Philadelphia in the 90s, in which the writer chose to use a Nash equilibrium to demonstrate why the fluctuation of price was caused by the rivaling strategies of insurance providers. However, given the unpredictability of today's market, a Nash equilibrium may not have many uses outside of being a mathematical model in the field of economics. Another famous example would be the Cold War between the 40s and 90s - the USSR and the US were stuck in a long period of tension which could be seen as mutually assured destruction, in which each bloc knew the positions of the opponent but didn't start a war. This correlates exactly to a Nash equilibrium situation, where each side has no incentive to switch their strategy given the payoff. Aside from it's obvious applications in machine learning, game theory has appeared in subfields of computer science such as social networks, recommender systems and resource managers. Similarly, in the field of video games, game theory is becoming more prevalent - specifically in games where strategic decision making is involved, such as Firaxis Games' 'Civilization VI' or Ensemble Studios' 'Age of Empires'. In these games, it is entirely possible, and quite likely that players will end up in situations similar to the prisoner's dilemma or even a Nash equilibrium due to its resource management and turn-based system. In a 2-player game of Civilisation VI, both players have the possibility of knowing where the other's resources (soldiers/capitals/units) are, but have a mutual acceptance of not attacking each other (Cooperation). However, at any point in the game, either player could defect from this peace and attack the other in order to increase their payoff, maybe winning the game. In a field such as multi-agent systems, the prisoner's dilemma is a common occurrence when dealing with e-commerce situations such as auctions. For example, an auctioning site such as eBay has the issues of shill bidding and sniping taking over the market. The general process of an auction is a prisoner's dilemma - if the seller is viewed as the judge/prosecutor in the classic dilemma scenario and the buyer is viewed as the prisoner, there is a communication 'grey-area' in which neither side knows the true price of what is being sold, both have an incentive to overcorrect, leading to cooperative and

defective choices.

1.5 Repeated Games

Repeated games, also known as iterated games or 'supergames' are either finitely or infinitely long games which repeat after finishing. These games are usually represented in extensive form, meaning each strategy and/or game is mapped out as a tree, with specific time-stamps for each game. Payoffs are included at the end of each branch. The main application and usefulness of repeated games is to examine how economically rational players may behave differently from game to game depending on previous strategies or moves. Arguably the most important instance of repeated games is Robert Axelrod's multi-agent tournament in 1980. This tournament was a simple 200 round prisoner's dilemma, in which mutual cooperation scored 3 points, mutual defection 1 point, single defection 5 points and single cooperation 0 points - with a 200 round maximum of 1000 points. Well known game theorists from multiple professions such as psychology, political science, economics, mathematics and sociology submitted FORTRAN (Formula Translation) programs which the agents would follow as strategies. The winning strategy was submitted by Professor Anatol Rapoport, which was a simple tit-for-tat program in which the agent would start with a cooperative choice, then mimic the opponent's choice on the previous turn. According to Axelrod in his primer, "This decision rule is probably the most widely known and most discussed rule for playing the Prisoner's Dilemma. It is easily understood and easily programmed" (Axelrod, 1980, p. 7). Interestingly, each participant of the tournament was made aware of the properties of the preliminary tournament, and thus, many of them made tit-for-tat programs which they tried to improve upon; but, the original and simple tit-for-tat program ended up performing better than the modified versions. In relation to this project, FORTRAN would not be applicable/feasible to use, given its dependency on supervision (previous data or knowledge). Essentially, this means that it is incredibly difficult to evolve individual, high-level lines of code from non-functional languages such as FORTRAN, Java, C# and so on. Evolutionary algorithms essentially make random changes - thus, if this was replicated with random changes to lines of code in an imperative language, the program would undoubtedly become unstable or not work. The same can be said for any attempts to 'evolve' lines of code like an evolutionary algorithm. As a subsitute to this, the project will use neural networks and finite state machines to evolve a strategy, as both methods are reliable, efficient and have the capability to easily 'evolve'. Another interesting approach to this project would be to use Lisp, another high-level language which is functionally similar to FORTRAN.
Within the tournament there were 3 strategies which were expected and known by the participants in advance - always defect, always cooperate, and random. A strategy in which the agent always defects is the safest strategy of any, and could be seen as a principle of game theory. However, although such a strategy is safe, there is a low chance of it being the best strategy due to it's 'no risk, no reward' drawback. The 'always cooperate' strategy performs well when matched against itself - as you can expect a maximum payoff, but when matched against a defecting opponent, there comes a minimum payoff. The 'random' strategy is simply cooperating 50% of the time, in an attempt to reap the benefits of both strategies. This strategy is more of a utility for making sure the opponent's strategy accounts for all possibilities, but in the end the random strategy didn't perform well. Scores for these strategies can be seen in figure 4.

