Benjamin Joshua Charlton

ID Number: 4262648
Supervisor: Professor Robert John
Module Code: G53IDS

2017/18

Computer Science with Artificial Intelligence MSci

G53IDS - Individual Dissertation

Applying Evolutionary Algorithms to Pokémon Team Building

Submitted April 2018, in partial fulfilment of the conditions for the award of the degree Computer Science with Artificial Intelligence MSci

Author
Benjamin Charlton
psybc3@nottingham.ac.uk
4262648

Supervisor
Prof. Bob John
Robert.John@nottingham.ac.uk

School of Computer Science
University of Nottingham

I hereby declare that this dissertation is all
my own work, except as indicated in the text:

Signature:_____

24th April 2018

Abstract

- Short overview of the entire project

The aim of this dissertation is to show AI search methods can be used to help pre-game decision making tasks such as team and deck building. As AI game playing moves towards more complicated games the idea of how to tackle these pre-game choices will come up whether in choosing the deck list in a collectable card game or item load outs in a turn based strategy game.

The focus of this project is the team building aspect of the Pokémon games and how it can be tackled using evolutionary algorithms.

Acknowledgements

I would like to express my thanks to all that have helped me develop the necessary skills required to undertake a project of this scale, and have kept me motivated through out the process.

The School of Computer Science at the University of Nottingham has given me much support in my academic studies building strong skills and good practices needed to develop quality code. Alongside the academic staff I would like to give thanks to the student body, particularly Hack-Soc for motivating and pushing me beyond the curriculum to discover, learn and practice new skills.

I would like to thank my supervisor Professor Robert John for allowing me to complete this project with his unwavering support and confidence in me. At times I wasn't sure whether I was going on the right path but his reassurance and kind words provided much motivation. Thank you for agree to supervise this bizarre and novel idea of mine so I could see this project to fruition.

A final thanks goes to my friends and family who have shared my excitement as I explain my project to them, been my rubber duck to explain my problems to when struggling often helping solve them without realising, and sincere thanks to those who put up busy schedule to allow me to continue with the project.

Special thank you to my younger siblings whom out of everybody understood the least about what I was trying to do but despite all of that showed love and support for me even though it missing out on spending time with me.

TODO List

- Finish each section with a summary and tell the reader what comes next
- Find and read and reference the book
- Clean and commit code ready for submission
- Folder
- Video

Contents

1 Introduction

1.1 Introduction

Video games are an ever growing field of interest for many people. Like many games people play competitively against each other and in some cases their are tournaments on an international scale with whole teams set up to win the large prize pools[7][6]. This field has become to be known as eSports.

Game playing is an obvious application of AI techniques as you can objectively score wins and losses. Initially this has been seen with traditional board games like Chess[5] and Go[10]. During The International 2017 for DotA 2 this all changed as the worlds of AI and eSports came together in a 1v1 show match[13]. Elon Musk, backer of this initiative, said that this is 'Vastly more complex than traditional board games like chess & Go'[13]. Typically in these set ups their is no decisions to be made prior to the game itself, in fact to limit the DotA 2 AI they limited it to a preselected character to play with.

Many strategy games have a decision making element before the game is played. Collectable Card Games (CCGs) have the choice of which cards to put in your limited deck or in Pokémon you have to choose which members of your team to use and how you have trained them. This element of the games is refereed to as deck/team building as the player has to build up what they bring to the game from an empty deck or team. Some players are very good at playing the game but not very good at choosing which deck/team to bring resulting in a practice called 'netdecking', being called so as the player will take someone else's deck/team from the internet instead of coming up with their own.

Pokémon has a tricky team building process that is rather hard for new players to comprehend. In the build up to the annual Pokémon World Championships, many players will go through the tedious process of team building to make sure they have the right strategies and counters in place to bring to the matches ahead[9]. This requires expert knowledge such as type match ups and speed tiers along side several rules of thumb. Once a general strategy is in place the players must perform several calculations to optimise the statistics of their team, optimising offensive stats to faint certain threats or optimising defensive stats to allow the team to survive after certain moves are used against them.

1.2 Motivation

The motivation for this project comes partly from the novelty of the idea. As evolutionary algorithms are designed to simulate survival of the fittest could it work on a representation of a biological population. In the Pokémon games there is an idea that their is a living ecosystem with predators and prey as well as a system for the Pokémon to fight in combat to decided who is the strongest.

Further motivation comes from the seeing how AI game playing is progressing and trying to create something that will combat other aspects that come into playing video games. This approach could be useful for players of the game allowing them to find new and winning strategies that might not have previously been known. It could also help the development team behind the games balance the game despite the ever increasing mass of combinations that they have to consider.

1.3 Aims and Objectives

This project aims to implement a evolutionary algorithm that will create a Pokémon team. The output of such will be a team of Pokémon, including all of the vital statistics and moves; this would be dependant of the format. These results will be then compared to human designed solutions to conclude if the evolutionary algorithm is comparable to an expert.

The objectives of this project are:

1. Research evolutionary algorithm methods used to approach similar problems. Looking in detail about issues such as representation, evaluation and validity of the chromosomes. This

information will be used to help direct how best to approach the problem and design the system.

2. Design an effective and efficient way to model and represent the problem in the evolutionary algorithm while still maintaining relevant data to the problem. This will also be used to form the output so will need to be readily be able to translate into a readable format for the user to understand.

3. Develop a Genetic and Memetic algorithm to tackle the problem from scratch and the required methods for the various stages in the algorithms. Making sure that all of the elements of the code base are created in a fashion that allows for reusability and adaptation. For example both the Genetic and Memetic algorithm could share the methods like objective evaluation and selection. This will all be developed from scratch (bar the use of a few relevant APIs).

4. Compare and analyse the results of each evolutionary algorithm (with a variety of settings) with each other and human designed solutions. Solutions for comparison will come from readily available teams from top players and analysis will be taken by evaluating the solutions. If solutions are viable enough they can be input into the game and used in some real world settings such as battling with other players, rather than being graded by a score.

1.4 What is Pokémon

- Find references to back stuff up
- Is wikipedia ok?

1.4.1 Pokémon Fanchise History

Pokémon was first created by video game designer Satoshi Tajiri with the first games in the series coming out in February 27, 1996 in Japan for the Nintendo Game Boy. The name Pokémon comes from a romanised contraction of the words 'Pocket' and 'Monsters' (ポケットモンスタ - **Poke**tto **Mon**sutā) due to the portable nature of the Game Boy, being able to take the games with you in your pocket. The initial concept of the game stems from creators Satoshi childhood hobby of insect collecting. When they first saw the Game Boy, in particular the link cable technology to wire 2 systems together for multiplayer, they imagined insects being transferred between two players by them crawling along the cable. This formed to create the idea of there being several Pokémon to catch and trade between players.

The first games in the series were developed by Game Freak and published by Nintendo, initially only releasing to the Japanese audience. Pokémon: Red and Green came out as two separate games that differed slightly in the available Pokémon to encourage players to socialise and interact to complete the games. After the hit success selling over 10 million units across the 2 games it was clear that Pokémon was going to be a huge success and began to franchise out, including rebuilding the games from the ground up to sell to the worldwide audience. The worldwide release saw combined sales of over 9 million units showing to the world that Pokémon was going to be a smash hit worldwide for years to come.

22 years later the Pokémon franchise has spanned into a cultural hit being recognisable by many people throughout the world. Alongside the 31 main series games, there are almost 100 spin off games, over 1000 episodes of the anime, 21 movies, a trading card game, and several toys being sold around the world in their Pokémon Center stores. The franchise is now managed by The Pokémon Company, which is owned equally by Nintendo, Game Freak and Creatures Inc.

1.4.2 Pokémon Core Gameplay

The core series games are role playing games following the adventures of a up and coming Pokémon trainer. The games start by the local Pokémon Professor offering the main character their very first Pokémon to begin their journey, in exchange they ask that they take a Pokédex to fill out on the journey. The Pokédex is a encyclopaedia for all Pokémon but only fills out once the player has seen and caught the Pokémon in question.

The second goal for the player is to tackle the Pokémon league and become the league champion. To achieve this title the player has to travel from town to town besting the local gym leaders in a Pokémon battle. The player will face many other battles through out the games as well facing non player characters such as hikers, park rangers and even evil gangs. The players can also challenge other players to Pokémon battles to test their might against each other.

At the highest levels players face off in the Pokémon Video Game Championships or VGC for short. VGC is a yearly season where players can go to tournaments to earn points by battling other distinguished trainers. If players earn enough points they are invited to the World Championships to try and find the best player in the world.

1.4.3 Pokémon Battles

Pokémon battles are a main point in the series and preparing for these battles is the focus of the project.

Pokémon battles are turn based fights between 2 trainers, each with a team of up to six Pokémon. To win a battle you must faint all of your opponents Pokémon by reducing their Hit Points (HP) to zero. This is done by selecting a move to attack your opponent with each turn, or players can also use items or switch to another Pokémon. The amount of damage you deal is based upon a few factors including the Pokémon's base stats, power of the move, type effectiveness and even stat boosts. The strategy comes from using your available Pokémon in the best way possible to beat your opponent.

When the trainer enters the battle they can't alter their team in anyway for this reason you need a team that works well together and can successfully counter several strategies. A good team has several strong Pokémon on it rather than relying on a select few. The team members are also their to cover the weaknesses of other team members. A strong trainer knows what makes up a good team and can select strong Pokémon that together also make a strong team.

