Caleb Dinsmore and Adam Pogwizd
Dr. White
Machine Learning
6 April 2016

Project Proposal II

Specific Problem We Are Solving

As stated in our previous proposal, we will be attempting to create a winning Pokémon team using a combined genetic algorithm and genetic programming solution.

Specifically, we will be attempting to create a team that will be able to defeat multiple teams composed of Pokémon in a higher tier.

- What this means:
  - Tiers – Smogon, a website and community dedicated to Pokémon battling, has every Pokémon ever created separated into tiers. Different Pokémon have very different stats and abilities, and Smogon aims to divide Pokémon of like power into tier groups. The two tiers that we will be focusing on are OverUsed (OU) and Uber (U). The U tier is designated for Pokémon that are so powerful that they should be disallowed from entry in tournaments. Essentially, this tier is the ban-list.
    - Thus, the teams we will be battling against will be composed solely of Uber-tier Pokémon, while ours will be limited to Pokémon only from OU and below, giving us a distinct disadvantage and an interesting problem to solve.

Experimental Design

1. Genetic Algorithm

- This part of the experiment is solely responsible for **team selection**.
  - The goal of this half of our solution is to figure out a team selection that has the highest potential to perform the best against the opposing teams.

1.1 Data Representation

- The Chromosome
  - Each Pokémon has its own ID number, and every possible move any Pokémon can make also has its own ID number. For our chromosome for this part of the solution, we decided to utilize these numbers.
  - Example chromosome: [“393007478256004”, “340042978498004”, …] (up to six Pokémon can be in a single chromosome)
    - See example of a single Pokémon’s genetic breakdown below [Figure 1].

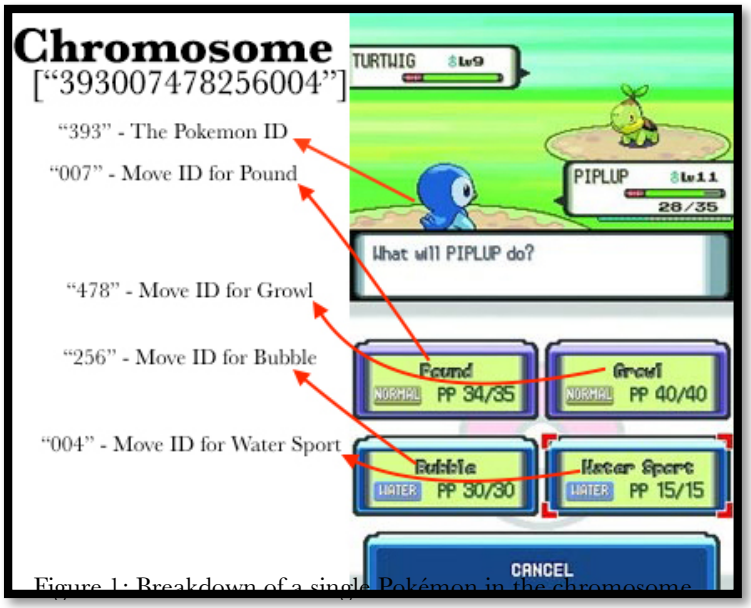

Figure 1: Breakdown of a single Pokémon in the chromosome

2.2 Fitness Function

- Scoring each individual will be based on health. Each Pokémon in a team has a certain number of health points. We will be concerned both with the percentage of total health that our team eliminated from the opposing teams and with the percentage of health the opposing teams took from us.
    - Scoring will be computed as follows:
        - <Percentage dealt to other team> - <Percentage taken from our team>
        - Thus, a positive score means that we completely defeated the other team.
- This fitness will be computed through simulated battles using the same basic AI for both sides. We will be developing the AI and the battle mechanism ourselves.
    - The AI will use the following heuristics for move selection:
        - If its current Pokémon has a damage-dealing move that is super-effective against the opposing Pokémon, use it.
        - If another Pokémon in its party is a better type match against the opposing Pokémon, switch to that Pokémon.
        - If its current Pokémon is a very poor type match against the opposing Pokémon (i.e. the other Pokémon's type is super effective against its current Pokémon), switch to another Pokémon.
        - If its current Pokémon is low on health and has an HP-restoring move available, use it.
        - If it would be advantageous for a Pokémon to use a stat-boosting move (e.g. if the Pokémon has a significantly high attack stat), use that once.
        - If there is no particular move that has a significant utility in the current situation, deal the highest damaging move possible.
    - More information about Pokemon battle format can be found here.

