An Investigation into the use of Genetic Algorithms in RTS Games

BSC (HON) COMPUTER GAMES TECHNOLOGY

Finlay McGrath

2017

Contents

[i. Abstract 2](#_Toc480227517)

[ii. Abbreviations 3](#_Toc480227518)

[1. Introduction 4](#_Toc480227519)

[1.1. Research Question 5](#_Toc480227520)

[1.2. Aims and Objectives 5](#_Toc480227521)

[2. Literature Review 6](#_Toc480227522)

[2.1 Background 6](#_Toc480227523)

[2.2 AI Techniques 7](#_Toc480227524)

[2.3 Summary 8](#_Toc480227525)

[3. Methodology 10](#_Toc480227526)

[3.1 Gameplay 10](#_Toc480227527)

[3.2 Rule Based AI 11](#_Toc480227528)

[3.3 Genetic Algorithm AI 12](#_Toc480227529)

[4. Results 15](#_Toc480227530)

[5. Discussion 16](#_Toc480227531)

[6. Conclusion and Future Work 17](#_Toc480227532)

[7. Appendices 18](#_Toc480227533)

[7.1 Units 18](#_Toc480227534)

[7.2 Buildings 18](#_Toc480227535)

[7.3 Technology 19](#_Toc480227536)

[7.4 Rule Based AI Priority List 19](#_Toc480227537)

[7.5 List of Actions 19](#_Toc480227538)

[8. References 20](#_Toc480227539)

[9. Bibliography 22](#_Toc480227540)

# i. Abstract

* Usually read first by the reader
* Best to actually write this last
* Summary of what you did, your results and conclusions
* It is not an introduction. No references
* 300 words long

# ii. Abbreviations

RTS – Real Time Strategy

AI – Artificial Intelligence

FPS – First Person Shooter

# 1. Introduction

Artificial Intelligence (AI) is an important part of how players interact with a game. However, frequently a computer controlled actor is underwhelming and can detract from the experience. This can be shown by a large spectrum of issues that range from the simple to solve such as AI actors running into walls to more complex issues such as predictability and being easy to exploit by a human player. This leads to situations where developers of games such as the Sid Meier’s Civilization series give the AI unfair advantages that the human player does not possess. This merely delays the problem as it forces players who wish to challenge the most difficult of opponents to rely entirely on exploiting weaknesses in the AI’s rules without the AI being able to adapt. This simply exacerbates the problem creating a cycle where a player finds an exploit, the developer gives the AI an unfair advantage therefore meaning the player must take advantage of the found exploit to keep up with the AI. At the same time, AI advantages can feel unfair to human players despite the natural advantages the human player possesses such as creativity and the ability to plan.

There are generally two main issues in creating good game AI. The first issue is in being able to teach an AI how to play the game well enough to provide a challenge to the player. When compared to a game such as chess which has only 32 pieces on the board and a limited number of moves, an RTS (Real-Time Strategy) game has thousands of actions and millions of possible permutations. Each action creates a new branch of possible variations of the game. Games that rely on the movement of a unit can be the most complex as each unit in the game can be moved to each location in the game creating countless numbers of decisions a player can make. The difficulty is that previous rule based systems are very limited as it requires the developer to predict each of those permutations for the AI to provide a response. This means that some kind of machine learning is necessary.

The second issue is in ensuring that the created AI is enjoyable for a human player to play against. This means that it must adhere to a human sense of fairness and it must be beatable yet still provide a challenge sufficient for the player to enjoy. One of the ways this is done in RTS games is by providing multiple levels of difficulty to cater to different skill levels. However, in the higher difficulties this frequently involves the AI receiving advantages that the human player does not. The advantages provided vary depending on the game but in an FPS (First Person Shooter) it could auto-aim meaning the AI would never miss, or in an RTS the AI could be provided with additional units or given stronger units.

RTS games usually involve two or more players competing over resources with the goal being to destroy the opponent. Each player can gather resources which are used to build buildings and to train units from those buildings. A unit can travel across the game map, which is generally specially designed for that game. Units must attack the opposing player’s units and buildings to destroy them and eventually defeat their opponent. RTS games can be symmetrical or unsymmetrical where in symmetrical games each player has the exact same capabilities as the other player. In unsymmetrical games, such as Blizzard Entertainment’s Starcraft II it is possible for a player to have access to a completely different roster of units and buildings than the opposing player. The goal of this project is to create an RTS AI that is capable of learning how to win while only having the same information a human player, without being exploitable. This would then be compared to a rule based AI such as used in Warcraft 3 or Age of Empires II in which the AI follows a rigid set of rules that a pre-set by the designer and cannot change without direct human input.

A genetic algorithm could be used for this purpose as they are very good at finding near optimal solutions to a problem (Simon Mardle and Sean Pascoe, 1999) without a reliance on unfair advantages. While doing so, it also creates many sub-optimal solutions meaning it can be very useful for creating an AI of varying abilities. This allows it to be used for multiple difficulties for players of different skill levels making it ideal for an AI in an RTS that can easily be retrained whenever changes are made to the game. The intention and expectation would be that the learning Genetic AI would be superior and would beat the rule based AI after a sufficient learning period. The gameplay would be symmetrical in nature with each player being given the exact same capabilities, units and buildings as the other player.

