Date of submission 08-12, 2023

Game Playing Pokémon Showdown Artificial Intelligence (December 2023)

Evan Switzer

ABSTRACT This paper investigates game playing artificial intelligence and it's uses on the video game simulator website Pokémon Showdown. It takes a look a multiple algorithms and their effectiveness at winning a Pokémon battle. It investigates the strategy behind random battles and what a bot may excel or fail at and why.

INDEX TERMS Pokémon, Stat, Buff, Algorithm, Bot

I. INTRODUCTION

Pokémon Showdown is an online website where players can emulate Pokémon battles. There is a wide range of formats ranging from the first Pokémon game to the current year's game. This paper will focus on the current generation of Pokémon, more specifically random battles in this current generation. Each format comes with it's own challenges. In the random battle format players go up against another player, each having 6 random Pokémon. Only the 1st Pokémon is revealed to the other player. Pokémon have standardized stats. Each pokemon has a 6 base stats; HP, attack, defense, special attack, special defense, and speed. These are dependent on the Pokémon. Next are effort values, these are a hidden stat for each Pokémon. Effort Values can range from 0 to 255, in other formats where team creation is a part of the format these can be changed to enhance Pokemon's stat slightly. For random battles each effort value is set to 88. The other hidden stat is individual values, this also gives a slight boost to Pokemon's stats. All of these are set to max unless a special case happens which is irrelevant to this paper. Random battles have adjusted levels for each specific Pokémon. A level is the amount of power each Pokémon could have. Levels range from 1 to 100, with 100 giving the most power. Pokémon Showdown uses levels to balance the random battle format, giving weaker Pokémon more levels and giving stronger Pokémon less levels. The last difference and the main reason random battles were chosen is that the moves each Pokémon can know are mostly pre-determined. A move is an action the Pokémon can use, usually to deal damage to opposing Pokémon. Each Pokémon has a set of moves they can know, with percentage change that they end up with that move[1].

II. Existing Work

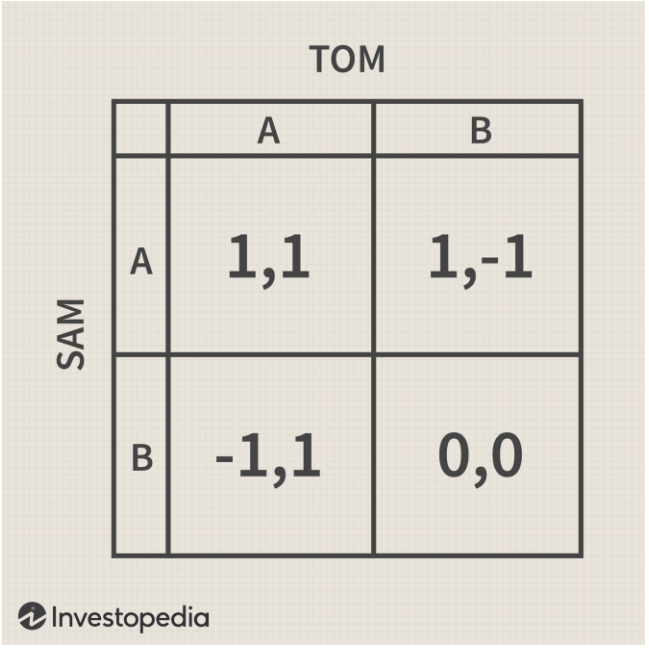
Pokémon Showdown has a large following of developers, this was used to not need to build a bot from scratch. A Pokémon Showdown server[2] was locally hosted to not affect anyone else while developing the bot.

A Pokémon battle-bot was used as the base to develop on top of. This bot came with multiple algorithms already, so I investigated those to see what they had already done. Using this battle bot as a building block was extremely useful but still very time consuming as understanding how everything works was not as easy as originally assumed. The difference between the artificial intelligence game playing bot that was created for this paper and other existing ones is the game mode and algorithms attempted. Generation 9 is the newest generation of Pokémon and thus has the least work done for it. The bot that I am using as a base goes up to generation 8. Other machine learning artificial intelligences are often in older generations often 10 to 15 years old.