1.5 What are Evolutionary Algorithms

Evolutionary Algorithms are a set of artificial intelligence methods used for optimisation problems. As the name suggests EAs are based upon the biological process of evolution.

1.5.1 Natural Evolution

Biological Evolution as percieved today was first layed out in Charles Darwin's Book 'On the Origin of Species' in 1859. In biological evolution individuals are encoded by genetic material referred to as genes. The population is made up of all of the individuals that are alive, and a single generation can be described as all of the individuals that were born around the same time. What has been seen in nature is that over several generations the population as a whole adapts to become better suited to the environment. Due to the shorter life span and therefore quicker turnover of generations, it is easier to see this effect in bacteria. Since the introduction of antibiotics it has been seen that some bacteria have evolved to become resistant to these antibiotics evolving to overcome a new threat in the environment.

This process of evolution happens due to several mechanics at play. Selection is the idea that the population inherently selects the stronger individuals in the population to pass on the genetic material to the next generation. The main way this happens is that weaker individuals will die before they have the chance to reproduce removing their genetic material from the gene pool, or stronger individuals will survive for longer creating more offspring. Crossover occurs during reproduction, often involving 2 individuals sharing their genes to create a child which has a mix of both of the two parents genes. Presuming both parents were strong individuals the child may inherit the parts of each parent that made them strong creating a stronger child. The final element of natural evolution is mutation, due to random processes it is possible that part of a individuals genome may mutate. This mutation could end up being beneficial for the individual making them stronger and eventually reproducing adding the mutation to the general gene pool.

Genetic evolution is only part of what happens in nature, in 1976 in the book 'The Selfish Gene' Richard Dawkins proposed the idea of memetics. A meme would be analogous to a gene but represent the passing on of information between individuals. An example of a meme could be

the idea of a tool and how to use it. This information is reproduced as it is passed upon between individuals, selection comes in the form of whether individuals think that it is relevant to pass onwards and sometimes ideas mutate as individuals adapt them.

1.5.2 Genetic Algorithms

Genetic algorithms take the idea of biological evolution and apply it to optimisation problems. This idea first pioneered by John Henry Holland in 1975 when he noted the similarities between natural and artificial systems to adapt to solve problems.

In this analogy a solution to the problem is an individual and the genetic material encodes this solution in some way. The population is a set of individuals with each generation being more rigidly defined in which all of the processes happen in one go to a set number of children. Before crossover between parents can happen the process of natural selection is simulated by selecting the parents in some form, often based upon their fitness. The fitness of an individual is determined by an objective function rather than the ability of the individual to survive in the environment. The children are created by applying a crossover method between the selected parents to simulate the passing on of the parents genetic material, afterwards a random mutation may occur to the child. Unlike in the real world solutions don't have a life span. To simulate this change over and removal of older individuals a population replacement method occurs.

1.5.3 Memetic Algorithms

Memetic Algorithms are a subtle adaptation of genetic algorithms. By adding in the idea of some information sharing between the generations as within the theory of memetics. This manifests itself as a local search step within the evolution process. This local search step can be seen as taking a child and training them to become stronger.

2 Related Work

- Investigate more works to talk about here (gaming and more general)
- GAs, MAs and other EAs

2.1 AI and Game Playing

Many AI approaches to games tackle the aspect of playing the matches and the decision making process to choose the best action. Lots of research and development has happened in these areas with many effective techniques being discovered. One key reason the problem of prematch decision making hasn't been tackled is due to the lack of need for it with classical table top games requiring no preparation before the match begins.

Chess was an early and significant example in the history of AI, with the Deep Blue computer from IBM successfully beating the at the time world champion Garry Kasparov[5]. This was significant as creating a winning chess AI was seen to be the next big milestone at the time in AI. To achieve this Deep Blue used a combination of techniques with the main underlying AI technique being a search method. To achieve this at the time custom hardware was created to help speed up the computation with a highly parallel structure that could evaluate nodes on the search tree quickly using hardware implementations. This meant that it could effectively search deeper than any other AI at the time.

Go is the most recent milestone game to be beaten with AlphaGo claiming its victory against one of the best Go players, Lee Sedol, in March 2016[10]. To achieve this the team uses Deep Neural Networks combined with Monte Carlo tree search, with a combination of supervised learning from human games and reinforcement learning via self-play.

Recently AlphaGo has been beaten by a variation of itself AlphaGoZero, named as it had learnt from zero human knowledge[11]. AlphaGoZero learnt entirely from self-play and achieved super human performance. This shows that not only can AI techniques successfully solve tasks but it is

possible to do so with no expert knowledge.

As the field of AI game playing moves forward into more complex games prematch decisions will need to be considered. With current methods it would be rather simple to build and train an AI to play turn based strategy games, such as collectable card games or in this case Pokémon, but the deck/team building would require an expert to decide what the AI will be trained to use. This is often problematic as season rotation could add in new elements to the game or make certain elements no longer useable, or shifts in the metagame will mean that the AI is easily countered.

Team building is a form of optimisation problem as you are trying to bring the optimal team to the match so you have the best chance of winning.

A variety of work has been conducted looking at optimisation via AI techniques, in this review of previous works a focus has been upon techniques that tackled having a large, vast search space and where the correctness of a solution was hard to judge. Both of these issues are problems that will have to be over come in building this project.

2.2 Hearthstone Deck Building GA

García-Sánchez et al. tackled a very similar problem using a genetic algorithm to approach deck building[8]. The example they used was a popular collectable card game, Hearthstone, and they tried to create a viable competitive deck through the genetic algorithm. This is of particular interest as several parts of their study directly relate to what the project is trying to achieve, as well as several short comings that will have to take into account.

An evolutionary algorithm was used as 'they commonly produce very effective combinations of elements'[8], which a deck can be described as a combination of cards. This also allowed for competitive decks to be built from scratch with no expert knowledge required. This was done by encoding the individuals as a vector of cards, in this case the 30 element vector became the deck.

In the deck building process it was possible to have a deck that would violate the rules of the game. To discourage the EA from creating decks that violate these rules; a correctness metric was taking into account when calculating the fitness of each individual. If an individual was incorrect it wasn't evaluated further in the fitness calculation process and given the worse possible score[8].

A large part of the work talks about the fitness evaluation, as with many of these processes it is hard to quantify how well a individual will do. Even though human experts will have a strong intrinsic idea to what is better or worse, they will often playtest their ideas for several matches, sometimes ranging into the hundreds, before applying some analysis to develop their idea further. To give the most realistic simulation a separate AI played each individual against some previously expert designed decks, that have been proven to be successful in the metagame that the AI was evolving in. It played several matches with each individual against the decks, with 16 matches versus the 8 chosen opponent decks totalling 128 matches per individual[8].

One of the key upsides of this was that the GA could evolve to combat common decks that it would face if it was to be played in competitive play. Knowledge of key opposing threats is something that high level experts take into account, this knowledge was taken from an article evaluating the current meta game written by professional players of the game. Another upside was from having a high number of matches being played for the evaluation; by playing a large proportion of games they can mitigate any randomness in the matches that aren't down to the deck building, such as the cards in the starting hand. To further mitigate randomness and to give a more accurate representation of the individuals fitness, statistical analysis was used to calculate the final fitness score. This was done to also as it is important to find 'a deck with a fair chance to win against all decks in the metagame' not one that is strong against a particular opponent and weak to all others.

Through this method of fitness evaluation some flaws and potential shortcomings where brought up. As the fitness evaluation required a large amount of matches to be played it resulted in 'every execution of the algorithm requires several days'[8], even when running on a small population size and number of generations (10 and 50 respectively). Although not a specifically a bad thing a longer run time would discourage the use of such tools, more so in games where the meta game changes more rapidly. The evaluation also required the expert knowledge of the meta decks to

be tested against or even that an established meta is in effect where as many experts can theory craft a rough idea before changes to the meta happen. The most significant flaw with this style of evaluation is that it relies on the ability of another AI to pilot the deck. It was suggested that it may overfit 'with regards to the AI capabilities and playing style' even going as far as pointing out that it 'can use some decks (and some playing styles) better than others'[8].

2.3 Evolutionary Algorithms

Evolutionary Algorithms have been shown to be a strong way to find near optimal solutions to complex problems. Zitzler and Thiele spoke about their merits in their case study suggesting that 'Many real-world problems involve simultaneous optimization'[15]. They pointed out that the often some factors would conflict and the EA were strong at finding a balance between them. It was concluded that 'All multiobjective EA's clearly outperformed a pure random search'[15].

Maumita Bhattacharya spoke about differing approaches to the problem of computationally expensive fitness evaluation[4]. Several of the techniques described approximated the fitness of each individual while only periodically calculating a more accurate value. Some of these techniques achieved this my simulating a simpler model while more specific techniques where discussed too. In a population search method like evolutionary algorithms, a technique called Fitness Inheritance could be used. As the children individuals will closely resemble the parents you could give a rough approximation to how different they actually are and use that difference to evaluate the children. This will greatly reduce the number of computationally expensive evaluations you need to do with potentially as few as only scoring the initial population. However it is suggested that you periodically recalculate using the true fitness evaluation as 'Using true fitness evaluation along with approximation is thus extremely important to achieve reliable performance'.

Fitness Inheritance was first proposed by Smith et al.[12] where they applied the idea to a simple problem OneMax. They investigated two simple methods to inherit fitness. Averaged Inheritance simply said 'a child is assigned the average fitness of its parents', this is extremely cheap computationally and requires no detailed analysis. Another method was Proportional Inheritance where 'the average is weighted based on the amount of genetic material taken from each parent', while slightly more expensive it takes into account of none symmetric crossover schemes such as one point crossover where the genetic material is not equal from each parent.