TABLE 2
Tournament Scores

Player	TIT FOR TAT	TIDEMAN AND CHIERUZZI	NYDEGGER	GROFMAN	SHUBIK	STEIN AND RAPOPORT	FRIEDMAN	DAVIS	GRAASKAMP	DOWNING	FELD	JOSS	TULLOCK	(Name Withheld)	RANDOM	Average Score
1. TIT FOR TAT (Anatol Rapoport)	600	595	600	600	600	595	600	600	597	597	280	225	279	359	441	504
2. TIDEMAN AND CHIERUZZI	600	596	600	601	600	596	600	600	310	601	271	213	291	455	573	500
3. NYDEGGER	600	595	600	600	600	595	600	600	433	158	354	374	347	368	464	486
4. GROFMAN	600	595	600	600	600	594	600	600	376	309	280	236	305	426	507	482
5. SHUBIK	600	595	600	600	600	595	600	600	348	271	274	272	265	448	543	481
6. STEIN AND RAPOPORT	600	596	600	602	600	596	600	600	319	200	252	249	280	480	592	478
7. FRIEDMAN	600	595	600	600	600	595	600	600	307	207	235	213	263	489	598	473
8. DAVIS	600	595	600	600	600	595	600	600	307	194	238	247	253	450	598	472
9. GRAASKAMP	597	305	462	375	348	314	302	302	588	625	268	238	274	466	548	401
10. DOWNING	597	591	398	289	261	215	202	239	555	202	436	540	243	487	604	391
11. FELD	285	272	426	286	297	255	235	239	274	704	246	236	272	420	467	328
12. JOSS	230	214	409	237	286	254	213	252	244	634	236	224	273	390	469	304
13. TULLOCK	284	287	415	293	318	271	243	229	278	193	271	260	273	416	478	301
14. (Name Withheld)	362	231	397	273	230	149	133	173	187	133	317	366	345	413	526	282
15. RANDOM	442	142	407	313	219	141	108	137	189	102	360	416	419	300	450	276

Figure 4: Scores from the Axelrod Tournament (Axelrod, 1980, p. 11)

1.5.1 Folk Theorem

Folk theorem is used within repeated games to show that a Nash equilibrium outcome in a game which is repeated infinitely is quantitatively and qualitatively equal and rational to that of a single game. While the origin of this theorem is unknown, it appeared in the 1950s and was quickly spread through the game theory field - thus the name, Folk Theorem. The first instance of a research paper to use the theorem was authored by James W. Friedman (1971) in his article, "Non-cooperative Equilibrium for Supergames", in which he details the payoffs of subgame-perfect [1] Nash equilibria in an infinitely repeated game, instead of using a single Nash equilibrium. This means that within a game such as the prisoner's dilemma, where mutual defection is a Nash equilibrium, folk theorem allows the possibility of a non-defection Nash equilibria in infinitely repeated games. Another important concept of folk theorem is that of duopolies and oligopolies; in an economic circumstance, a duopoly is a point in which two suppliers own all or nearly all of the market for a product or service. An oligopoly is the same, except the number of suppliers is more than 2 but remains a small number. When applied to the prisoner's dilemma, any choice other than mutual defection is unstable - however, if the games are infinitely repeated, there exists a possibility that one player may 'threaten' the other player to defect, in which case they would always play defect from that point onwards. In such a situation, if the second player is aware of this threat, they may choose to collude with their opponent and play cooperate, assuming there is a beneficial payoff guaranteed. In both economics and game theory, you can see a riskier, higher payoff as 'discounted' when colluding. Given that scenarios using the folk theorem are infinitely repeated, there are possibities of replicating the results with an indefinitely-long game, as long as there was a high chance of success that an agent would follow the same strategy. As a final thought in reference to Axelrod's tournament, it is confusing that Axelrod didn't consider or even mention folk theorem in his findings. Given the structure of the tournament, there are definite foundations for subgame Nash equilibria, which would've made for interesting comments from Axelrod on the value of specific equilbria at various points of the tournament.

