The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Benjamin Joshua Charlton

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

2018/04

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

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 an evolutionary algorithm.

Acknowledgements

- Both of these sections shouldn't be proper sections

Contents

1 Introduction

- What can be done to improve this section

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[5][4]. 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[3] and Go[8]. During The International 2017 for DotA 2 this all changed as the worlds of AI and eSports came together in a 1v1 show match[11]. Elon Musk, backer of this initiative, said that this is 'Vastly more complex than traditional board games like chess & Go'[11]. 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[7]. 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.

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[3]. 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[8]. 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[9]. 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[6]. 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'[6], 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[6].

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[6].

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'[6], 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'[6].

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"[13]. 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"[13].

Maumita Bhattacharya spoke about differing approaches to the problem of computationally expensive fitness evaluation[2]. 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.[10] 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

- What the project is meant to achieve
- How is it meant to function
- Brief list of functional requirements

3.1 Goals of the project

3.2 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

- Expand current sections

4.1 Evolutionary Algorithms

- Choice of GA and MA

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.

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[12] 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

- Should this be moved into the evaluation

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:

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

- Would more results be better

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

- Description of the designs
- How it addresses the problem
- Why it is designed that way
- Languages/platforms chosen
- Problems encountered
- Design changes based upon 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 highest 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

5.2 Representation

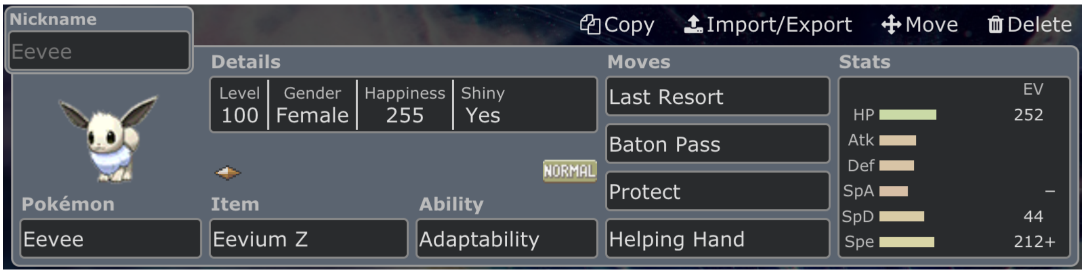
In any evolutionary algorithm the representation of the individuals is a key consideration in both design and implementation.

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 from an expert designed team.



Nickname: Eevee

Details: Level 100 | Gender Female | Happiness 255 | Shiny Yes

Pokémon: Eevee

Item: Eevium Z

Ability: Adaptability

Moves: Last Resort
Baton Pass
Protect
Helping Hand

Stats	EV
HP	252
Atk	
Def	
SpA	–
SpD	44
Spe	212+

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. 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 (Separate values for HP, Atk, Def, SpA, SpD, Spe)
- IVs (Seperate values 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.

Characteristic	Stored As	Valid Range	Associated Pokéapi Call
Pokémon Species	Integer ID	1-10147*	pb.pokemon(ID)
Gender	Integer ID	1-3	pb.gender(ID)
Item	Integer ID	1-918*	pb.item(ID)
Ability	Integer ID	1-3	NOT SURE
Level	Integer Value	1-100	N/A
Shiny	Boolean Value	True, False	N/A
Nature	Integer ID	1-25	pb.nature(ID)
EVs	Integer Value	0-255	N/A
IVs	Integer Value	0-31	N/A
Moves	Integer ID	TODO	pb.move(ID)

* Not all IDs in this range are associated with something

5.2.3 Representation Problems

6 Evaluation

- How the EA was tested
- Statistical evaluation of performance
- Evaluation of software performance

7 Summary and Reflections

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

8 Appendix

8.1 Code

- Add in the Code here with the various subsections

8.2 Meeting Minutes

- Add in any final meetings

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.