3 Description of the Work

The objective of this project is to develop an EA that evolves a population of Pokémon teams. This is to show that such a system could be used to aid human players in quickly evaluating the large amount of possibilities or to assist another form of AI to eventually play these games at an expert level.

The project should function by the user setting the parameters and then setting it running via a command prompt. This could be potentially wrapped in some form of user interface in the future but won't be developed during this project to allow for focus on the EA.

3.1 Functional Requirements

The project doesn't have many functional requirements due

- Parameters of the EA should be able to be edited at minimum by editing a separate file. These settings should include:
  - Whether its running as a MA or GA
  - The number of generations
  - Population size
  - Mutation rate
  - Number of local search steps to perform

- The EA should display to the user progress throughout the process via some simple text output in a command line.
- The EA should present the highest rated individual once finished in a human readable format. This format will be in 'Showdown Format' allowing the user to import the solution in to the popular simulator Pokémon showdown.
- The EA should be able to import a benchmark function to prove that it is working correctly.

4 Methodology

4.1 Evolutionary Algorithms

The main AI technique behind this project is evolutionary algorithms, this technique has been chosen as it successful optimises problems with little expert knowledge as shown in the related works. There are a few different types of evolutionary algorithms, with genetic algorithms and memetic algorithms being focused on in this project. The reasoning for choosing these two types is that they are closely related and will allow for a lot of the code to be shared between them, in fact a memetic algorithm can be seen as an extension of a genetic algorithm. Another potential extension would be a multimeme memetic algorithm but the scope of the project does not allow time to develop this.

The key advantage of evolutionary algorithms is that they can effectively explore the search landscape due to their population based search method, a critical feature in this problem as there are a a lot of possible combinations to search through. For example in a team of Pokémon each team member on a team of 6 can be chosen from the 805 different species of Pokémon. This gives the following number of possibilities:

$$\binom{species}{teamSize} = \binom{807}{6} = 3.765 \times 10^{14}\ (4sf) \tag{1}$$

Once you begin to factor the the other potential values it becomes a lot larger very quickly, a rough estimate for the number of possibilities is as follows:

$$\binom{species \times IVs^{stats} \times EVs^{stats} \times levels \times happiness \times abilities \times genders \times items \times \binom{moves}{moveslots}}{teamSize} \tag{2}$$

$$\binom{807 \times 31^6 \times 256^6 \times 100 \times 256 \times 3 \times 2 \times 354 \times \binom{728}{4}}{6} = 5.893 \times 10^{261}\ (4sf) \tag{3}$$

Although this is an astronomically large search space, a large proportion of these possibilities will be invalid solutions as if any of the team members aren't valid then the whole solution becomes invalid. By using a evolutionary algorithm this search space can be quickly explored and iterated upon.

Another advantages of evolutionary algorithms is that they allow for the usage of both domain specific methods and more general methods, potential to be parallelised, ability to translate the solution to a human understandable result and the simplicity of the overall concept.

After the idea that a GA could be used to tackle the solution, extending it to a MA isn't much harder. This was one of the primary reasons to develop the project in such a way that it could be used as a MA. Another reason for adapting the project to an MA was that great success can be found with certain local search methods. Some interesting domain specific search operators could be defined to allow for experts to shape the development of the population with heuristics used without hampering the wide exploration that the GA produces across solution space.

The issues with these techniques come from how to accurately represent the problem in a form that can be easily manipulated by the algorithm. This is rather complex on real world problems and requires careful thought and design to mitigate any issues later down the line. Another issue comes from need to score each solution, in this problem there is no real objective way to score a

solution. Practically an EA also has the issue of run time, like most AI techniques they are very computationally expensive and having a longer run time will result in more optimal answers.

Evolutionary algorithms were chosen over other AI techniques not only due to the advantages above but as there are issues with other techniques when approaching this problem.

As shown with the calculations above, brute force methods are unfeasible as the search space is far too large. Even if you effectively limited the search space by employing several powerful domain specific heuristics or pruning of the search space it would not limit it enough to allow for brute force methodologies. Brute force methods still suffer from the lack of an objective score for each solution to be able to return the best result.

Many techniques such as Decision Trees and Support Vector Machines concentrate on classifying the result based upon an already labeled dataset that they learned form. To adapt the problem into a classification style problem would be impractical as mentioned earlier it would be hard even for human experts to say what is good and bad to classify a solution. Artificial Neural Networks also approach classification problems effectively and thus have similar issues to other classification problems Although human made solutions exist there is not a mass of solutions to learn from so these machine learning techniques aren't easily applicable.

Mathematical optimisation techniques can't be effectively used to calculate an optimum as the objective value that you are trying to optimise is extremely difficult to calculate. Other AI optimisation techniques could be used to solve this problem, such as Local Search methods. The key draw back of the local search methods is defining a move operators for this problem, as it is such a high dimensionality problem and not all of the dimensions have obvious move operators.

4.2 Object Orientated Design and Python

The project will be coded in Python, a powerful Object Orientated programming language. Python was chosen due to its strong object orientated design features that will be rather suitable for coding of evolutionary algorithms. By following several object orientated design practices the project will be easily adaptable through out development, primarily allowing for the easy adaptation to the MA further in the project by reusing multiple classes.

Python was also chosen as it is a very commonly used language, being the 3rd most popular language on GitHub[14] as of the writing of this report. This popularity means that there is extensive resources on the language and it goes to show how powerful it is. Alongside its popularity comes many usable APIs, in particular this projects takes advantage of Pokéapi, which has a python wrapper called pokebase. The use of this specific API and wrapper were critical to the project as it allowed easy access to the large amount of data the project needs to function.

4.3 Software Engineering Tools

To aid with the large scope of this project, a variety of software engineering tools are being used. This section will list some of the larger tools and processes that they provide.

Git Git is a version control system the main draw of such a technology is to allow for easy restoring of code to a working state if a change breaks something. This is done by having a tree of 'commits' within each are the changes to the files at that point, normally stored as line differences in the code files; at any point you can go back in the tree to a commit that needs to be reverted too. The tree structure can also branch to allow different modifications to be made, this feature is currently being used to allow for documentation and code to be written at the same time. It will useful later on with the adaptation from the GA to the MA, allowing a stable branch with working GA code in it and a MA branch that is modifying the same files.

To use the Git system there are a few pieces of software being used. Although you can work on a local repository, it is highly recommended that a remote repository is used as a form of backup and allows for the code base to be worked upon from multiple devices so long as the remote repository is up to date. The project will be stored on the School of Computer Science GitLab server, leveraging the functionality of the remote repository while keeping the project private for the foreseeable future.

LaTeX LaTeX is a software to help with documentation preparation, which has been used to create all of the documents thus far. A key reason to use this is that it stores the source files in a manner that allows for Git to effectively show the changes. This allows for the entirety of the project to leverage the Git repository rather just the code base.

Referencing is a lot easier in LaTeX as during the compile it will automatically work out what the correct references are with easily human readable tags for each citation. Another feature of LaTeX is the use of images, by storing them in a separate folder and loading them in any changes that I make to the image will be replicated in the document. Although it has taken some time to get used to the nuances of LaTeX it is allowing for much more powerful and granular control of the documentation writing, while relieving stresses such as how parts are brought together.

Atom Both the code and the documentation are being written in a text editor called Atom. The main use of Atom in this project is that it edit both the code and documentation simply by installing packages to recognise both languages. Additional packages have also been installed to allow for easy compilation and viewing of the LaTeX files and a project management system which will save the workspace for the various segments of the project. It also integrates well with Git showing which files and specific lines have been modified, allowing for easy management at a glance.

Together all of these tools help with the software engineering process by both encouraging good practice and streamlining some more complex processes. In the long term this will help with the project management of the project by helping keeping it organised and clear.

4.4 Testing

The overall EA structure has been tested on a benchmark function to prove that it can be run successfully. This also shows that EA is adaptable and able to run on other problems.

The benchmark function is used is the Sum of Squares, in this the objective value is the sum of the squares of all the elements (2 in this example) where each element can be in the range -5 to 5, as real values. The objective function can be calculated using the following equation:

$$where\ n = 2 \quad \sum_{i=1}^{n} x_i^2 = x_1^2 + x_2^2 \tag{4}$$

This objective value is to be minimised, giving a global maximum of every element being 0. In this case the following hyper parameters were used:

$$generations = 50,\ populationSize = 10,\ mutationRate = 0.1,\ LocalSearchSteps = 5 \tag{5}$$

Running it 5 times gave the following results to 3 significant figures:

x_1	x_2	Objective Value
0	0	0
9.74e-5	0	9.49e-9
0	0	0
3.44e-4	1.17e-3	1.50e-6
7.45e-4	0	5.55e-7

These results show that the overall outline of the EA does work and on simple well defined problems it can successfully find an acceptable result.