[1] A subgame-perfect equilibrium means that if players were playing a smaller game which was part of a bigger game, their behaviour would represent a Nash equilibrium of that smaller game.

1.6 Machine Learning

Machine learning is a subfield of articifical intelligence which combines pattern recognition and computational learning theory, an idea pioneered by Alan Turing in 1950, and developed by Arthur Samuel in 1959 through his paper, "Some Studies in Machine Learning Using the Game of Checkers".[2] The main goal of machine learning is for algorithms to become 'smarter' on each iteration of instructions, such as a move in a game of checkers, to then make predictions based on the data it has constructed. Within Samuel's introduction to his paper, he states, "The studies reported here have been concerned with the programming of a digital computer to behave in a way which, if done by human beings or animals, would be described as involving the process of learning" (Samuel, 1959, p. 1). Samuel's checkers algorithm used a search tree to identify each of the board positions reachable from the available pieces[3], which would feed into a scoring algorithm - incorporating von Neumann's minimax strategy to choose the best move. The result of this algorithm, through rigorous testing, was a piece of artificial intelligence which could, "greatly outperform an average person", and was envisioned to be economically viable in real-life situations/problems. This is the first instance of an algorithm which has developed itself without being given information directly. Since the 1950s, machine learning has become almost ubiquitous; in Pat Langley's paper, "Applications of Machine Learning and Rule Induction", he shows applications of machine learning in over 15 different fields - such as economics, insurance, astronomy and more. Given the two main methods of learning, supervised and unsupervised, only unsupervised learning is applicable to this project. Supervised learning requires prior data in order to predict patterns; Instead, representations of cognitions will be used, as explained in the following section(s). The algorithm will eventually learn and get better with each generation, which is unsupervised.

1.7 Evolutionary Algorithms

Evolutionary algorithms, or evolutionary computation, is a facet of artificial intelligence in which an algorithm filteres through a set of data, removing the least fit values within a specified iteration, while keeping the most fit values until better values are found. The first appearance of a theory for automated problem solving originated in the 1950s, while application for said theory was developed in the following decade by Lawrence J. Fogel. Fogel devised the idea of evolutionary programming while working for the National Science Foundation, when he saw the approach to heuristic algorithms and primitive neural networks simulations as limited. He theorised that in order for artificial intelligence to progress, the approaches to simulating behaviour should be focused on evolution and increasing intellect rather than model human behaviour. Primarily, "Fogel considered intelligence to be based on adapting behavior to meet goals in a range of environments" (De Jong, 2014, p. 2). This meant that simulated experiments using a finite state machine could be used to investigate intelligent behaviours in a range of situations, with certain rules in consideration. Upon experimenting and publishing multiple papers on the topic, Fogel described, at a very general level, behaviour as the composite ability to predict one's environment, and thus adapt/respond to it suitably. According to Kenneth De Jong of George Mason University, Fogel made a similar proposal for a finite-state machine to replicate this behaviour to that which follows:

[2] While some may argue that Marvin Minsky pioneered the first instance of a self-learning machine in 1951, many still question whether or not this project was artificial intelligence, given the amount of missing information.

[3] Samuel described this process as 'looking ahead a few moves' like a human player might do.

"*A population of finite-state machines is exposed to the environment, that is, the sequence of symbols that have been observed up to the current time*[4]*. For each parent machine, as each input symbol is offered to the machine, each output symbol is compared with the next input symbol. The worth of this prediction is then measured with respect to the payoff function (e.g. all–none, absolute error, squared error, or any other expression of the meaning of the symbols). After the last prediction is made, a function of the payoff for each symbol (e.g. average payoff per symbol) indicates the fitness of the machine.*"

The general process involved picking a 'best' or best-suited machine to predict new symbols in the environment, and then the process is repeated until the payoffs can't increase any further, relative to the accuracy (or worth) of their previous predictions. Further work on this research in fields such as sequence prediction, pattern recognition and gaming was conducted by academics such as Bernard Lutter and Ralph Huntsinger (1968), Akihiro Takeuchi (1980) and many others.