III. Algorithms

A. NASH EQUALIBRIUM

Nash equilibrium is a solution concept in game theory. It states that with both players in a non-cooperative game know the optimal strategy the players would make the best choice assuming the opponent plays their best option and know what either player will do has no effect on the choices of both as they have already chosen the option that maximizes their response to the opponent. This is an interesting algorithm but it has a few issues when it comes to Pokémon. Pokémon does not have consistent damage, the amount of damage dealt by a single move can vary by about 10%, along with this there is a chance for a critical hit every attack. A critical hit does 1.5 times damage and ignores negative stat changes. Even with this randomness in Pokémon it should still be possible to find a true Nash equilibrium since Pokémon works on a health point system. Health points or HP means that each Pokémon has a certain set of health and attacks will take away a certain amount of health. This means that the randomness turns into a range of values it can be, not any real number which would be infinite. Unfortunately, this randomness still creates a great deal more computation when finding the optimal path. To simplify the calculations the implementor of this algorithm takes the max damage value. To find the Nash equilibrium the algorithm creates copies of games with the different options being chosen, calculates a score for each option and continues more layers down to branch out and choose next turns options. This implementation is good but does not actually create a Nash equilibrium ever. Of course there are limitations in place so it can run without having to think through every possibility, but even on a supercomputer it cannot figure out the optimal path as there are unknowns of what Pokémon the opponent has and what specific moves each Pokémon has.



B. MOST DAMAGE

Most damage is an extremely simple algorithm. It checks all of the players current moves and how much they do to the opposing players Pokémon. It then chooses the move that results in the most damage.

C. SAFEST MOVE

Safest move is more complicated than most damage but not as complicated as Nash equilibrium. Safest move takes into account the opposing players options as well as their own. It can do a depth first search same as Nash equilibrium to look past just the next move. It chooses the option with the highest score and uses that move.

D. GENERATION ONE

Generation one is barely an algorithm. It was the first test to make sure things were working. It will prioritize choosing

super-effective moves over all others. A super-effective move is based on the type of the move used against the type of opposing Pokémon. Each Pokémon and Move has a type, such as grass, fire, or water[3].

So, generation one will use a super-effective move first, if there are multiple it will choose the one that does the most damage, if there are none it will then choose the one that does the most damage. This means that a Pokémon could end up using a move that would be super-effective but that does no damage. This is an homage to Twitch Plays Pokémon[4] where the final battle of the game was won because the computer controlled Pokémon was using a move that did no damage but was super-effective.

E. MIN-MAX

Min-Max was the first real algorithm I attempted. Min-Max is similar to Nash equilibrium and probably a better way to describe what was created by the original creator. The goal of a min-max algorithm is to create a tree that gets the player to the end goal of winning, or at least to a value found acceptable deep in the tree. The tree is made of alternating layers, the first one being a maximizer, the second one being a minimizer. The maximizer layer would be the player using the move that gets the best result. The minimizer layer would be the opponent using the move that gives the player the worst result, or the opponent the best result. This has some issues as Pokémon turns happen at the same time. The speed stat determines who goes first in a turn so technically it could work but the layers would not be alternating as sometimes the move order would be ABBA as opposed to ABAB. Min-max did not seem to be the ideal solution here because of how turn orders worked but the recursive tree that was used felt useful.

F. NAÏVE-MOVE MATRIX

The naïve-move matrix creates a matrix of every possible Pokémon the player could get and every possible Pokémon the opponent could have. This would be calculated at the start of the program before the bot even touches a battle. This would allow for the matrix to be used as a lookup table, giving the players current Pokémon, and the opponents current Pokémon and getting back the best move to use. This has issues as it does not think about the future.