5 Design and Implementation

5.1 Evolutionary Algorithm

The main part of this project is the EA, this is the section that binds all of the other parts together into a useable form. The design of the algorithm is rather simple and has an out line as follows:

5.1.1 EA Algorithm

```
Result: Returns the best individual in the population
generate an initial population of individuals;
for Number of Generations do
    evaluate the current population;
    for Population Size / 2 do
        Select 2 parents from the population;
        Create child via Crossover between selected parents;
        Apply Mutation to the child;
        Validate the child;
        if Memetic Algorithm Mode then
            for Number of Local Search Steps do
                Apply Local Search on child;
            end
        end
        Evaluate the child;
    end
    Apply population replacement;
end
Find best rated individual in population;
```

Algorithm 1: Evolutionary Algorithm Structure

5.1.2 EA Implementation

This algorithm is implemented in evolutionaryAlgorithm.py alongside some print statements to allow the user to see progress. The implementation mainly is relatively short as it abstracts the main processes to the problem classes. The population is stored in an array with each element of the array being an individual, this is created during the initialisation stages. The fitness is stored in an equally large array and the corresponding index of the population array refers to the fitness of that individual in the fitness array.

To facilitate the requirement of being adaptable to a benchmark function, an instance of the problem is set during the initialisation. The problems share the key methods including initialising an individual. As python is dynamically typed language these initialisation methods can return a new individual to be dynamically added to the population array. The representation of the individuals is also handled in a different class.

Variables that are used in the EA (population size, number of local search steps) are stored in the file constants.py to facilitate the requirement of being able to edit settings. Due to the functionality of python these values aren't truly constants but they are never manipulated without changing the file.

5.1.3 EA problems

A major problem was hit during this section of the project that was not identified until much later on in the process. The bug came from the final section where it found the best rated individual in the population. As the system was initially tested on the sum of squares benchmark problem the algorithm to find the bast rated individual was the lowest scoring one. The opposite was true for the Pokémon problem as a higher score was better, but the algorithm wasn't changed initially meaning the worst rated individual ended up being printed to the user. This was a simple fix by having each algorithm have a comparison function used to compare the fitness of two individuals.

5.2 Representation

In any evolutionary algorithm the representation of the individuals is a key consideration in both design and implementation.

5.2.1 Representation Design

The overall design of the representation is high level for several reasons. The first is that it allows for a simpler design through out other parts of the EA, validating individuals would be easier as you can directly access the variables you need. Another reason to keep the representation high level is to aid in the displaying of results to the user.

To come up with the design for the representation, relevant data to store was chosen by looking into what data is need to create the teams in game. The primary resource used was Pokémon Showdown, 'a battle simulator for Pokémon battles'[1]. This is a community created tool to allow players of the Pokémon games to play online against other players. Another important feature of the simulator is its team builder, a tool in which a player can create a team with aids such as filtering of choices and simple sliders. Once created a player can share their team using the export feature which gives a short readable plain-text description of the team, this is both easily readable by human players and can be imported into Showdown to allow other players to use the team. Although this can't be imported into the Pokémon games, players can quickly test the teams in the simulator before going through the longer process of building the teams in game. The format of the import/export text of Pokémon Showdown is a popular way of sharing teams in the Pokémon community so it was chosen as the target output for the representation. This would also allow for the solutions to be brought into Pokémon Showdown for testing.

A Pokémon team is comprised of 1-6 Pokémon and each Pokémon has the same characteristics because of this a team can be represented as up to 6 Pokémon. This is reflected in the Showdown format as a team is simply 6 Pokémon displayed one after each other separated by empty lines. As can be seen in the figure below the information about a single Pokémon is quite a lot. This example is a single Pokémon from an expert designed team.

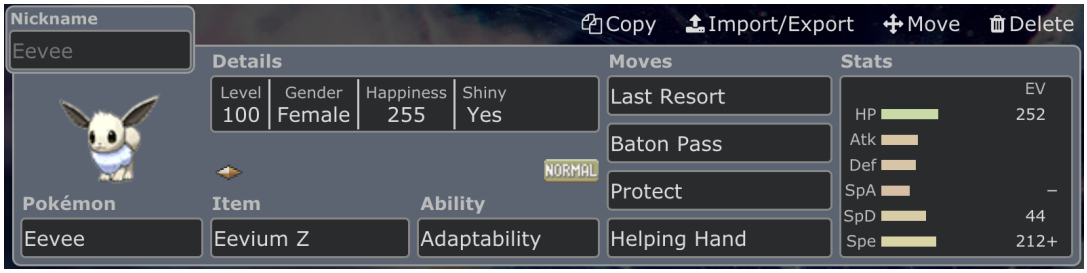


Figure 1: Pokémon Showdown team builder graphical UI

The same Pokémon shown in the figure has the following import text:

```
Eevee (F) @ Eevium Z
Ability: Adaptability
Level: 100
Shiny: Yes
Happiness: 255
EVs: 252 HP / 44 SpD / 212 Spe
Jolly Nature
IVs: 31 HP / 31 Atk / 31 Def / 31 SpA / 31 SpD / 31 Spe
- Last Resort
- Baton Pass
- Protect
- Helping Hand
```

To encode this information each characteristic will be stored in a separate gene as a Pokémon and a team will comprise of six Pokémon. This allows for the easy extraction of the individuals into a printable format as shown above. The list of these characteristics is as follows:

- Pokémon Species
- Gender
- Held Item
- Ability
- Level
- Shiny
- Happiness
- Nature
- EVs (x6 for HP, Atk, Def, SpA, SpD, Spe)
- IVs (x6 for HP, Atk, Def, SpA, SpD, Spe)
- Moves (4 Seperate moves)

5.2.2 Representation Implementation

The implementation relied heavily upon the ease of integration with the Pokéapi system. Most of the implementation is therefore created in such a way to allow for calling the values within Pokéapi to retrieve further information. Altough Pokéapi allows for things to be referenced by name, it was chosen to use Pokéapi's ID system making initialisation and limits much easier to evaluate.

Not all possible combinations of this representation are valid for a variety of reasons. Many of the IDs aren't associated with anything such as there are no values for Pokémon Species 803-10000. The EVs have individual values for the separate statistics of the Pokémon but in the Pokémon games the total sum of all of the EVs can't surpass 510. The largest constraint on the combinations is whether it is 'legal' in game, for example not all Pokémon can learn all of the moves that are listed.

In the Pokémon games it is possible some things to be left blank. This happens in 3 cases, Number of Pokémon on a team, held items, and Number of moves. A Pokémon team can be made up of 1-6 members, so the representation allows for this by having 6 slots that can store Pokémon objects. Any Pokémon can have a held item although this isn't required. A Pokémon also has 1-4 moves so the representation has 4 move slots that each store an individual move id. Any of these can be listed as None which refers to their being no object stored there.

Characteristic	Stored As	Valid Range	Associated Pokéapi Call	Can be Null
Pokémon Species	Integer ID	1-10147*	pb.pokemon(ID)	No
Gender	Integer ID	1-3	pb.gender(ID)	No
Item	Integer ID	1-918*	pb.item(ID)	Yes
Ability Slot	Integer ID	1-3	Shown in code appendix	No
Level	Integer Value	1-100	N/A	No
Shiny	Boolean Value	True, False	N/A	No
Happiness	Integer Value	0-255	N/A	No
Nature	Integer ID	1-25	pb.nature(ID)	No
EVs	Integer Value	0-255	N/A	No
IVs	Integer Value	0-31	N/A	No
Moves	Integer ID	1-718	pb.move(ID)	Yes

* Not all IDs in this range are associated with something

5.2.3 Representation Translation

Despite having a fairly high level representation their is some need to translate the encoded representation into a useable format. The first example of this is when printing the end result to the user, as a series of ID values aren't particularly helpful. In particular the functional requirements state that the end result should be printed out in 'Showdown Format' to the user. Not only does this mean that IDs have to be converted into string names, they also have to be in a certain format. By using the Pokéapi system translating these values into the required strings was easy enough as you can simply request the name associated with the ID.

During validation and evaluation there is a need to check certain combinations of values. For example during validation, a Pokémon may have a valid species ID and gender ID but certain Pokémon maybe limited to certain genders which has to be checked. Some values also have associated meaning that is required for some of the calculations in the evaluation. Due to the way that the data is stored the ID values can be used with Pokéapi as a database lookup. This allows for the minimal amount of data to be stored in the representation.

5.2.4 Representation Problems

Most of the representation is built to use ID values for a the connected Pokéapi database. For both initialisation and validation a range of ID values needed to be known to initialise to and validate against. Within Pokéapi there is no default way to access these ranges so it took time to find out what they were.

For some of the smaller ranges it was possible to find it out through trial and error, often knowing the size of the range and only correcting for off by one errors. This method was not viable for larger ranges where the size wasn't known and there was potential that some ID values weren't going to be used. This also an issue of maintainability as whenever Pokéapi updates these ranges would have to be recalculated. The end solution was to look on Pokéapi's GitHub repository and read the backend data directly to find out these values[3]. This isn't a favourable solution as it still relies on looking up the new values when Pokéapi is updated.

The representation design brought up issues during implementation due to the some things being able to be represented by None objects. At several stages the project would have to check whether a value was None to avoid crashing when referencing it. Occasionally this would result in a major bug that would crash the program as these checks weren't performed. All of these bugs were found in the end but it was a common occurrence and only arose due to the design of the representation.

5.3 EA methods

For the EA to run there are several methods that need to be implemented. Within each part of this section a method used by the EA will be explained at a high level explaining its function and how it approaches it.

5.3.1 Initialising Individuals

Initialising individuals in the population is done via random selection of each gene in the individual. This is done in a manner to make sure that the individuals generated are valid. Values that can be stored as None have a random rate (that can be found in the constants file) to appear as None.