Since these first occurrences of evolutionary programming, many strides have been made in the field - taking it in a number of directions such as neural network training, image processing and general computing optimisation. Within the facet of neural networks, foundational and important work comes from researchers such as Peter Angeline in 1994 with his paper on evolutionary algorithms and neural networks, J.R. McDonnell and D. E. Waagen's paper on evolving recurrent perceptrons, and Vincent Porto's paper on alternative neural network training methods. In Angeline's paper, he explains why the previous methods of constructing/modifying neural networks are limited - Timur Ash authored a paper on dynamic node creation, but Angeline outlines that only feedforward networks work in application (which isn't effective as a standardised method).[5] Another researcher of neural networks, Scott Fahlman, authored a paper explaining a recurrent version of the Cascade-Correlation learning architecture in the context of neural networks, but assumes a restricted form of reccurence according to Angeline, which limits the types of input/output, and is generally less robust. A version of the cascade-correlation learning architecture can be seen as follows:

[4]For the sake of generality, the environment was described as a sequence of symbols taken from a finite alphabet.

[5]A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle.

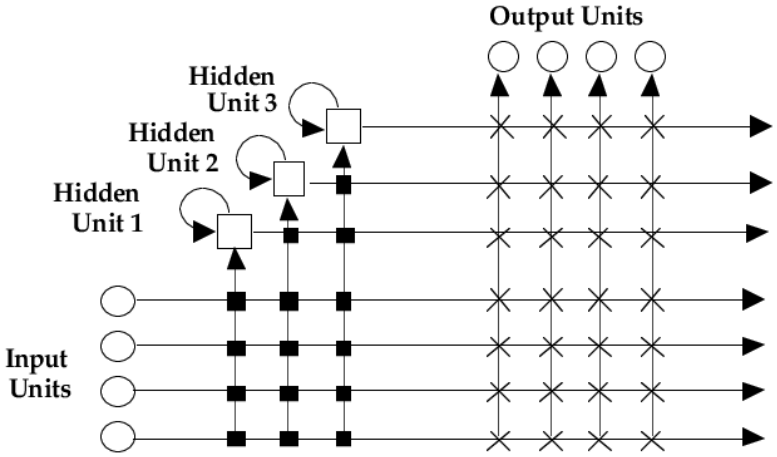

Figure 5: Cascade Correlation Learning Architecture

Finally, Dong Chen authored a paper on the constructive learning of recurrent neural networks with the help of C.L. Giles, G.Z. Suna, H.H. Chen, Y.C. Lee and M.W. Goudreaub. However, Angeline explains that the paper only explores fully connected topologies (in which all nodes are connected), without accounting for any other possibilites. Considering these limitations, Angeline presents GNARL (GeNeralized Acquisition of Recurrent Links), "a network induction algorithm that simultaneously acquires both network topology and weight values while making minimal architectural restrictions and avoiding structural hill climbing" (Angeline, 1996, p.2). This is essentially an evolutionary algorithm which is 'better suited' for evolving neural networks as opposed to genetic algorithms, as it has the boone of function optimisation and the ability to fit a variety of tasks/problems. Angeline then goes on to detail the process of evolving neural networks with both genetic and evolutionary algorithms, and eventually his own 'GNARL' algorithm, which "nonmonotonically constructs recurrent networks to solve a given task". Essentially, this algorithm decides the number of input and output nodes relative to the task it has been given, and the number of hidden nodes can range from 0 to whatever the user specifies - avoiding the limitations of the previous work mentioned before.