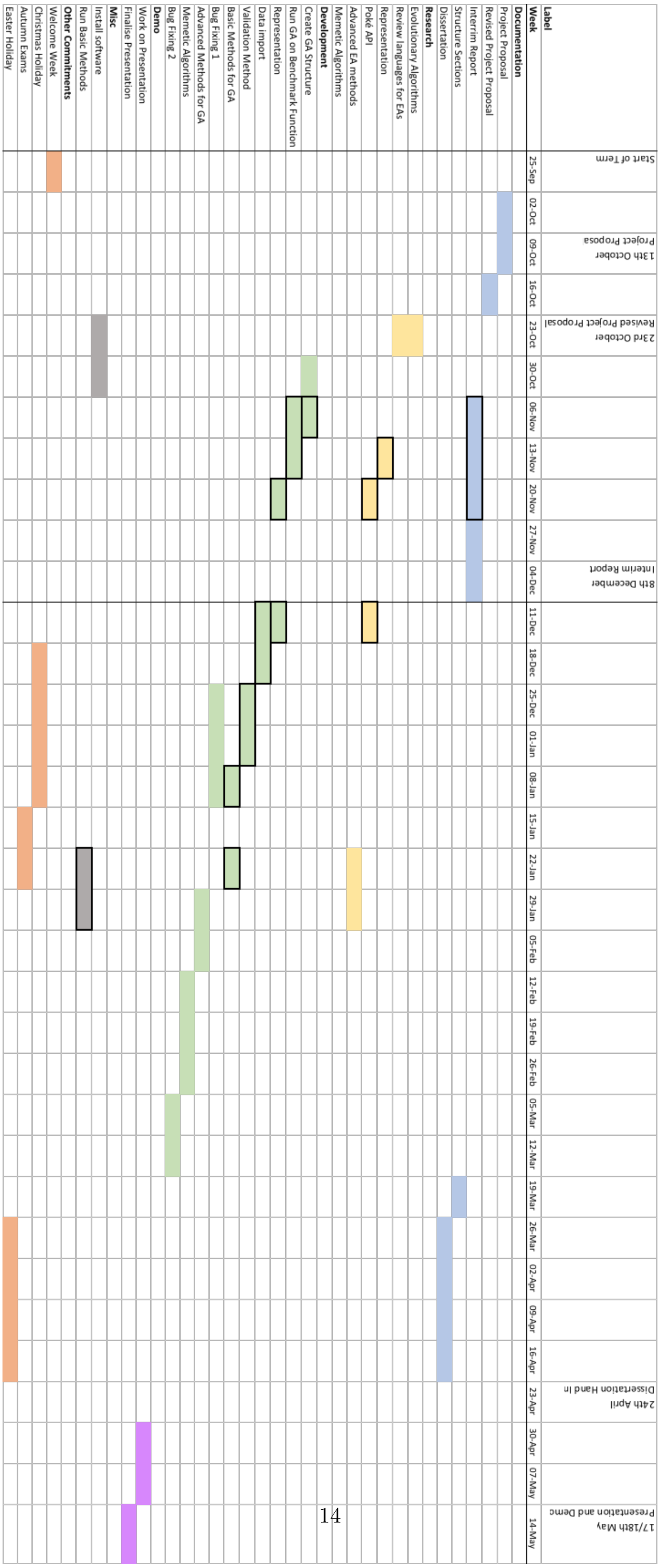
7th February Discussed the mark and feedback of the Interim Report.
Discussed progress made over the past weeks.
Brought up that the project is running slightly behind but it is believed that it can be caught up soon enough.
Agreed to meet on 15/2 at 11:00 with a working demo of the code running, this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

8.3 Work Plan


Label
Week
Documentation
Project Proposal
Revised Project Proposal
Interrim Report
Structure Sections
Dissertation
Research
Evolutionary Algorithms
Review languages for EAs
Representation
Poké API
Advanced EA methods
Memetic Algorithms
Development
Create GA Structure
Run GA on Benchmark Function
Representation
Data import
Validation Method
Basic Methods for GA
Bug Fixing 1
Advanced Methods for GA
Memetic Algorithms
Bug Fixing 2
Demo
Work on Presentation
Finalise Presentation
Misc
Install software
Run Basic Methods
Other Commitments
Welcome Week
Christmas Holiday
Autumn Exams
Easter Holiday
25-Sep
02-Oct
09-Oct
16-Oct
23-Oct
30-Oct
06-Nov
13-Nov
20-Nov
27-Nov
04-Dec
11-Dec
18-Dec
25-Dec
01-Jan
08-Jan
15-Jan
22-Jan
29-Jan
05-Feb
12-Feb
19-Feb
26-Feb
05-Mar
12-Mar
19-Mar
26-Mar
02-Apr
09-Apr
16-Apr
23-Apr
30-Apr
07-May
14-May
Start of Term
13th October
Project Proposa
23rd October
Revised Project Proposal
8th December
Interim Report
24th April
Dissertation Hand In
17/18th May
Presentation and Demo

8.4 References

References

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

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

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

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

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] T.C. Sottek. The worlds best dota 2 players just got destroyed by a killer ai from elon musks startup. `https://www.theverge.com/2017/8/11/16137388/dota-2-dendi-open-ai-elon-musk`, August 2017.

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

[13] 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.