Due to the symmetric nature of have 6 Pokémon the method uses a helper method that generates an individual Pokémon. For each Pokémon that is generated the following structure is followed.

```
Result: Returns a randomly initialised individual
for Each member of the team (6) do
    if random value ≤ No Team Member % Chance then
        Set Pokémon slot to None;
    else
        if random value ≤ No Item % Chance then
            Set itemID to None;
        else
            Randomly set itemID from range;
        end
        Randomly set level from range;
        Randomly set shiny from range;
        Randomly set happiness from range;
        Randomly set nature from range;
        for Each IV (6) do
            Randomly set IV from range;
        end
        for Max EV Total do
            Randomly +1 too a EV;
        end

        Randomly select a species from range;
        Get a list of all the varying forms of the species;
        Randomly choose a form from the list and set the formID;

        Get a list of all the abilities the Pokémon can have;
        Randomly choose an ability slot from the list and set the ability;

        Get a list of all the genders the Pokémon can have;
        Randomly choose an gender from the list and set the gender;

        Get a list of all the moves the Pokémon can have;
        Randomly choose 4 moves from the list;
        Remove duplicate moves;
        Set the moves chosen;
    end
end
```

Note This is rearranged from the actual implementation for clarity of reading

Algorithm 2: Initialisation of Individuals

5.3.2 Fitness Evaluation

Creation of a objective fitness evaluation function is difficult with this problem. To evaluate the population a score was given to each individual compared to the rest of the population. This can more easily be achieved by approximating how well the two teams being compared at any time would perform against each other in a Pokémon Battle.

For each member of the population the individual is compared against, a score is given based upon how many members of the team win against members of the other team. This compares each team member to every member of another team. The final score is an average of all these scores against every team in the population.

A win between 2 Pokémon is determined by comparing the potential damage done by each of the Pokémon's moves. The Pokémon that deals the highest damage is said to be the winner. Either a team member is set to None then they automatically lose. The calculations to find this

information was taken from the Pokémon games and are shown below[2].

$$Damage = \left(\frac{\left(\frac{2 \times Level}{5} + 2 \right) \times Power \times A/D}{50} + 2 \right) \times Modifier \tag{6}$$

$$where \text{ A is the relevant attack stat of the attacking Pokémon} \tag{7}$$

$$\text{D is the relevant defense stat of the defending Pokémon} \tag{8}$$

There are 2 modifiers calculated which are to do with the Pokémon's types. If the attacking Pokémon shares a type with the move it is using it gets a Same Type Attack Bonus (STAB) of $\times 1.5$. Type Effectiveness of the move type on the defending Pokémon is the other modifier. If the defending Pokémon is weak to the move, a $\times 2$ modifier is applied. If it resists the move then the modifier is $\times \frac{1}{2}$, and if the Pokémon is immune the modifier is $\times 0$. For Pokémon with two types, type effectiveness is calculated for both types and is multiplied together.

5.3.3 Parent Selection

Parent Selection happens by randomly selecting any individual in the population.

5.3.4 Crossover

Crossover only happens at the level of the Pokémon in a team as crossing over other characteristics per Pokémon like move set would almost certainly make the individual invalid. This happens by generating a list of all the Pokémon from 2 parents and then randomly selecting 6 of them. This may generate duplicates but that is taken care of in validation.

5.3.5 Mutation

The mutation happens randomly at a % chance that is controlled in the settings. If it is randomly chosen that a child should be mutated one of the Pokémon that make up the team is randomly chosen to have a mutation apply.

There is currently 25 different possible mutation that can occur with a variety of outcomes.

Characteristic to Mutate	Mutation
Pokémon	Reinitialise the Pokémon
Gender	Reinitialise
Ability	Reinitialise
Level	Reinitialise
Level	Increase by a random amount
Level	Decrease by a random amount
Shiny	Reinitialise
Happiness	Reinitialise
Happiness	Increase by a random amount
Happiness	Decrease by a random amount
Nature	Reinitialise
Nature	Select nature with same +ive boost
Nature	Select nature with same -ive boost
EVs	Reinitialise
EVs	Swap a pair of EVs
EVs	Decrease and Increase a pair by the same amount
IVs	Reinitialise
IVs	Swap a pair of IVs
IVs	Increase one IV by a random amount
IVs	Decrease one IV by a random amount
Item	Reinitialise
Item	Set to None
Moves	Reinitialise all moves
Moves	Reinitialise one move
Moves	Set a single move to None

At any time only one of these mutations happen and once mutated the child is returned.

5.3.6 Validation

Validation attempts to check if an individual is this is critical as it is possible for crossover/mutation to create an invalid individual. This part of the algorithm heavily relies upon expert knowledge to know what can be invalid and how to check for it. Some of the checks are done to determine whether the team would be able to be created in the games, were as other checks are to abide by common rulesets for competitive battling.

The first check is to make sure none of the Pokémon on the team are the same species, as this is banned in many rulesets. If duplication is found the Pokémon team member is set to None.

After the team structure as a whole has been validated, each member of the team has to also be validated. The resulting checks happen to all of the team members and if anyone of the team is invalid that member is set to None.

The gender of the Pokémon is checked against its gender ratio through Pokéapi, as some Pokémon can only be a certain gender. The held item is checked to see if it is both in the database and if it is a holdable item. The ability slot is checked to see if a Pokémon has a ability in that slot. Levels, happiness, IVs, and natureID of the Pokémon are all checked by seeing if they are in a valid range. The EVs are checked by seeing if the total is within a valid range and whether each EV is within the valid range. Moves are checked to see if they can be learnt by the Pokémon, Moves are also checked to see if there are no moves listed and if there are any duplicates.

5.3.7 Population Replacement

Population Replacement happens by randomly selecting individual from the population and comparing it to the child. If the child has a better fitness value than the random individual the child will replace the individual. This happens for each child created each generation.

5.3.8 Local Search Method

The local search method applies to all of the Pokémon IVs, changing them by ±3. The two new individuals are then compared to each other using the fitness function. It then returns the better of the new individuals.

5.3.9 EA Methods Problems

Many of the methods required for the EA were simple to implement and could be directly copied from the proven benchmark function. Unfortunately not all methods were as easy to write. Fitness evaluation was a particular issue as not only was it difficult to come up with a successful way of approximating the problem research into the calculations used in game needed to be found and translated into usable code.

Testing all of these methods was also an issue as there are many code paths to follow and some may not be obvious until after several iterations of the program. This could be more tackled with unit tests but the schedule didn't have time for these to be written unfortunately.

Many methods have well defined solutions to them but required to have a certain representation of the individuals, such as binary representation. By not being able to use these methods some difficulties were encountered as novel solutions had to be created and tested.

6 Evaluation

- Explain what is being evaluated
- How the EA was tested
- Statistical evaluation of performance
- Evaluation of software performance

6.1 Software Performance

The performance of the software to run efficiently is hard to empirically state, although this serves to be some comments on the matter. The evaluation part of the program takes a while to run due to the way it ranks the population. Firstly the frequent database lookups are rather slow and this can be seen as the program runs as each win/lose takes a while to print to the screen. On slower systems will low disk speeds such as a raspberry pi this can easily be confused with the system being unresponsive. Secondly the exponential nature of comparing each individual in the population means there is a lot of time spent in the evaluation process on the whole.

Even on the relatively small runs used for evaluation purposes the run time was around 48 hours. AI applications don't need to be snappy and responsive like user facing applications but for the quality of results found this implementation took a considerable amount of time.

6.2 EA Performance

In this section the performance of the EA to generate a solution is being evaluated. The solutions will be analysed by an expert to see how the solutions appear in theory, the better solutions will be compared against random results and real world scenarios of increasing difficulty to try and properly assess them. There were 2 sets of runs were performed to get answers one running in GA mode another in MA mode, with a total of 11 solutions being generated.

6.2.1 Expert Remarks

The solutions generated by the EAs are on the whole subpar, but evidence of some strong ideas can be seen in the results. Only one of the generated solutions did not contain a full team of six Pokémon, there is no penalty to having more team members as it is more bodies that the foe has to beat and it gives you a wider variety of counters to your opponents tactics.

On the whole most of the teams contained a few Pokémon that were much stronger compared to the others. This shows that the way the objective function works evaluates the solutions allows for teams to be carried by one or two stronger Pokémon. As these Pokémon are in the gene pool though it can be suspected that if left to run for several more generations a team of 6 strong Pokémon would be produced.

Optimally a Pokémon should be a the max level, in this case 100. Of all of the 65 Pokémon only 4 were max level, although it was clear that the trend was towards higher level Pokémon. This is probably that high levels is one of the biggest impact on the strength of a Pokémon meaning that individuals with high levels will have performed well in the evaluation.

Another optimal setting that is followed is having all of the Pokémon's IVs to be 31. This is done as the higher the IVs the higher the Pokémon's stats will be influencing the strength of the Pokémon. There is often no reason to set a Pokémon's IVs to not be max, in those cases it is because you want a certain stat to be as low as possible therefore having an IV of 0. Although in general the solutions had high IVs none had maximised all of the IVs.

Another thing that can be seen in the solutions is the high base stat totals of the Pokémon species chosen. Base stat total (BST) is a rule of thumb metric used by experts to show how strong a Pokémon is. A high BST indicates that the Pokémon is strong and will generally perform well as it can deal more damage and take less. This can be seen as 6 of the Pokémon are classified by the community as top tier and are often banned from competitive matches.

