Interim Report

G53IDS

Project Title: Applying Evolutionary Algorithms to Pokémon Team Building

4262648 Benjamin Charlton (psybc3)

8th December 2017

1 Introduction

- Talk about aims and objectives
- Overall goal with the project
- Major steps to be taken
- Some of this can be adapted from the project proposal

2 Motivation

- Talk about other AI game playing and how they approach it
- Mention how these approaches assume things like pregame decisions
- Detail the problem
- Again some can be adapted from the project proposal

3 Related Work

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 ranging from Tic-Tac-Toe to Chess to Go requiring no preparation before the match begins.

As the field of AI game playing moves forward into more complex games these 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 particular I looked into techniques that tackled having a large, vast search space and where the correctness of a solution was hard to judge. Both of these issues were things that I saw as issues that I would have to tackle to solve the team building problem.

García-Sánchez et al. tackled a very similar problem using a genetic algorithm to approach deck building[2]. 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 I am trying to achieve, as well as several short comings that I would have to take into account.

An evolutionary algorithm was used as 'they commonly produce very effective combinations of elements', 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.

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.

One of the key upsides of this was that the AI 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', 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'.

Maumita Bhattacharya spoke about differing approaches to the porblem of computationally expensive fit-

ness evaluation[1]. 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.[3] 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.

4 Description of the Work

- Detail the output specifically
- Discuss building this solution from scratch (besides using relevant APIs outside of the main project), why this is beneficial
- Talk about both the EA and MA and how they will be linked

5 Methodology

5.1 Object Orientated Design and Python

- Easy adaptation
- Conversation from GA to MA
- Commonly used, lots of APIs

5.2 Evolutionary Algorithms

- Choice of GA and MA
- Why this over other AI techniques like ANN and Local Search

5.3 Software Engineering Tools

- Git
- Latex
- Atom
- Unit Tests
- How these tools work together and help with project management and speed up work

6 Design

- Genetic Algorithm
- High level Representation Explanation

7 Implementation

- Language and Platform choices
- Separation of settings
- Low level Representation Explanation
- Validation methods (if made by then)
- Use of the benchmark example

8 Progress

8.1 Project Management

- Discuss bringing Interim Report work forward
- Adding in of a benchmark function to test the GA Structure
- Review of individual tasks and how long they took
- Revised work plan, including changes to current progress and any prospective changes

8.2 Contributions and Reflections

- Difficulties due to unpredicted work loads in other areas
- How moving forward aspects and adding additional tasks helped and hindered
- Current achievements in the project
- Personal Reflections

9 Appendix

References

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

[2] 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.

[3] 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.