G. FUTURE ALGORITHMS

For the future to find an algorithm that works the best it would be worth considering combining multiple. The move matrix is excellent because it is fast, while Nash equilibrium and min-max are excellent because they use a tree to find a path to the end. Combining both to allow for traversing a large tree but with fewer decisions as they are already made could create a powerful algorithm. Anecdotally, however there never seems to be one correct answer as human players often find themselves playing footsies with each other. An algorithm may get stuck in a loop of switching. Switching is changing Pokémon who are currently fighting, it takes a turn to do usually. If the algorithm got stuck in a loop of constantly switching a human player could predict a switch and get an advantage off that. This may mean that the matrix may need to be re-calculated each action, except since there is a tree there should be that option already explored. It is hard to say if this weakness is truly a weakness and would get taken advantage of but if it is it could be solved by giving each option a weight, and randomly choosing one of the best options depending on how good it is. For example option A could be good 95% of the time, while option B is the best 5% of the time. The algorithm could introduce a little bit of randomness which would selection option B rarely which could help it's predictability.

IV. ISSUES

A. DEVELOPMENT

There were numerous issues with developing an algorithm that I was happy with, and in the end, I did not end up completing an algorithm more complex than generation one. To run this bot there needed to be a server where I could have full control to allow for no unintended consequences. Setting up a Pokémon Showdown server was not too hard as there were nice simple instructions, but getting the bot to connect to it was difficult as I did not yet fully understand how they connected. I also had plenty of trouble with creating the algorithms. There were lots of useful helper functions given to me, I just could not figure out how they worked well and how to use them for what I wanted to do. I tried to both create similar things from scratch and use them how they were intended but I did not succeed.

B. ALGORITHM

While creating and thinking about the possibilities for the algorithms some issues appeared that may cause issues. In Pokémon players can boost their Pokémon's stats, this can make attacks more powerful, Pokémon harder to knock out, Pokémon faster than all other so they move first. These moves that increase stats often do no damage. Any algorithm that wanted to successfully use these moves would have to either look through a tree far enough to see the benefits of all of those buffs or have some artificial tuning that may push them to favor those types of moves. Similar to stat raising moves there are status condition and entry hazard moves. These do not do any immediate damage either but do damage over time. Without knowing what Pokémon are on the opponents team entry hazards could do very little or a large amount of damage. The uncertainty makes it hard to perfectly predict what to do. As discussed in the algorithms section as more information is revealed things should be recalculated, depending on the information given that could be time consuming. Another issue is that random battle teams are not made equal. While all Pokémon are balanced with levels to attempt to get them to an equal power area, if one player gets mostly similar Pokémon with mostly the same weakness that creates a major flaw in the team and a human would be able to play around this information, while a bot may not be able to use this information efficiently.

V. STRATEGY

Random battles have a great deal of strategy that is hard to see a bot successfully navigating. Using the original makers safest option it would use questionable moves often. Along with this is rarely switched Pokémon which is an essential part of random battles. The strategy for random battles can be broken into 3 parts: the beginning, the middle, the end. The beginning is where both players are trying to feel each other out. Neither one knows the others team yet, both players are trying to hide as much information as possible while trying to gain information from the opponent. It could be worth exploring valuing forcing information out of the opponent while hiding it's own information for the algorithms especially at the start. Often one player will either find a weakness in the opponent's team or will slowly gain an advantage. Once this happens the game moving into the middle. The middle is where most players attempt to make big plays, knock out Pokémon, and set up their own Pokémon with buffing. Players will try and get a large enough lead that they push past the end without needing to worry about it. However, if players do not get a large enough lead by the end, it turns into a game that is much easier to see the ending. One player will often be in the lead, all Pokémon will be revealed, and the losing player will either have to hope for a mistake or go for a hail-mary play. This is where a bot could have the most benefit, having almost all the information it could figure out how to close out the game the safest way possible.

VI. CONCLUSION

Random battles still have a long way to go for public bots. In this paper multiple different algorithms were explored and none of them were perfect. So for now the human reigns supreme.

REFERENCES AND FOOTNOTES

A. REFERENCES

[1] https://pkmn.github.io/randbats/data/stats/gen9randombattle.json

[2] Smogon. "Pokémon Showdown." GitHub, 2023, https://github.com/smogon/pokemon-showdown.

[3] "Pokemon Type Chart." Pokémon Database, 2023, https://pokemondb.net/type.

[4] Wikipedia contributors. "Twitch Plays Pokémon." Wikipedia, The Free Encyclopedia. Last modified Year, https://en.wikipedia.org/wiki/Twitch_Plays_Pok%C3%A9mon.