Following this work, one of the next milestones to occur in the field of evolutionary algorithms was the work of J. R. McDonnell and D. E. Waagen. This paper was centered around providing an alternative to feedforward networks for 'nonlinear models of time-series data', by creating the s imple parts of evolutionary algorithms and neural networks in a more efficient and effective way, so that they can be applied to more complex structures or networks. Specifically, the authors decided that, "Once feasibility is demonstrated for simple recurrent perceptron structures, the evolutionary search method can be applied to highly recurrent perceptron networks with complex architectures" (McDonnell; Waagen, 1994, p. 1). This is an interesting way to look at this topic as it can be applied to a large number of tasks/problems, given that the method doesn't address a specific topology - such as stochastic, recurrent or hierarchical. Pseudocode for a generic evolutionary algorithm can be seen as follows:

Algorithm 1 Basic Evolutionary Algorithm

```
1: procedure GeneticAlgorithm(S - set of blocks)
2:     Initialisation()
3:     t ← 0
4:     Initialise P_t with random individuals from S*
5: Evaluate-Fitness-GA(S, P_t):
6:     while termination condition not met do
7:         Select values from P_t (fitness proportionate)
8:         Recombine individuals
9:         Mutate individuals
10:        Evaluate-Fitness-GA(S, modified individuals)
11:        P_{t+1} ← newly created values
12:        t ← t + 1
   return (Superstring derived from best individual in P_t)
13:
14: Evaluate-Fitness-GA(S - set of blocks, P - population of individuals):
15:    foreach individual i ∈ P do
16:    generate derived string s(i)
17:    m ← all blocks from S that are not covered by s(i)
18:    s'(i) ← concatenation of s(i) and m
19:    fitness(i) ← 1 / ||s'(i)||^2
```

1.7.1 Artificial Neural Networks

Artificial neural networks (ANN) are information processing systems based on biological nervous systems such as the brain. Neural networks are a subfield of machine learning as they make decisions and perform tasks based on previous information they have been given, called training examples. Each system has three layers - input, output, and a layer for filtering the input to something which can be used by the output layer. This is possible due to an algorithm called backpropagation (backward propagation of errors), which calculates the gradient of the error function[6] with respect to the weights of the network. When designing a neural network, there are various paradigms or methods which can be used - only one of which is applicable for this project: control tasks. The other being classification, but this requires labeled training data to train the network, which this project will not have. Backpropagation and supervised learning also isn't valid/feasible within this project, as there is no prior training data to filter/modify to the output layer. Control tasks are reward-based behaviours which can train the decision making process of an evolutionary algorithm. With a neural network being the controller in question, a control task would reward correct behaviour as a method of training the network. Classification is the process of grouping similar values within the network based on their attributes/characteristics. Assuming the possbility of a 'noisy' data set at any point during the program lifecycle, this method would allow the network to classify patterns from data which they have not yet been fed/trained.

1.7.2 Finite State Machines

Finite state machines, or finite state automaton, are computational models used to simulate sequential logic or solve problems relating to software architecture. At a very basic

[6]The Gauss error function is the integral of the standard normal distribution.

level, computers can be seen as state machines; each instruction that a computer receives and execute will change it's state in some way, altering the behaviour and causing subsequent actions or allowing further instructions to occur. Arguably the pioneers of finite state machines, the first proposal and description of finite automata came from neurophysiologists Warren McCulloch and Walter Pitts in 1943 with their paper, "A Logical Calculus Immanent in Nervous Activity". Within this paper, McCulloch and Pitts comprehensively cover topics such as neural network theory, the theory of automata and the theory of computation and cybernetics. Around 10 years later, the first implementation of the finite state machine appeared, created by computer scientists G. H. Mealy and E. F. Moore, with their Moore and Mealy machines. The Mealy machine is focused on determining ouput through the input and current state, while the Moore machine bases the output on the current state alone. Another notable paper which covers finite state machines is Matthijs van Veelen and Julián García's paper, "Direct reciprocity in structured populations". Within this paper, van Veelen and García tackle the topic of finite state automata within an 'open-ended, infinite strategy space'. This topic encompasses that, "Our simulations contain a mutation procedure that guarantees that every finite state automaton can be reached from every other finite state automaton through a sequence of mutations. Thus, every strategy that can be encoded by a finite state automaton is a possible mutant" (van Veelen; García, 2012, p. 9929). This is done because the state machines that require fewer mutations are essentially the best fit. Nowadays, one of the most significant applications of finite state machines is within video games; games ranging from basic like Pacman to incredibly detailed such as Horizon: Zero Dawn (H:ZD) both use finite state machine for their AI agents. In a game such as H:ZD, creatures will start in a state such as 'idle' or 'roaming' to simulate random animal behaviour, but will switch behaviours to something like 'hunt' once a player is spotted. Other important applications of finite state machines include things such as pattern searching in text editors or IDEs, vending machines, or even system/software modelling.