This project would aim to create an AI that is challenging yet beatable and operates on the same level as a human player without receiving unfair advantages. This will be done using a genetic algorithm that uses the fitness function to find the correct way of playing against another AI.

## 1.1. Research Question

* Can a genetic algorithm be used to create a more adaptable and enjoyable experience for a human player when facing a computer player?

## 1.2. Aims and Objectives

* Research the application and use of genetic algorithms.
* Investigate how AI techniques are used in current RTS games for AI.
* Create an AI that can learn how to play an RTS game using genetic algorithms.
* Compare the genetic algorithm to other methods of AI in games.

In section two this dissertation will discuss the sources used to build evidence for the most effective method for creating a more adaptable AI for RTS games that is superior to the rule based AIs that are generally used in RTS games. It will then show in section three how this can be demonstrated by creating a game where the genetic algorithm AI trains and learns from a rule based AI. Results of this process are shown in section four and there is a discussion of what the results mean and how effective this process has been in section five. A further analysis on the project has been made to debate the overall success of the project and how it could be improved on in the future as part of section 6.

# 2. Literature Review

## 2.1 Background

Since the dawn of industrialisation, machines have been used to automate processes which previously would have been done by a human. The modern version of this is robots which are used in factories to carry out tasks at a far more reliable and relentless rate than was previously possible. These techniques, while effective within their operational parameters, do have their limitations. For example, if the robot is taken out of its operational environment, it would not be able to function as it is an unintelligent machine. A potential solution to this problem would be to find a way for the robots to adapt to its current environment through learning.

Artificial Intelligence (AI) is a highly active area of research in computing and robotics with a “'Partnership on AI' formed by Google, Facebook, Amazon, IBM and Microsoft” to advance public understanding of AI (The Guardian 2016). During 2016, Google’s Deepmind AI defeated the world Go champion in a best of five series of games (BBC 2016) using AI learning techniques that enabled it to learn how to play.

AI has been used to great success in several industries such as online shopping where Amazon uses AI both for their website and in the physical logistics of moving goods in a warehouse. Amazon uses machine learning to predict what customers want, sometimes before they know they want it. It is also used for creating a more efficient customer support, meaning the customer can solve their issue without contacting an actual member of customer support. On the physical side, Amazon has used machine learning to maximise efficiency of staff when they are retrieving items from the shelves in a warehouse by optimising routes between objects.

Figure 1 Starcraft 2. Blizzard Entertainment

In the games industry, AI has not advanced significantly since the 1990s with games like Starcraft (1997) and Starcraft II (2009, figure 1) using the same rule based AI despite a twelve-year time gap and enormous advances in other areas such as graphics. One of the main reasons for this is due to the complexity of many video games as they can contain hundreds of variables at a time. This means that a brute force algorithm such as used by IBM’s Deep Blue in 1997 to defeat the then chess grandmaster Garry Kasparov is not feasible. This is due to it requiring enormous processing power to the exclusion of all other aspects of gameplay and even then, it would not be possible for a modern RTS (Real Time Strategy) game.

## 2.2 AI Techniques

There are several AI techniques which have been used successfully in video games in the past. The ones which will be discussed are neural networks, fuzzy logic, rule based systems and genetic algorithms. A neural network is a form of machine learning that is based on the human brain. A brain is made up of many thousands of neurons which are a linked together. Each neuron can be described at the most basic level as taking in an electrical input, doing some process and then outputting the result. This idea was given a mathematical basis in *A Logical Calculus of the Ideas Immanent in Nervous Activity* (McCulloch and Pitts, 1943). This process can be transferred to AI by replicating the connections between artificial neurons defined as functions and the network can learn through training on a model by example. The training phase adjusts weightings of actions that define the likelihood of an action being taken, based upon the example that it is training on. The main flaw of this methodology when applied to games is that the AI can only ever be a copy of the player that the AI is training against. This means that the if the player is bad at the game so will the AI and it also means that the AI will need to be retrained with every change to the game that is made. In addition, training times can prove extremely long and this can prove to be somewhat impractical.

The previously mentioned rule based AI is a system where the AI takes an action if a condition is true. For example, in an RTS game a rule could be that the AI only attacks after it has reached enough units. Ligêza, 2006 rule based systems are one of the most common ways of solving problems as they explicitly state what is and what is not possible, this makes is simple to define what actions the AI will carry out. The actions an AI can carry out are defined by the rule base and the AI cannot deviate from them. Due to this, machine learning is not possible and the AI cannot adapt to a changing situation. However, rule based AIs are comparatively simple to set up if the rules are known in advanced and this is one of the main reasons why they have been used in the games industry to such a large degree. Another use for rule based AI is that they can be a useful benchmark for repetitive testing of other AIs such as neural networks or genetic algorithms which require a great deal of training before they prove successful.