The EVs of all of the Pokémon are roughly evenly distributed, which is highly influence with how they are generated. This is a far from optimal solution to go with as it doesn't play to the strengths and weaknesses of a Pokémon. EVs are rather complicated with every 4 EVs in a stat gives 1 stat point at level 100. There are also a limited number of EV points you can distribute in total and each stat has a limit of the EV points it can have. The common approach by players is to split the points 252/252/4 to maximise 2 of the stats required typically an offensive stat and a defensive/speed stat. This points out a flaw in the solutions which is investing this limited resource into stats that it doesn't benefit from. Pokémon are reffered to physical or special attackers depending on what stat their move set relies upon, with some rare cases of mixed attackers relying on both. Their are some solutions which aren't mixed attackers but still invest EVs into the other stat that they don't benefit from. This appears to be heavily biased by the way EVs are distributed in the initialisation and this emergent property might not appear for thousands of generations.

Happiness often doesn't make a difference to the performance of Pokémon but in select cases it does. Those specific cases are if a Pokémon knows the moves return or frustration, as their power scales with the Pokémon's happiness. They both have the same maximum power so are functionally identical when at the optimal happiness. In some solutions these moves can be found but they don't optimise the happiness in accordance.

When looking into the move sets (the set of moves that a Pokémon has) a few conclusions can be drawn. The first is that the majority of the moves are known as STAB moves for the Pokémon using them. A Same Type Attack Bonus applies to Pokémon who uses a move that has the same elemental type, such as a Fire Pokémon using Ember a fire type move. Most of the Pokémon had a STAB move but this could be heavily influence by the fact that most Pokémon's learn sets (the list of available moves they can learn) are primarily STAB moves. Typically the moves had a high base power, meaning that they do more damage than other moves. The hallmark of a strong moveset is a concept called coverage, as some moves are weak to different types of Pokémon. In some of the solutions it can be seen that the Pokémon have some coverage moves enabling them to be stronger against a wider range of Pokémon. Many other Pokémon can be seen with only one type of move though showing that it may just be random chance.

The final part of the solutions have been grouped together due to them all being seemingly random. Of all of the remaining attributes shininess of Pokémon serves no purpose other than an aesthetic preference. Gender, in extreme situations, can make a difference but this doesn't commonly happen. As such having a random solution for these 2 parts isn't uncommon. Natures are a big part of optimal solutions as they provide a significant change in the stats, $\pm 10\%$. This part of the solutions might be something that isn't optimised until much later in the generations as the other things have a much more drastic effect upon the Pokémon end stats. Items can play a large role in the battles but a limited amount of them are useful, with the majority not having an effect when held by a Pokémon and most of the items in solutions don't have an effect. As the

search space of abilities is so small per Pokémon it is hard to determine whether the solutions have evolved those outcomes or not. Although it is clear that not all of them are optimal and some even have no use in battle.

6.3 Project Limitations

6.3.1 Fitness evaluation

As there isn't an objective way to evaluate the individuals an approximation has to be made. As with any approximation compromises have been made which limit the system. Ultimately these result in the fitness function not accurately modelling the actual problem.

Many assumptions are currently made about the the moves used. There is the assumption that moves have no side effects and only deal damage to the opponent, which is grossly incorrect in fact there is a whole category of moves which only produce these side effects. This will mean that moves are only rated for the damage and not the utility. Another assumption is that a move can be used unconditionally, which while mostly true does not hold for some moves. These moves are substantially higher powered than other moves but are balanced by these conditions, for example Hyper Beam causes the user to miss a turn after being used. By not evaluating these conditions it rates solutions with this higher than they should be.

The fitness evaluation also doesn't take into account some team synergies but rather grades the team based upon the contributions of each member. Theoretically there could be a team where one member is only there for a small case where it covers another team member, this would be a successful strategy as it gives better coverage across different opponents. This team would be effective in practice as when not needed the other team member will carry the more niche members to victory but when required the other members will counter the opponent. In the evaluation this comes would score poorly as each team member is equally weighted, being biased towards teams with all rounders rather than strong specialists.

6.3.2 Pokéapi

A major limitation of the project is it reliance on the Pokéapi system. The api uses HTML requests to send the data to the user and therefore the host will have hosting fees. As Pokéapi is free to use and maintains its costs via donations, it sets counter measures to ensure that users don't abuse the api. As with many APIs this is set as a rate limit, a set number of times a client can use the API within a time frame.

A few times during development this limit was hit, blocking the IP address of the client computer requesting the information. This resulted in a temporary pause on the development happening at the time as the examples couldn't be ran. Fortunately this was only a temporary ban and it was lifted after a few hours each time.

Although this limit wasn't hit when running several instances of the final code, it could theoretically be reached with larger population sizes or more instances running at once. In its current form the program will crash if this limit is reached. This could be tackled by error handling but it would still end up pausing the program for a significant time while the limit is reached.

Another limitation of this API is it reliance on an internet service to function. This is not a major limitation of the project as due to the run times it is unlikely people would require it to run in a mobile environment. On top of not being able to function with out an internet connection, the API implementation is not built to handle short drops in connectivity crashing the program due to the error. Although this is unlikely to happen it is a limitation that can cause issues.

Another issue with this API is that over the course of the project a new Pokémon game was released, introducing new Pokémon and moves. Pokéapi hasn't been updated to include these updates, meaning that the EA isn't running with all the available data. This is a minor issue with the latest updates as they don't have major changes and solutions generated will still be valid. There is no guarantee that Pokéapi will continue to be updated in the future which could leave the project out of date in the future. If it was updated though there is no native way to collect the

minimum and maximum ID values, these would have to be found again and added to the constants file.

7 Summary and Reflections

- Project Management
  - Time and Resource management
- Contributions and Reflections
  - Innovation, Creativity and Novelty
  - Personal reflection on plan and experience

7.1 Reflections on Work Plan

Through out project management has been a key consideration, both to allow clear progress to be seen and to get updates on how to move forward. To aid with the management of the project, tasks have been reconsidered to make progress easier or to give time to tackle arising problems. To both show and predict progress a work plan was created and has been updated to add in new tasks or extend others.

To aid with project management and accountability frequent meetings have taken place. These have often discussed upcoming tasks in relation to the work plan and what work has taken place. For each meeting a short set of minutes was created and emailed to attendees, allowing for all parties to know what was discussed and what was to happen before the next meeting. All of the minutes can be found in the appendix.

As set out in the project proposal, a work plan has been followed to both show and predict progress. This has been updated to show many changes such as new tasks or extensions, these updates can be found in the appendix. The highlighted task indicate changes such as extensions, moving to accommodate delays, or a new task entirely. I have outlined the major changes to the work plan in the paragraphs below discussing the reasoning behind these changes and any risks and outcomes of these changes.

As a significant amount of the interim report could be written earlier on, it was decided to start work on it earlier; also the amount of work needed for the interim report was underestimated in my initial estimations. To facilitate these changes the time spent on the work plan was extended by starting 3 weeks earlier. This was also to combat that the week of the interim report due date there was a lot of deadlines, by starting earlier this mitigated the risk that there would be less time in the final week.

Something that wasn't initially planed for was the running a benchmark function on the outline of the GA. This task was added after a discussion that it would provide proof that the outline was working correctly without the need of the rest of the project to be implemented. This also helped create a general idea of how the other parts of the GA would work on a technical level such as how the individual genes/values would be encoded and accessed. To allow for this and any required modifications 2 weeks were scheduled, although this took time from other aspects it allowed for confidence and testing of the GA structure. This also brought up potential issues for later down the line that have already been tackled on a smaller problem.

When researching representation it was found that the data import methods would be more closely tied to the underlying representation than initially realised. Through the use of Pokéapi a higher level representation was achieved with ease of use in collecting the data. Research into representation was extended and split to allow more time to focus on how to integrate Pokéapi into the project. By spending more time on the underlying representation and how the data is accessed other tasks are being pushed back but this will allow for the project to be more adaptable in the future.

Some of the tasks have took longer than initially planed. The majority of this can be attributed to an unfamiliarity with python, a shortcoming that was reduced been overcome as the project

progressed. Other shortcomings have been with other work loads and how they have been balanced; this was a constant struggle with some weeks seeing little development of the project due to deadlines in other modules and other weeks making more progress than scheduled due to a lack of other work. The largest loss of time to another module was G53DIA as it contained a heavy coursework element, to show this it has been added to the work plan.

A minor change to the initial work plan is the allocation of time for the creation and finalising of a presentation or demonstration. This wasn't initially outlined previously as the date for the presentation or demonstration hadn't been set.

7.2 Self Reflection

7.3 Project Appraisal

The project seems to show that there maybe potential in the area of pregame decision making. This highly novel application of evolutionary algorithms and shows some shortcomings of the methods as well as some potential.

Evaluation of a population is extremely difficult when there is no clear metric to rank individuals by. The use of grading relative to the population is possible but takes exponential time with respect to the population size if every individual is rated against every other individual like in the project. This was by far the biggest time sync when running the project. Other methods for objective functions could be looked into in the future. Having an expert create a grading system could be beneficial but could limit the EAs potential to find novel solutions, as shown with AlphaGoZero. A similar problem arrises if you evaluate the solutions against known good solutions as you may be overfitting to the known solutions. A way this could be overcome would be evaluating an individual against a subset of the population, whether x random individuals or with a tournament bracket.