1.8 Research Questions

Although a lot of content has been created through many years of research into game theory, machine learning and evolutionary algorithms, there are still a lot questions yet to be answered. In this project, I aim to tackle some of these unanswered questions, with topics such as these:

1. Given the results of research papers in the past such as Nash's, Harrald's or Axelrod's, how do those results look now when used within a model of more realistic computational power?

2. When using different representations of an agents' cognition, how does the payoff vary between evolved strategies from neural networks and finite state machines? How easy is it to evolve those representations at a high payoff?

3. Within an evolutionary algorithm, which neural network topologies are favoured in 2-player games when a neural network is being used as a representation of the agents' cognition(s)?

4. During this project, neural networks will be used with a fixed shape and payoff matrix. If it is allowed to evolve, how does the shape of the neural network change? (Optional)

1.9 Conclusion

In summary, we have analysed multiple facets of fields such as game theory, machine learning, neural networks and evolutionary algorithms in terms of their history, use and sometimes applicability to video games and other forms of media. In all cases, each field has some form of possibility for being in video games - machine learning, neural networks and evolutionary algorithms can all be applied to AI agents to improve their realism and general behaviour. Papers from researchers such as Nash, Harrald and Axelrod have avenues of further research, and while there are continuations of said work, there is yet to be solid proof of applications similar to this project. The results of this project will touch on applications to specific facets of video games and possibly computer science which may not have been explored before. Research for this project arguably starts with Nash's paper in the 50s - the existence of the Nash equilibrium was the driving force behind this project, which eventually led me to read more into game theory. This led to papers from researchers such as Axelrod and Harrald; work from these reports was instrumental in aiding my understanding game theory from a theoretical standpoint, but not necessarily from an application standpoint. The next step was to read work from Kakutani and Brouwer, then Flood and Dresher. While the Kakutani and Brouwer papers were helpful in understanding the processes behind finding a Nash equilibrium, the Flood-Dresher paper was helpful in understanding how game theory scenarios can be applied to real life situations.

With this knowledge of game theory and the Nash equilibrium in mind, a necessary step was to find ways to apply it in application. The first works to read were that of Fogel, Waagen and Porto. These papers comprehensively covered neural networks, such as their creation, evolution and learning methods. While helpful, it was the later papers from Fogel and works from researchers such as Chen, Ash and Fahlman that solidified my knowledge on the operation of neural networks. However, what was lacking was a comparitive component. Very little, if any, papers did a comprehensive comparison of methods for game theory application, so it seemed beneficial to tackle that exact topic. With a topic in mind, I came across papers from Angeline, van Veelen and Jong which were extremely useful for learning the history and applications of both machine learning and evolutionary algorithms - another possible method of game theory application. I had to find more papers to read to make sure that a comparison of finite state machines and neural networks within evolutionary algorithms was a fair avenue of research, which led me to papers from Samuel, Lutter and Huntsinger, and Langley. This finalised my understanding of how to approach the project, as well as possible research questions, which you can see above. This project will be an amalgam of the knowledge gained from all of these papers, in an effort to tackle the questions which previous papers did not ask or answer.

References

[Angeline et al., 1994] Angeline, P., Saunders, G., and Pollack, J. (1994). An evolutionary algorithm that constructs recurrent neural networks. Report, Ohio State University.

[Ash, 1989] Ash, T. (1989). Dynamic node creation in backpropagation networks. *Connection Science*, 1:365–375.

[Axelrod, 1980] Axelrod, R. (1980). Effective choice in the prisoner's dilemma. Report, Sage Publications, Inc.

[Axelrod, 1984] Axelrod, R. (1984). *The Evolution of Cooperation*. Basic Books Inc, New York.