Fuzzy logic as described by P Wang, D Ruan and E Kerre, is a form of AI which, as opposed to actions or choices being correct or incorrect, relies on the idea that not all the facts are known and there is an element of uncertainty to all decisions being made. It also allows for a degree of ambiguity in the information provided to the AI and the main idea behind it is to turn an analogue world into a system which a digital machine can understand. This can work well for competitive games as by their nature a player does not know what the other player is thinking, therefore they can only prepare for the most likely action. Each action is given a truth value between zero and one, the most appropriate action is given the highest weighting. The truth value represents the percentage chance of an action being taken but does not guarantee that any specific action will be taken as an action will rarely if ever have a truth value of one. However, a flaw with this method is that you either need to know in advance what each truth value should be or have some method of machine learning for the AI to adjust its own truth value.

Genetic algorithms were initially described by Holland, 1975 and are based upon evolutionary theory. They start with a completely random set of choices from the AI that are evaluated by a fitness function with the most successful being crossbred. The fitness function is what defines the success parameters, which is application dependent and the ’fitter’ the solution is, the higher the chance of being chosen. The chosen solutions are then used as parents for a new generation. Each generation is mutated by randomising some of the variables to ensure there is some variation between generations. This means that they cannot end up in a situation where an optimal solution could be found but the algorithm cannot find it as it is trapped in a local minima.

Genetic Algorithms have been studied in a variety of domains from data mining and bioinformatics (Maulik 2011) to finding the best methods of energy efficiency (Liu and Huang 2012). Directly related to games it has been used to make an AI that “is capable of evolving a team's behaviours and optimizing the commands in a shooter game” (Liang 2013). This paper created an AI for Quake III Arena that was designed to be competitive with humans. It discusses the original rule based AI in Quake and how it limits the quality of the AI. Later it goes into detail about their use of a finite state machine to simplify the complex systems in a first-person shooter to ensure the AI can understand. It operates through assigning scores to different states such as winning a game, capturing the flag and killing an enemy. These scores are fed into the fitness function and used to calculate the most successful solution of that generation. This allows the AI to learn how to select the most beneficial actions by passing through states as quickly as possible until it wins.

A different approach to the problem was taken by Mora et al (2012) and Fernández-ares et al (2013) who worked together to create a bot for the game Planet Wars which was part of the Google AI challenge. It is a turn based multiplayer game where each player controls a planet and have a certain number of troops. Also on the map are a large number of uncontrolled planets. The object of the game is to take over all your opponent’s planets by sending troops to defeat defending enemy troops, neutral planets can be taken over in the same way. The more planets you have the more troops you have, meaning you can launch more invasions and eventually win. The first paper (Mora et al 2012) uses the genetic algorithm to tune the weightings of a rule based AI to provide an initial base. Then, the AI is trained against other AIs that follow different methods of learning in order to provide more precise and accurate results.

The second paper (Fernández-ares et al 2013) starts with what was produced from the previous year’s work but takes a different route in improving the quality of the AI. Here they created multiple methods of determining success by using three different fitness functions that operate differently and are “based upon victories and numerical performance”. They mapped the output of the different versions of the genetic algorithm using behaviour trees in order to understand and map the choices that the AI made. Their results showed that the best method for creating the most effective genetic algorithm was through making the fitness function victory orientated when compared to any other parameter such as planets controlled or resources gathered.

## 2.3 Summary

There are many different AI techniques that can be used in games but the crucial factor is having an AI that can learn and adapt to a changing situation while providing a more enjoyable experience for the player. The most commonly used technique is the rules based system so a comparison should be made with a rule based AI. A genetic algorithm was chosen as it allows for an AI that unlike a rule based or fuzzy logic system is capable of learning. This is necessary for the game to learn how to play and adapt to a certain scenario to overcome its opponent. A genetic algorithm also has the advantage compared to a neural network in that it does not merely copy the example provided. Instead, the fitness function is the definition of success rather than whatever the neural network example is. This means the fitness function is the most important factor in the genetic algorithm as there is the danger that a genetic algorithm could become superior to the player that is being used to train it.

