Project Report: Pokémon Battle Simulator with AI Decision-Making

1. Introduction

This project implements a turn-based battle simulator inspired by Pokémon, with a focus on AI strategy and adversarial gameplay. Players command teams of Pokémon, selecting moves or switching team members in real-time combat. A key component of the project is the integration of game-playing AI capable of simulating future states and making informed decisions, drawing from concepts such as minimax search and heuristic evaluation.

---

2. Objectives

- Simulate turn-based Pokémon battles with minimal rules.
- Implement an AI agent that can make strategic decisions in an adversarial setting.
- Use future state simulation to inform AI decisions (i.e., simulate consequences of moves).
- Lay groundwork for more advanced adversarial AI (e.g., minimax with alpha-beta pruning).

---

3. AI System Design

3.1 AI as an Adversarial Agent

The AI in this simulator plays against an opponent, making it an adversarial agent. It operates under the assumption that each move it makes could be countered by an opponent's move. This adversarial dynamic closely follows the structure of two-player zero-sum games — a classic scenario for **game-tree-based search algorithms** like **minimax**.

3.2 Minimax Inspiration

Although the project does **not implement a full minimax algorithm**, its decision-making draws inspiration from it in these ways:

- **Simulation of Game States**: Before choosing a move, the AI simulates the outcome of using each available move using `simulate_move`.
- **Evaluation Function**: Each simulated state is evaluated by `evaluate_state`, which acts as the heuristic function in minimax. It scores the state based on:
    - Remaining HP of each side
    - Number of Pokémon fainted
    - Estimated threat level
- **Best Move Selection**: Among all available options, the move leading to the highest evaluated state is selected — mimicking the "maximize utility" behavior of the maximizing player in minimax.

This is essentially a **1-ply minimax**, where the AI only considers the immediate result of its own action — future enhancements could involve simulating **opponent reactions** (2-ply and beyond).

3.3 Heuristic Evaluation Function

The `evaluate_state` method is crucial to the AI's effectiveness. It uses a **hand-crafted heuristic** to assess simulated states, considering:

- **Team Strength Differential**: Difference in total HP between the AI and opponent.
- **Pokémon Advantage**: Bonus if the AI's Pokémon count is greater.
- **Type Advantage and Status Conditions**: (Placeholder or potential future additions)

This heuristic is designed to **approximate the utility function** of the minimax algorithm, giving the AI a way to rank possible moves.

3.4 Game State Simulation

A critical AI feature is the ability to **simulate game states without affecting the actual battle**. This is done using:

```
simulated_battle = copy.deepcopy(self)
```

This cloned battle object allows the AI to:

- Apply a move (`simulate_move`) as if it were used
- Evaluate the result without consequences
- Make decisions based on "what-if" scenarios

This mirrors the **state expansion process** in classical AI planning and search algorithms.

---

4. AI Behavior Summary

Situation	AI Behavior
Pokémon is fainted	Automatically switches to a healthy Pokémon
Choosing a move	Simulates all available moves, evaluates resulting states, picks best
Switching Pokémon	Chooses based on highest remaining HP
Status condition	AI ignores status, but this can be incorporated into the evaluation

---

5. Future AI Improvements

1. **Full Minimax Implementation**
   - Extend simulations to include possible opponent responses (2-ply or more).
   - Use **alpha-beta pruning** to reduce computation.
2. **Learning-Based Heuristics**
   - Replace hand-crafted evaluation with learned models (e.g., using reinforcement learning or supervised learning from expert data).

3. **Monte Carlo Tree Search (MCTS)**
    - Simulate multiple random playouts for each move and choose the one with the best average outcome.
4. **Opponent Modeling**
    - Track and predict opponent strategies or tendencies for better counter-play.

---

6. Conclusion

This project successfully demonstrates how AI can use simulation and evaluation to act strategically in a turn-based adversarial game. Although simplified, it lays the groundwork for more advanced AI systems using techniques from classical game AI such as minimax, state evaluation, and heuristic search. The current design is modular and extensible, making it ideal for academic exploration or future expansion into a full game AI system.