# AI Agent for Pokémon Battles

Jesse Paul Burkett, Raghav Tibrewala, Michael Meng Zhang
Department of Computer Science, Stanford University

## Introduction

- Pokémon is a game created in 1995 to catch fictional creatures and battle with others
- In a battle, both players have a maximum of 6 Pokémon's which battle each other
- Each player's Pokémon possess a set of attacks which they use **sequentially** which causes the other Pokémon to lose its health.
- The game ends when all of the Pokémon of either player completely lose their health.


*A Pokémon Battle*

## Implementation: Learning Results

**Monte Carlo Methods:**
Monte Carlo Tree Search: heuristic search through decisions:

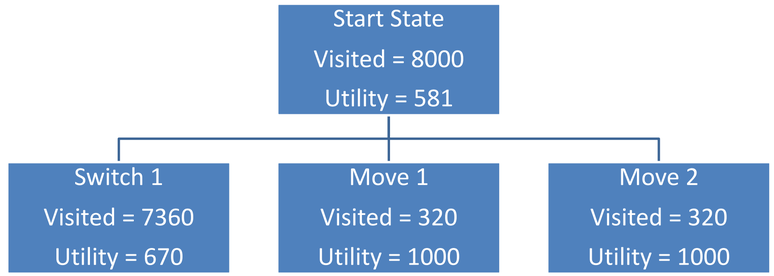
1. Selection
2. Expansion
3. Simulation
4. Backpropagation

UCT: Upper Confidence Bounds

set to $\mu + c\sqrt{\frac{\log(N)}{n}}$

$\mu$ = average utility as node
$c$ = UCT constant (determined empirically)
$N$ = number of times parent node was visited
$n$ = number of times current node was visited

**Results to show:**
- Large run time due to vastness of state spaces
- Using a UCT constant = 30

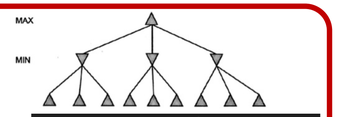



The start of the tree, root is the start state
Minimax version of the MCTS for multiplayer
Note that the first Switch has more visits, because the opposing player goes first!

## Implementation: Modeling Results

**Definitions**:
- State based AI algorithm to model a Pokémon battle
- For the adversarial game, we define:
- Players = {Agent, Opponent} rep. as +1 or -1
- States = { Array of agent's Pokémon, Array of opponent's Pokémon, current agent Pokémon, current opponent's Pokémon, whose turn it is}
- Actions = {attack 1, attack 2, attack 3, attack 4, switch Pokémon}
- If State s is: {Our Array, Opponent's Array, p , q, +1}
- Succ(s, a = attack) = {Our Array, Opponent's Array', p , q, -1}
- Succ(s, a = move) = {Our Array, Opponent's Array, p' , q, -1}
- Utility(s) = + inf if agent wins, - inf if opponent wins

**Results:**
- The minimax algorithm uses:
- Eval(s) = Σ Agent's Health – Σ Opponent's Health
- We created a game for the human to play against the bot which just minimizes the utility at every step. Starting off from same start state, the human plays a suboptimal strategy and loses.
- The minimax agent wins the same game. The agent intelligently switches based on the damage function. The list of the agent's actions is shown here.




```
Raghavs-MacBook-Air:cs221-mast
[pikachu, venasaur, charizard]
[pikachu, charizard, venasaur]
('thunderbolt', 'moves')
('thunderbolt', 'moves')
('thunderbolt', 'moves')
('quick attack', 'moves')
('thunderbolt', 'moves')
('thunderbolt', 'moves')
(2, 'switch')
('fire blast', 'moves')
('fire blast', 'moves')
('fire blast', 'moves')
('flamethrower', 'moves')
('flamethrower', 'moves')
('flamethrower', 'moves')
(1, 'switch')
('solarbeam', 'moves')
(2, 'switch')
('fire blast', 'moves')
Your Pokemon
('pikachu', ' hp: ', 0)
('venasaur', ' hp: ', 0)
('charizard', ' hp: ', 29.0)
Opponents Pokemon
('pikachu', ' hp: ', 0)
('charizard', ' hp: ', 0)
('venasaur', ' hp: ', 0)
You Win!
```

## Related Work

We find that the project has been done before. The links are given below:

1. https://github.com/Sisyphus25/CynthiAI
2. http://game.engineering.nyu.edu/wp-content/uploads/2017/02/CIG_2017_paper_87-1.pdf

We are yet to implement the expecti-minimax algorithm to account for transitional probabilities in the game. This algorithm has not been previously used to model the Pokémon battle. We will include it as a part of the final report.

## Data Source

We have used generation 1 Pokémon statistical data from the following link:
https://www.kaggle.com/abcsds/pokemon