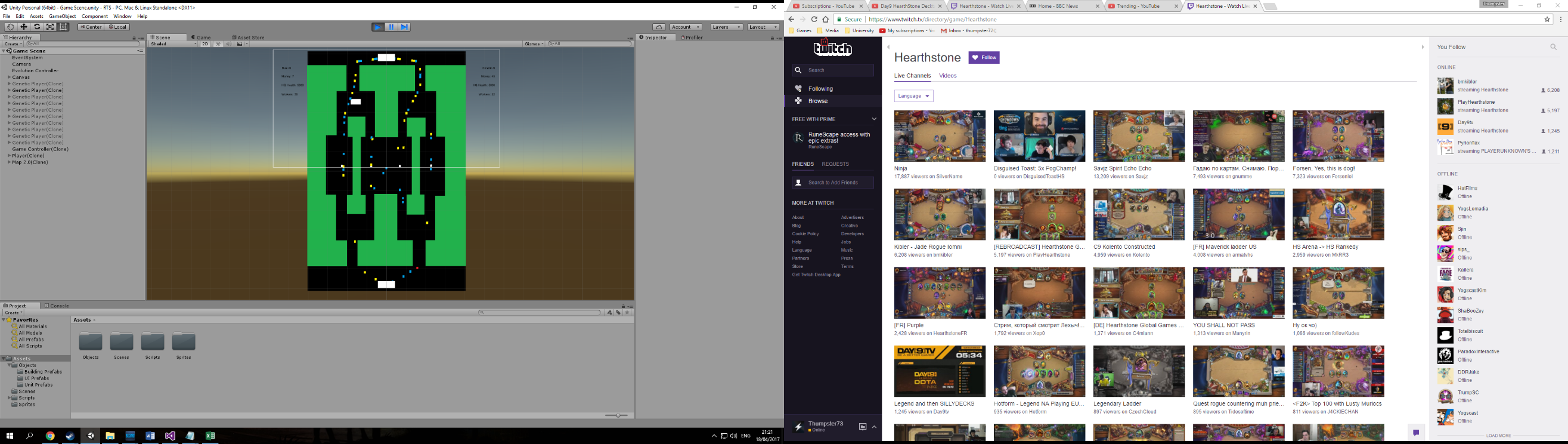
# 3. Methodology

Based on the research carried out, the chosen methodology is an AI that uses a genetic algorithm to adapt and learn how to win against a rule based AI. The rule based AI will have a fixed ruleset and cannot change what actions it carries out between games but the genetic algorithm AI can. The genetic algorithm AI has a list of initially randomised actions and both AIs use a finite state machine to act upon the action chosen. Then after several games have been played against the rule based AI, the fitness function will evaluate which versions of the genetic algorithm AI are most successful and breed them together. This creates a new list of actions that the AI will carry out during gameplay. The fitness function will take victory over the rule based AI as the single most important factor in choosing how successful the AI was in a game though other factors are considered.

This project has three sections, a genetic AI, a rule based AI, and an RTS game created for the purposes of testing the AIs. The RTS segment is loosely based upon (Age of Empires II 1999), taking the core gameplay such as building units, gathering resources and attacking the enemy in order to provide sufficient complexity to test the AIs. The gameplay is just a canvas upon which the AIs can be tested and is not the focus of the project, instead gameplay is used to provide meaningful comparisons of the AIs.

## 3.1 Gameplay

The gameplay has three main elements, player actions, unit movement and unit interactions. Each player can carry out the same types of actions, however the method in which they are carried out is different. Here detail will be provided on the types of actions and the requirements needed for those actions to be carried out. One of the main requirements for most actions is for a player having enough resources to carry out that action. More resources can be obtained by increasing the number of worker units a player has or by controlling more resource points on the map (Figure 2). There are many other unit types other than worker units however the worker unit is the only unit that can be trained at the start of the game. Other units require a resource investment by constructing buildings before they can be trained. These buildings include the barracks which trains melee units that only attack at close range, the archery range which trains ranged units and the stables which train fast hard hitting cavalry. Different units and building cost different numbers of resources and generally the most expensive a unit is, the more powerful it is in combat. There is also a tiered structure where a barracks is required before an archery range, and an archery range before a stable. A further step in the chain is that for the most powerful units to be built, a player must invest resources in research to unlock the ability to train that unit. The research itself is only available after a stable has been built. Other building opportunities for the player include walls and towers. These can be placed at specific chokepoints on the map that protect the player’s headquarters and are impassable to enemy units. The requirements of building walls and towers are that a player must control the area around the building site and have enough resource to construct the building. A player can capture base expansion points which allow the construction of an additional resource gathering building. These resource gathering buildings allow for a higher worker count as the Headquarters building only allows for twenty workers to gather resources at once. Each additional expansion point once built increases the cap by twenty units meaning that a player can gather more resources allowing for more units to be trained. A full list of units, buildings and technologies can be found in appendices one to three.

The unit interactions take place between neutral objects and enemy objects. Neutral objects include the terrain of the map and building points that can be captured. The tower bases can be captured and become friendly by having the most units near it. A friendly tower base can be turned into a tower by expending resources causing it to deal damage to nearby enemy units and must be destroyed to prevent this. The terrain of the map is a physical blockage that the units cannot travel through and acts as the boundaries of the map. Interaction with enemy objects involves attacking nearby enemies until either unit dies. These interactions follow the game rules as opposed to direct control by the player, the AI trains the unit then the actual attacking is resolved automatically based upon the proximity of units. A player can create walls that are passable to friendly units but unpassable to enemy units. These walls can be damaged and rebuilt similarly to towers.

Expansion

Headquarters

Figure 2 – Game Map

The final part is the movement of units across from one side of the map to the other. This uses a system of waypoints where a unit is given a list of directions by the player and an initial vector location to head towards. This is the only control the player has over player movement. For example, a unit could be told to go left and is given the vector location of the first waypoint. The unit would then travel towards the waypoint and upon reaching it the unit is given the location of the next waypoint. This continues until either the unit has died or it has reached the final waypoint of the opposing player’s headquarters. On the map (figure 2), there are paths which can be branching, such as at the beginning where a unit can be sent either left or right. The green blocks are the walls of the map, the coloured squares are units and white squares are buildings. Changing the direction counts as a game action and is global for all units, however if a unit goes left initially it does not mean it will always go left. Upon reaching a branching waypoint the waypoint queries the unit’s owner on the current global direction of all units and sends the unit the corresponding direction.