Another main slowdown of the project was the multiple database lookups that had to be performed to provide relevant data. The way the API (Pokéapi, specifically the Pokebase implementation) functions is that it checks to see if the data is stored on disk in a cache otherwise it looks requests it over HTML. Initially for the first time running there is no cache of the data, this particularly happens in the first generation. As the algorithm progresses there are only incremental changes meaning the cache has most of the relevant data. Even though the data is stored on disk, it can still take a long time to access this all when required. Improvements to this could help improve the speed of the run time allowing for more generations to be ran in the same time span.

The reliance on an external API has brought a lot of insight to the project. There are many limiting factors to using such an API, particularly one that is maintained by freelance developers who have no obligation to update the API. As stated in the project limitations this is the main source of technical limitations to the project but the time saved in development offsets these concerns. The use of APIs particularly Pokéapi is something to consider if the project was taken further as it would rely on these systems heavily.

The project has shown some positive results as well,

- Pros of the project

7.4 Further Considerations

If the project was to be expanded further there are many avenues that could be explored, as well as several improvements to make.

7.4.1 Reducing the Search Space

Like with most AI systems reducing the search space helps tremendously in the effectiveness of the system. This increase could be used to run a larger population, run for more generations or produce an output significantly faster. In section 4.1 the size of the search space was given an rough estimate of 5.893×10^{261} $(4sf)$ this could greatly be reduced for this problem.

A large part of this large search space is that a team is comprised of 6 Pokémon, if the EA was set up to instead optimise a single Pokémon the search space drastically reduces. Another reduction

would come from removing characteristics that don't effect the performance of the Pokémon in battle. Many games have these cosmetic features that the AI wouldn't need to optimise as they provide no functional difference. This could be take further by not trying to evaluate characteristics that relate to fringe cases that have easy to understand heuristics that players use. In Pokémon the effects of happiness and gender are limited to very specific moves which human players already know how to optimise. Potentially further reductions could come by implementing expert knowledge to limit certain values in the system to a known good range / sets. For example rather than allowing any item to be chosen a preselected set of items could be created by an expert or from a list of commonly used items. This could be expanded even further by limiting characteristics such as EVs and IVs of a Pokémon to extreme values which are seen as optimal depending on the case.

If the search space was reduced, by optimising a singular Pokémon and removing characteristics that only are required for certain cases, it could be estimated by the following equation.

$$species \times IVs^{stats} \times EVs^{stats} \times levels \times abilities \times items \times \binom{moves}{moveslots} \tag{9}$$

$$807 \times 31^6 \times 256^6 \times 100 \times 3 \times 354 \times \binom{728}{4} = 2.485 \times 10^{41}\ (4sf) \tag{10}$$

This shows a decrease of 10^{220} which is a significant decrease. Further advantages to this would be that there is less factors to be validated which would speed up the program as well.

7.4.2 Software Optimisation - Threading

Software optimisations could be made to speed up the program as a whole. The main way this could be done would be to take advantage of the multicore or multithreaded processors that are abundant nowadays. EAs have large potential for threading as many parts can happen in parallel due to the population based search methods. Initialisation and Evaluation could all happen in parallel rather than sequentially, and the creation of children could also be done in parallel. The reason for not doing this was Pokéapi couldn't handle multiple calls at the same time and would require careful design to avoid crashes. Alongside this more conventional methods of optimisation could take place to make sure the system runs more efficiently.

7.4.3 Improvements to Fitness Evaluation

Major research and improvements can be made into evaluating individuals of a population relative to each other. Many conventional EA algorithms use the fitness evaluation to apply population replacement and parent selection. Some of these methods such as roulette wheel selection already chooses individuals based upon their relative fitness to each other. If these methods are already not using the raw fitness value of more concrete problems and showing strong results there could be potential that an objective value doesn't have to be given to an individual.

Comparative fitness makes sense for evaluating game based solutions, where the individuals in the population can play against each other. With larger populations and more complex games, comparing the population by simulating a game between all individuals will be a monumental task and result in extremely long run times as seen in this project.

There is potential to see whether individuals can be accurately ranked by comparison to only a subset of the population to reduce run times. Creation of a tournament bracket could be used and integrated within tournament selection as a potential way to reduce the number of comparisons. Another candidate could be to compare all individuals to a random subset of the population each generation, changing the random subset each generation. This random subset could be comprised of a proportion of the initial population or a fixed amount.

Another improvement could be made with better approximations of the game when comparing 2 individuals. The more accurate the end result can be predicted the better the EA will be at creating solutions to the problem. The best case would be to simulate the games between the individuals but this brings up the issues of needing a separate AI to play the games for both individuals.

8 Appendix

8.1 Code

- Add in the Code here with the various subsections

8.1.1 Pokémon abilities via Pokéapi

```
# POKEMON_SPECIES_ID is the species ID of the individual
# POKRMON_ABILITY_SLOT is the slot of the ability to find
pokemon = pb.pokemon(POKEMON_SPECIES_ID)
abilities = pokemon.abilities
ability = None
for i in range(len(abilities)):
    if( abilities[i].slot == POKEMON_ABILITY_SLOT ):
        ability = abilities[i].ability
# The ability of the individual is now stored in ability
```

8.2 Meeting Minutes

6th October Overview and recap of the project idea, including background and information on the topic.
Discussion of the upcoming deliverables and deadlines.
Agreed upon a deadline of the 11/10 for the project proposal and ethics form as well as the next meeting to discuss it on the 18/10.

26th October Recap of the project proposal and confirmed it has been submitted.
Using python as the language to develop the project in Overview of the next couple of weeks as per the work plan; revision/research of evolutionary algorithms and installing basic software.
Brief discussion on the possibility of writing up any research taking place, I will be looking into how to tie this into the interim report and bring up ideas at the next meeting.
Agreed to meet on 2/11 at 2pm.

2nd November Spoke about the potential of drafting the interim report over the course of the project, agreed to do this continually through out and send the first iteration before the next meeting with sections and some bullet points.
Overview of the next stages in the plan.
Suggested having the basic GA outline to be running on a benchmark test to prove its working, this is to be shown at the next meeting.
Agreed to meet on 21/11 at 12 noon.

21st November Reviewed the outline for the Interim Report, suggesting that its good but could potentially merge a few sections together.
Demonstrated the GA structure with on the sum of squares benchmark.
Discussed writing sections of the interim report now, particularly the related work section. This will be sent before the next meeting.
Agreed to meet on 24/11 at 1:30 to review the completed section of the interim report.

28th November Reviewed the current draft of the related work section of the interim report, discussing how it can be improved.
Agreed to iterate the interim report over email with a final draft being sent by the morning of Thursday the 7th of December.
Agreed to meet on 15/12 at 12:00 to discuss the project progression and next steps.

15th December Brief Discussion of the Interim Report.
Discussed the work to take place over Christmas/Exam Periods.
Agreed to meet during the first week of 2nd term, exact times to be confirmed at a later date.

7th February Discussed the mark and feedback of the Interim Report.
Discussed progress made over the past weeks.
Brought up that the project is running slightly behind but it is believed that it can be caught up soon enough.
Agreed to meet on 15/2 at 11:00 with a working demo of the code running, this just requires a basic objective function being written.

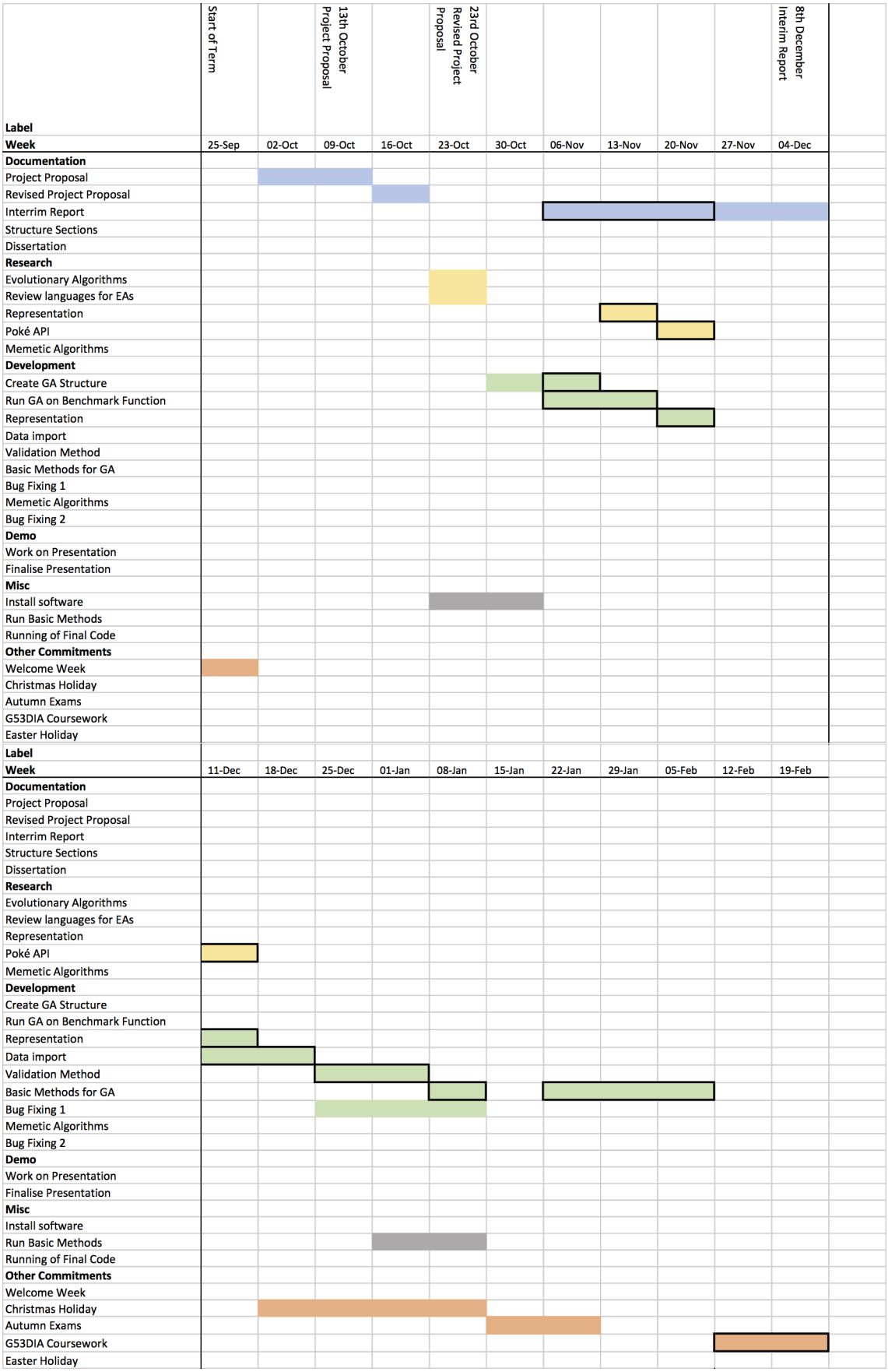
15th February Discussed the current progress, unable to show a demonstration of the running code due to a bug in the objective function evaluation.
Discussed the long running time of the code and how to overcome this for demonstration purposes, agreed that the demo day would work best as a more formal presentation containing the results and that output could be sent via email to show the project running.
Agreed to send the output of a working solution once the bug has been fixed, next meeting to be agreed once that has been completed.