[Aziz et al.,] Aziz, D. A., Cackler, J., and Yung, R. Basics of automata theory. Stanford University report detailing automata theory/finite state machines.

[Binmore, 2005] Binmore, K. (2005). *Natural Justice*. Oxford University Press, Oxford.

[Chen et al., 1995] Chen, D., Giles, C., Sun, G., Chen, H., Lee, Y., and Goudreau, M. (1995). Constructive learning of recurrent neural networks. *IEEE Transactions on Neural Networks*, 6:1196–1197.

[Chen et al., 1998] Chen, J., Lu, S., and Vekhter, D. (1998). Game theory. Report, Stanford University.

[Duignan,] Duignan, B. Prisoner's dilemma. Website article detailing game theory.

[Fahlman, 1991] Fahlman, S. (1991). The recurrent cascade-correlation architecture. Report, Carnegie Mellon University.

[Fogel et al., 1995] Fogel, III, D., C., E., and Boughton, E. (1995). Evolving neural networks for detecting breast cancer. *Cancer Letters*, 96:49–53.

[Friedman, 1971] Friedman, J. (1971). A non-cooperative equilibrium for supergames. Report, Oxford University.

[Gifford,] Gifford, A. Payoff matrix in economics: Theory & examples. Useful website detailing payoff matrices.

[Harrald and Fogel, 1996] Harrald, P. G. and Fogel, D. B. (1996). Evolving continous behaviours in the iterated prisoner's dilemma. Report, Manchester School of Management.

[Herdt, 2003] Herdt, T. (2003). Cooperation and fairness: the flood–dresher experiment. *Review of Social Economy*, 61:184–191.

[Jong et al., 1997] Jong, K., Fogel, D., and Schwefel, H.-P. (1997). A history of evolutionary computation. Report.

[Kakutani, 1941] Kakutani, S. (1941). A generalization of brouwer's fixed point theorem. *Duke Mathematical Journal*, 8:457–459.

[Kaznatcheev,] Kaznatcheev, A. Short history of iterated prisoner's dilemma tournaments. Website detailing iterated prisoner's dilemma tournaments.

[Knight,] Knight, V. Background to axelrod's tournament. Website detailing Robert Axelrod's tournament.

[Langley, 1995] Langley, P. (1995). Applications of machine learning and rule induction. Report.

[Levine,] Levine, D. What is game theory? Website detailing general game theory principles.

[Lutter and Huntsinger, 1968] Lutter, B. and Huntsinger, R. (1968). Engineering appllications of finite automata. *Sage Journals*, 47:264–265.

[McCulloch and Pitts, 1943] McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

[McDonnell and Waagen, 1994] McDonnell, J. and Waagen, D. (1994). Evolving recurrent perceptrons for time-series modeling. *IEEE Transactions on Neural Networks*, 5:24–38.

[Nash, 1950a] Nash, J. (1950a). Non-cooperative games. Report, Princeton University.

[Nash, 1950b] Nash, J. (1950b). Two-person cooperative games. Report, Air Force Project RAND.

[Policonomics,] Policonomics. Game theory iii: Folk theorem. Website detailing folk theorem.

[Porto et al., 1995] Porto, V., Fogel, D., and Fogel, L. (1995). Alternative neural network training methods. *IEEE Expert: Intelligent Systems and Their Applications*, 10:16–22.

[Ross,] Ross, D. Game theory. Website detailing game theory principles.

[Samuel, 1959] Samuel, A. (1959). Some studies in machine learning using the game of checkers. *IBM Journal*, 3:535–554.

[Smith and Wright, 1991] Smith, E. and Wright, R. (1991). Why is automobile insurance in philadelphia so damn expensive? Report, University of Essex.

[Smith, 1982] Smith, J. (1982). *Evolution and the theory of games*. Cambridge University Press, Cambridge.

[Stergiou, nd] Stergiou, Christos. Siganos, D. (n.d.). Neural networks. Report, Imperial College London.

[Veelen et al., 2012] Veelen, M., García, J., Rand, D., and Nowak, M. (2012). Direct reciprocity in structured populations. 109:9929–9934.