## 3.2 Rule Based AI

Initially a basic rule based AI was created that can play the game at a beginner level. It was given a list of orders to use in a situation, for example early on it focuses on building a strong economy then steadily builds up units to attack while advancing through research. The rule based system has no adaptability and is not capable of learning. The goal of the game is to destroy the opposing player’s headquarters building and the game will end when either player loses their headquarters. A new game will begin with the same rule based AI but the genetic algorithm will have a different set of actions to follow.

The main limiting factor on what actions can be carried out is the number of resources that a player has. Therefore, to resolve this, a fixed priority list was created that causes actions to be carried out in a specific order dependant on how high they are in priority. Programmatically this was implemented as a series of IF ELSE statements in C# with only one action being selected per update, with the code travelling down a series of checks until one is true. A basic overview of the rule based AI priority list is that it trains worker units until it has trained 20 units, then it builds a barracks to allow training of warrior units, then begins to train warriors while saving up enough money to build an archery range. The AI works its way down the priority list until an action is possible, if no actions are possible then no action will be taken. If an action further up the priority list becomes possible while resources are too low for the next action, the higher priority action will be carried out instead on the next update. One difference between the rule based AI and the genetic AI is that the rule based AI changes the global unit direction whenever a unit is trained and is not a discrete action. When an action has been selected, it utilises the same finite state machine as the genetic algorithm AI does. The full priority list can be seen in appendix four.

## 3.3 Genetic Algorithm AI

The two AIs choose the action they are going to carry out in different ways. At the beginning of the initial training, a list of actions is created for the genetic algorithm for each of the original genetic AIs. The list corresponds to the idea that genetic algorithms represent DNA and each action is a chromosome. Each action in the list has a position and a value, the position refers to the order in which the actions will be carried out and the value refers to the type of action that the AI will attempt to process. This list is randomised using the Unity Random.Range function which generates an integer when given two other integers. Each integer represents an action, for example, if one was the result then the corresponding action would be to build a worker. If an action proves invalid for any reason, the AI will move onto the next action without trying to repeatedly attempt the failed action. The full outline for types of actions can be seen in appendix 5. An initial list of five thousand actions is created and if this list runs out then more actions are generated randomly. After ten games have been played, using different randomly generated lists the fitness function will evaluate the effectiveness of the list of actions to select individuals for the mating pool.

The fitness function has winning the game as the most important factor when evaluating candidates (Fernández-ares et al 2013). In an RTS having excess resources but not spending or having no way to spend them, is inefficient as the resources could be used to create more military units to aid in the AI’s victory. Therefore, having more money than the most expensive unit results in a lower fitness factor. Other factors include how long the game took; with the faster wins accounting for more, dealing damage to the enemy headquarters, surviving for a longer game and how much damage per second was dealt on average through the game. These factors were chosen to try to ensure that the longer lasting versions of the AI would be most likely to be chosen, but if the AI wins it will be chosen regardless of other factors. The fitness function works by giving each candidate a score based upon the above factors then the two highest scoring candidates are added to the mating pool to be bred together during the crossover stage to create the child candidates for the next phase of games. The crossover occurs by taking alternating actions from the parent candidates to create a new child candidate that has properties of both the parents. An example of this process for the range 124 to 132 can be seen in figure 3.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Candidate | …Action 124 | Action 125 | Action 126 | Action 127 | Action 128 | Action 129 | Action 130 | Action 131 | Action 132… |
| Parent 1 | 12 | 9 | 4 | 2 | 1 | 5 | 16 | 12 | 15 |
| Parent 2 | 5 | 2 | 9 | 11 | 15 | 17 | 18 | 14 | 7 |
| New candidate | 12 | 2 | 4 | 11 | 1 | 17 | 16 | 14 | 15 |

Figure 3 - Genetic AI Crossover stage

A small percentage of the new candidate’s actions is mutated to prevent stagnation as otherwise this could lead to a situation where the path the genetic algorithm chooses leads to a dead end. This is known as a local optimum. The mutation is carried out by re-randomizing 2% of the actions in the list using the method that was used to create the initial population. This is carried out ten times to create a new population. The new population then attempts to defeat the rule based AI and the process repeats as shown in figure 4.

Initialize the Population

For each candidate play a game

Evaluate candidates using fitness function

Perform crossover

Generate new population

Perform mutation on population

Figure 4 – Genetic algorithm process

Once the genetic AI can beat the rule based AI consistently, more complexity was added to the game with more unit types with varying combat abilities and different buildings to train them from. In addition to this, it has been made more difficult for a player to build the more powerful units by adding research requirements that mean the player must make an additional investment of resources. Carrying out research occurs from a building and means that building cannot train units during that time. This means that the player will be at a temporary military disadvantage during gameplay the research time, therefore the genetic algorithm must only carry out research when it will not lose if it does. Similarly, it is necessary for the genetic algorithm to find the most optimal timing to build an expansion. If the expansion is built too quickly, they will be rapidly overrun by the opposing player’s units and lose the game. If it is built too slowly, then the opposing player will have more resource and therefore more units than the genetic algorithm AI. This timing is a critical area for which the genetic algorithm must find the best expansion timing or they will ultimately lose.