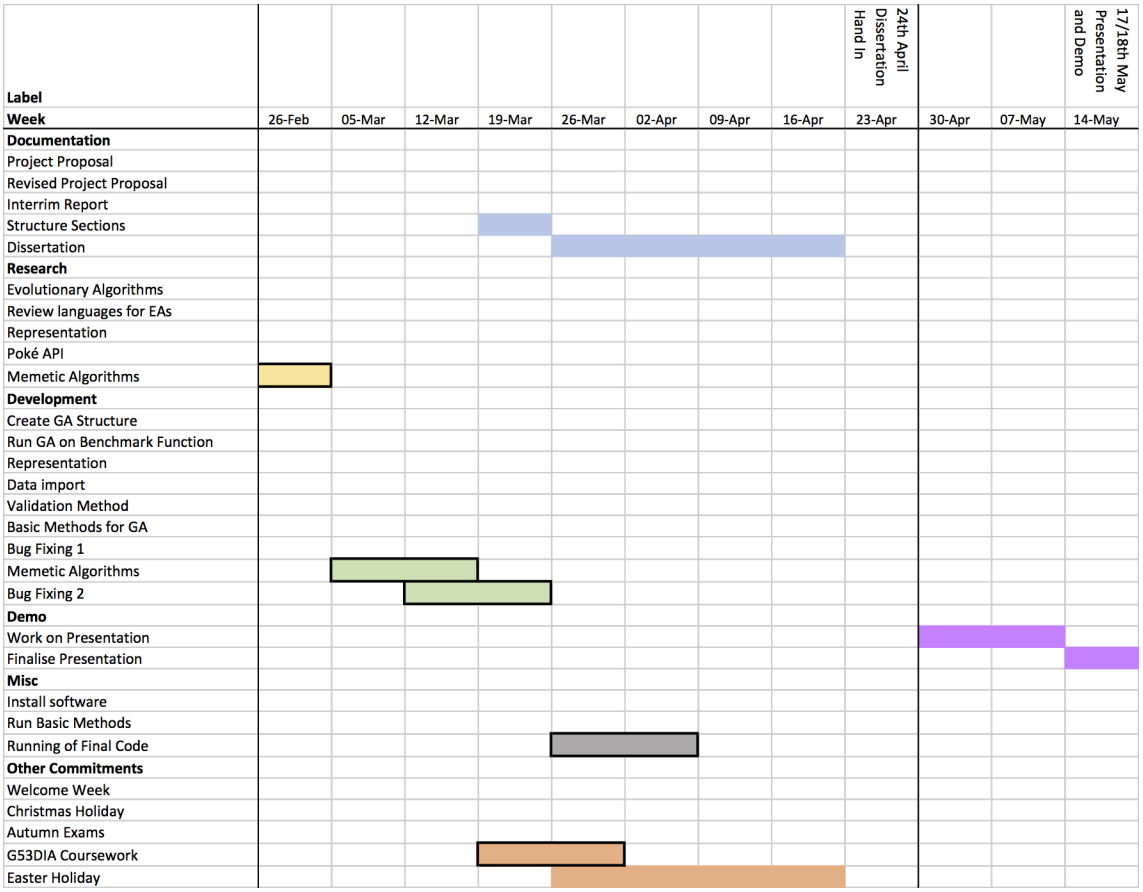
9th March Discussed the next stages of the project, prioritising adaption to a memetic algorithm and more advanced methods.
Discussed future meetings involving the writing of the dissertation, emails with feedback for each chapter.
Creation of class diagrams for the dissertation to show the overall structure of each class and how they interact.
Next meeting to be arranged over email as and when required.

6th April Discussed current position of the whole project including dissertation progress Discussed the based way to approach the running of the program to get data for evaluation.
Discussed adapting the benchmark function to show the MA works Agreed to send the current state of the dissertation, mainly the related work section.
Agreed to send first draft before the morning of the 16th April, with most sections fully completed.
Next meeting for 16th April at 4pm.

16th April Discussed the current state of the dissertation and what needs to be added.
Suggested the addition of a background on Pokemon and EAs in the introduction.
Further additions and discussions to happen over email, sending additions as they are written.

8.3 Work Plan


Start of Term
13th October Project Proposal
23rd October Revised Project Proposal
8th December Interim Report
Label
Week
25-Sep
02-Oct
09-Oct
16-Oct
23-Oct
30-Oct
06-Nov
13-Nov
20-Nov
27-Nov
04-Dec
Documentation
Project Proposal
Revised Project Proposal
Interrim Report
Structure Sections
Dissertation
Research
Evolutionary Algorithms
Review languages for EAs
Representation
Poké API
Memetic Algorithms
Development
Create GA Structure
Run GA on Benchmark Function
Representation
Data import
Validation Method
Basic Methods for GA
Bug Fixing 1
Memetic Algorithms
Bug Fixing 2
Demo
Work on Presentation
Finalise Presentation
Misc
Install software
Run Basic Methods
Running of Final Code
Other Commitments
Welcome Week
Christmas Holiday
Autumn Exams
G53DIA Coursework
Easter Holiday
Label
Week
11-Dec
18-Dec
25-Dec
01-Jan
08-Jan
15-Jan
22-Jan
29-Jan
05-Feb
12-Feb
19-Feb
Documentation
Project Proposal
Revised Project Proposal
Interrim Report
Structure Sections
Dissertation
Research
Evolutionary Algorithms
Review languages for EAs
Representation
Poké API
Memetic Algorithms
Development
Create GA Structure
Run GA on Benchmark Function
Representation
Data import
Validation Method
Basic Methods for GA
Bug Fixing 1
Memetic Algorithms
Bug Fixing 2
Demo
Work on Presentation
Finalise Presentation
Misc
Install software
Run Basic Methods
Running of Final Code
Other Commitments
Welcome Week
Christmas Holiday
Autumn Exams
G53DIA Coursework
Easter Holiday



8.4 References

References

[1] Pokemon showdown. `https://pokemonshowdown.com/`, December 2017.

[2] Bulbapedia damage calculations. `https://bulbapedia.bulbagarden.net/wiki/Damage#Damage_calculation/`, April 2018.

[3] Pokeapi github repository. `https://github.com/PokeAPI/pokeapi/`, April 2018.

[4] Maumita Bhattacharya. Evolutionary approaches to expensive optimisation. *arXiv preprint arXiv:1303.2745*, 2013.

[5] Murray Campbell, A Joseph Hoane, and Feng-hsiung Hsu. Deep blue. *Artificial intelligence*, 134(1-2):57–83, 2002.

[6] Esports Earnings. Highest overall team earnings. `https://www.esportsearnings.com/teams`, October 2017.

[7] Esports Earnings. Largest overall prize pools in esports. `https://www.esportsearnings.com/tournaments`, October 2017.

[8] Pablo García-Sánchez, Alberto Tonda, Giovanni Squillero, Antonio Mora, and Juan J Merelo. Evolutionary deckbuilding in hearthstone. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE, 2016.

[9] Griffin McElroy. Becoming the very best: The pokemon world championships. `https://www.polygon.com/2013/7/20/4539528/becoming-the-very-best-the-pokemon-world-championships`, July 2013.

[10] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[11] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[12] Robert E. Smith, B. A. Dike, and S. A. Stegmann. Fitness inheritance in genetic algorithms. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, SAC '95, pages 345–350, New York, NY, USA, 1995. ACM.

[13] T.C. Sottek. The world's best dota 2 players just got destroyed by a killer ai from elon musk's startup. `https://www.theverge.com/2017/8/11/16137388/dota-2-dendi-open-ai-elon-musk`, August 2017.

[14] Carlo Zapponi. Githut. `http://githut.info/`, December 2017.

[15] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271, 1999.