        - Note: We are intentionally omitting the use of items for the sake of simplicity.

2.3 Parameters

- **Crossover Operator/Rate** – single-point crossover at a rate of **0.6**.
- **Mutation Operators**
    - We will have two different types of mutation occurring in this algorithm: **Pokémon mutation** and **move mutation**.
        - Pokémon mutation will occur at a rate of **.005**. This will consist of a random Pokémon being selected from an individual's team and switched with another random (but eligible—i.e. from the same tier and below) Pokémon.
            - This introduces a problem; if the new random Pokémon is unable to use the previous Pokémon's move set, we potentially introduce an invalid Pokémon to the population. We will solve this by doing a "smart mutation." Shared moves between the previous Pokémon and its mutated replacement will be kept, and the other moves will be replaced with random moves from the new Pokémon's available move library.
        - Move mutation will occur at a rate of **.01**. This will consist of a random move from a random Pokémon being randomly replaced by another move from that Pokémon's available move library.
- **Population Size – 100**
- **Generation Limit – 10,000 generations**

2.4 Convergence

- We are defining convergence in this case as when the entire population has converged

2.5 Experimentation Details

- We will run this experiment **30** times across multiple machines. Since this sort of experiment has never been attempted before (that we could find), we are unsure as to how long each iteration will run. This will largely be dependent on how well we implement battle simulations and the AI.
- Whichever team configuration the genetic algorithm converges on the most will be used in the following part of our solution.

3. Genetic Programming Solution (Optional)

- This part of the solution is optional and will be written if there is time available. This part will consist of the evolution of the AI for the team selected by the previous part. I.e. whereas the genetic algorithm found a potentially ideal Pokémon team configuration, this part should evolve an optimal AI for this team specifically.

3.1 Data Representation

- Using a tree structure, we will evolve a program consisting of the following decisions:
  - Switch to Pokémon 1
  - Switch to Pokémon 2
  - Switch to Pokémon 3
  - Switch to Pokémon 4
  - Switch to Pokémon 5
  - Use move 1
  - Use move 2
  - Use move 3
  - Use move 4
- The decision tree will take in the following inputs:
  - Opponent Pokémon type
  - Type of Move 1
  - Type of Move 2
  - Type of Move 3
  - Type of Move 4
  - Damage of Move 1
  - Damage of Move 2
  - Damage of Move 3
  - Damage of Move 4
  - Type of Pokémon 1 (Party)
  - Type of Pokémon 2
  - Type of Pokémon 3
  - Type of Pokémon 4
  - Type of Pokémon 5
- Each node in the decision tree will have an if-statement that will compare any one of these inputs with a value also stored in that node. If the statement evaluates to true, we will descend further down the tree through the right child, else the left—continuing until we reach a leaf node (i.e. a decision).

3.2 Fitness Function

- Fitness for the genetic program will work exactly like it did for the genetic algorithm, except our team will be using the AI generated by the genetic program and not the basic AI used before. The basic AI will still be used by the opposing team.

3.3 Parameters

- **Crossover Operator**
  - Crossover for this part will consist of randomly selecting a sub-tree from each of the parents and swapping them.
  - Rate: **0.6**
- **Mutation Operator**

    - We will also need to utilize a "smart mutation" approach with this part of our solution.
        - If mutation occurs, a random node in the individual is selected.
        - Then, the program essentially flips a coin. If heads, we mutate the operator (<, >, ==, etc.), else mutate the value.
            - We will also need to be smart about value mutation. For example, if the value is a type ("Water", "Grass", "Rock", etc.), we should replace it with another random type. If it's a number, we should just randomly replace the number.
    - Rate: **0.01**
- **Population Size – 100**
- **Generation Limit – 10,000 generations**

3.4 Convergence

- Convergence is a little trickier to define in this part of our solution. We decided to declare that a population has converged on a solution when there has been no significant improvement in fitness across **5** generations.

3.5 Experimentation Details

- We will run this experiment **30** times across multiple machines (the same as the previous part).
- The final solution will be a team and AI with the highest fitness compared against all 30 results.