To further increase complexity and reduce brute force luck, the map was changed to provide multiple avenues of attack and defence, meaning there is more that the AI must get correct to win. Other changes were made such as walls and towers which respectively provide a temporary blockage for enemies or attack enemy units. More complexity means more choices for the AI therefore the initially random choices made by the genetic algorithm are less likely to be correct initially and it must learn how to play through evolution by being evaluated using the fitness function.

# 4. Results

This section will discuss the results produced from the application created as described in the methodology. Data will be shown that represents the success or failure of the genetic algorithm in learning how to defeat the opposing player as well as the effectiveness of the fitness function. To begin with, one hundred iterations of the genetic algorithm were carried out, where each iteration involves the genetic AI playing the rule based AI for ten games. After each ten games the genetic algorithm evaluates using the fitness function which versions of the AI proved most successful by assigning a fitness factor as described in the methodology. The two most successful candidates are bred and mutated to create a new generation and this process as shown in figure 4 is called an iteration. The rule based AI carried out the precise same actions every game without changing as it is designed to do and there are no results to show for that. Figure 5 shows the fitness factor over time for the successfully bred candidates, there are two hundred candidates for one hundred generations.

Figure 5 – Fitness factor over time

As can be seen, early on the fitness factor is negative to represent that the genetic AI is failing to meet any of the fitness function’s success criteria that were described in the methodology. As time goes by the most successful versions of the AI are identified and the fitness factor begins to rise over time. The large spikes on the graph represent the genetic AI winning a game as that is the most important factor and the main means of defining success. The smaller spikes represent the algorithm finding other factors which from observation of the gameplay, were found to be the genetic AI dealing damage to the opposing player’s headquarters. Towards the end, the algorithm has identified the actions which align with the fitness function’s success criteria and it is consistently winning games against the rule based AI.

Figure 6 shows how the genetic algorithm was steadily learning and moving towards a more successful outcome early during the testing. It started with a large negative number to reflect that no damage was being dealt to the opposing player’s headquarters then proceeded to steady trend towards a higher fitness factor. This reflects that despite neither winning nor dealing damage towards the opposing player’s headquarters, it was still managing to meet some of the success criteria. Similar to figure 5, it shows the fitness factor for both parent candidates therefore there are two fitness factors per ten games played.

Figure 6 – Early Fitness Factor

Figure 7 shows the win rate of all games played per iteration of the genetic algorithm. One hundred iterations were carried out with ten games being played for each iteration. Figures 5 and 6 only show the fitness factor for the successfully chosen candidates with the highest fitness factor but figure seven shows the win rate for all ten games. A 0% win rate would mean that no games were won by the genetic algorithm AI while a win rate of 100% means that the genetic algorithm won all ten games in that iteration. As can be seen, early on the genetic algorithm does not win any games therefore the win rate is 0%. Towards the end of the hundred iterations carried out during testing, the genetic algorithm AI proved more successful with a steadily higher chance of winning throughout the last 150 games played. The genetic algorithm reached a peak of a 90% win rate towards the end, meaning it won nine out of the ten games that it played during that iteration.

Figure 7 – Win Rate

During the course of gameplay, several observations were taken on the actions made by the genetic algorithm AI. Early iterations would tend to result in the genetic algorithm AI losing quickly against the opposing rule based AI without surviving for long or providing a meaningful challenge against the rule based AI. As time passed the genetic algorithm AI would last longer and deal more damage over the course of a game. One particularly notable observation is that the genetic AI would predict the direction of the rule AI’s units by sending its own units to attack in that direction. For example, the rule based AI attacks from the left side at the very start of the game. To counter that, the genetic algorithm AI would send its own units towards the left pathways of the map (figure 2) preventing attacks from that direction meaning it survived for longer. A further side effect of this was that because the genetic algorithm AI was sending units down the left pathway, sending even more units along that path would lead to the AI damaging and eventually destroying the opposing player’s headquarters. This was an example of a chain of actions that showed how surviving in the short term leads to ultimate victory for the genetic algorithm AI.

As the genetic algorithm AI initially had a completely random set of actions to begin with, it could clearly be observed that it was training more resource gathering units than was optimal. The gameplay works that for every expansion building or headquarters a player controls, they can have twenty workers effectively gathering more resources to be spend on units, giving a cap of twenty workers at the start. However, the genetic algorithm AI was training far too many workers wasting resources. As the testing progressed, the wasted resources decreased as the number of workers increased more slowly when above the cap. Notably in the games where the genetic algorithm built an expansion meaning a higher cap of forty or sixty, it would build more workers.

Other observations made include that the genetic algorithm AI was more successful when it built more archer units from the archery range and less infantry from the barracks. It tended not to build an expansion in the vast majority of games, though there was a noticeable increase in the chance it would build one expansion in a game it was winning. It never built more than one expansion.

# 5. Discussion

* Evaluate your findings
* Comment on their significance in relation to previous work on the same topic
* Refer back to your literature review where appropriate
* Use the key performance indicators outlined in your proposal if appropriate to aid your evaluation, referring back to initial project requirements
* Guide – 2000 - 2500 words
* Discuss expected results
* How they may different from actual results and why that may have occurred
* Discuss how different rule based AI produce different results.
* To what extent the genetic AI changes over time
* How fast it changes
* How mutation level choice affects
* How quickly the genetic algorithm can adapt to the rule based AI

This section will discuss the results that were expected and explain how they may differ from those produced by the project and why. It will also discuss how the work is relevant to previous pieces of work on the same topic. Success of the project will be measured by the genetic algorithm being able to identify the successful gameplay choices made by the AI and then be able to replicate them in future games without much change occurring. Ideally the genetic AI would utilise all the features of the game, training a variety of different units and buildings. The AI should focus on the economic aspect by building expansions and training workers, as well as the military aspect by training combat units and constructing military buildings. However, as the fitness function’s largest factor is whether the genetic AI has won, and using all the game features is not a factor at all, it was expected that this will not happen. Instead, it was expected that the genetic algorithm would favour fast victories that rely very little on defence.

Against a human player the optimal solution may not be ideal as an unbeatable AI is not fun to play against, therefore it may be necessary to add a deliberate level of error into the AI by selecting a suboptimal solution. This is done by outputting the list of actions to a file, which can be read from and used to recreate that list of actions at a later date. Commonly in RTS games there is the option for different difficulty settings, this could be done by using different solutions to represent easy, medium and hard. This will hopefully create a more realistic computer controlled player that can make mistakes as a human player can, but also capable of winning.

# 6. Conclusion and Future Work

* What conclusions can you draw from your investigation?
* What are the implications of what you have discovered?
* How might further work in this area be continued?
* Guide -750 - 1000 words
* Are genetic algorithms a viable option? Yes, is the expected answer but depends on results.
* How could this be used in an actual game.
* Problems with this method
* How effective is it in a real situation

# 7. Appendices

## 7.1 Units

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Unit Name** | **Health** | **Speed** | **Damage** | **Special** | **Requirement** | **Building** |
| Worker | Low | Medium | Low | Can gather resources | None | HQ |
| Warrior | Medium | Low | Low | None | None | Barracks |
| Spearman | Medium | Low | Medium | Good versus stable units | Barracks Spears | Barracks |
| Swordsman | High | Low | Medium | Expensive | Barracks  Swords | Barracks |
| Archer | Low | Medium | Low | Can attack targets at range | None | Archery Range |
| Crossbowman | Low | Medium | High | Can attack targets at range  Expensive | Archery Crossbows | Archery Range |
| Horseman | Medium | High | Medium | Fast | None | Stable |
| Knight | High | High | High | Best Unit | Stables Feudalism | Stable |
| Catapult | Low | Low | Low | Can attack targets at range  Deals bonus damage to buildings | Archery Range Ballistics | Archery Range |

## 7.2 Buildings

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Builds** | **Special** | **Requirement** |
| Headquarters | Workers  Catapult | If this is destroyed, you lose  Building Upgrades | None |
| Barracks | Warriors  Swordsman  Spearman | Infantry Upgrades | None |
| Archery Range | Archer  Crossbowman  Catapult | Archery Upgrades | Barracks |
| Stables | Horseman  Knight | Cavalry Upgrades | Archery Range |
| Expansion | Nothing | Can gather more resources here with workers | None |
| Walls | Nothing | Blocks Enemy Units  Can be upgrades with towers | none |
| Tower | Nothing | Shoots Enemy units | None |

## 

## 7.3 Technology

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Building** | **Requirement** | **Unlocks** |
| Advanced Research | Headquarters | Stables | Towers, Swords, Ballistics and Feudalism |
| Spears | Barracks | None | Spearman |
| Swords | Barracks | Spears  Advanced Research | Swordsman |
| Crossbows | Archery Range | None | Crossbowman |
| Ballistics | Archery Range | Crossbows  Advanced Research | Catapults |
| Feudalism | Stables | Advanced Research | Knights |

## 7.4 Rule Based AI Priority List

## 7.5 List of Actions

|  |  |
| --- | --- |
| **Action** | **Value** |
| None | 0 |
| Train Worker | 1 |
| Build Barracks | 2 |
| Train Warrior | 3 |
| Train Archer | 4 |
| Build Archery Range | 5 |
| Train Knight | 6 |
| Build Stables | 7 |
| Build Tower | 8 |
| Send units to Left waypoints | 9 |
| Send units to Right waypoints | 10 |
| Research Advanced Research | 11 |
| Train Swordsman | 12 |
| Train Spearman | 13 |
| Train Crossbowman | 14 |
| Train Horseman | 15 |
| Train Catapult | 16 |
| Train Wall | 17 |
| Build Expansion | 18 |
| Research Swords | 19 |
| Research Spears | 20 |
| Research Catapults | 21 |
| Research Knights | 22 |
| Research Crossbowman | 23 |

# 8. References

Simon Mardle and Sean Pascoe. 1999. An overview of genetic algorithms for the solution of optimisation problems. *Computers in Higher Education Economics Review.* 13(1). Available from: <https://www.economicsnetwork.ac.uk/cheer/ch13_1/ch13_1p16.htm> [Accessed October 2016]

Chishyan Liaw, Wei-Hua Wang, Ching-Tsorng, Chao-Hui Ko, Gorden Hao. 2013. [online]. Evolving a team in a first-person shooter game by using a genetic algorithm. *Applied Artificial Intelligence.* 27(3). Pp.199-212. Available from:

<http://www.tandfonline.com.libproxy.abertay.ac.uk/doi/abs/10.1080/08839514.2013.768883>

Antonio Mora, Antonio Fernández-ares, Juan J Merelo, Pablo García-sánchez, Calros M Fernandes. 2012. Effect of Noisy Fitness in Real-Time Strategy Games Player Behaviour Optimisation Using Evolutionary Algorithms. *Journal of Computer Science and Technology.* 27(5). Pp.1007-1023. Available from: <http://search.proquest.com.libproxy.abertay.ac.uk/docview/1197182214?OpenUrlRefId=info:xri/sid:primo&accountid=8159>

Antonio Fernández-ares, Antonio Mora, Juan J Merelo, Pablo García-sánchez, Calros M Fernandes. 2013. [online]. Analysing the influence of the fitness function on genetically programmed bots for a real-time strategy game. *Entertainment Computing*. Volume 18. Pp.15-29. Available from: <http://www.sciencedirect.com.libproxy.abertay.ac.uk/science/article/pii/S1875952116300222>

The Guardian. 2016. [online]. 'Partnership on AI' formed by Google, Facebook, Amazon, IBM and Microsoft. Available from: <https://www.theguardian.com/technology/2016/sep/28/google-facebook-amazon-ibm-microsoft-partnership-on-ai-tech-firms> [Accessed October 2016]

BBC. 2016. [online]. Google’s AI wins final Go challenge. Available from: <http://www.bbc.co.uk/news/technology-35810133> [Accessed March 2016]

John H. Holland. 1975. Adaptation in natural and artificial systems. USA. University of Michigan Press.

[Ujjwal Maulik](http://link.springer.com.libproxy.abertay.ac.uk/search?facet-creator=%22Ujjwal+Maulik%22), [Sanghamitra Bandyopadhyay](http://link.springer.com.libproxy.abertay.ac.uk/search?facet-creator=%22Sanghamitra+Bandyopadhyay%22), [Anirban Mukhopadhyay](http://link.springer.com.libproxy.abertay.ac.uk/search?facet-creator=%22Anirban+Mukhopadhyay%22). 2011. Multiobjective Genetic Algorithms for Clustering. Berlin. Springer-Verlag Berlin Heidelberg.

Cheng-Hsiang Liu, Ding-Hsiang Huang. 2012. Reduction of power consumption and carbon footprints by applying multi-objective optimisation via genetic algorithms. *International Journal of Production Research.*

Blizzard Entertainment. 2002. Warcraft III: Reign of Chaos. [disk]. PC. Blizzard Entertainment.

Blizzard Entertainment. 1997. Starcraft. [disk]. PC. Blizzard Entertainment.

Blizzard Entertainment. 2010. Starcraft II. [disk]. PC. Blizzard Entertainment.

Ensemble Studios. 1999. Age of Empires II: Age of Kings. [disk]. PC. Microsoft Corporation.

King, David. 2016. *Genetic Algorithms.* [online image]. Dundee. Abertay University. Available from: <https://blackboard.abertay.ac.uk/webapps/blackboard/execute/content/file?cmd=view&content_id=_388273_1&course_id=_6156_1> Accessed October 2016.

Kenneth Aizawa*.* 2013.Warren McCulloch’s Turn to Cybernetics: What Walter Pitts Contributed *Interdisciplinary Science Reviews.* 27 (3). Pp. 206-217.

Warren McCulloch, Walter Pitts. 1943. A Logical Calculus of the Ideas Immanent in Nervous Activity. *The Journal of Symbolic Logic.* Volume 5. Pp. 115-133.

P Wang, D Ruan and E Kerre. 2007. Fuzzy Logic: A Spectrum of Theoretical & Practical Issues. *Studies in Fuzziness and Soft Computing.* Volume 215. Pp 1-13.

Antoni Ligêza. 2006. Logical Foundations for Rule-Based Systems. USA. Springer.

# 9. Bibliography

Firaxis Games. 2010. Sid Meier’s Civilization V. [disk]. PC. 2K Games

IBM. Frequently Asked Questions: Deep Blue. [Online]. Available from: <https://www.research.ibm.com/deepblue/meet/html/d.3.3a.html> [Accessed October 2016]

Amazon. 2017. [online]. Amazon Machine Learning. Available from: <https://aws.amazon.com/machine-learning/> [Accessed February 2017]