Bot Pokemon Showdown

GABRIEL FAZENDA

2016/2

Algoritmo utilizado

- Para o projeto, foi utilizada a técnica MiniMax para definir qual a ação do bot em cada rodada.
- Foi feita somente a análise dos possíveis movimentos do próprio bot para a próxima rodada (depth = 1), visto que para calcular movimentos das próximas rodadas tornaria a árvore muito extensa.

Implementação

- A função "decide" baseia-se inteiramente no resultado devolvido pela função Evaluator.getNextMove(state), que retorna um objeto que contém o "tipo" do movimento que deverá ser feito na rodada atual e um nome (do Pokémon em caso de switch e do Movimento em caso de ataque);
- Dentro do "Evaluator" o código é o seguinte:

```
getNextMove(state){
  var theWinner = undefined;
  myOptions = this.getMyOptions(state);
  console.log('heuristic bro');
  var firstNode = this.createNodes(state, 'move', myOptions);
  var theResult =  this._doMinimax(firstNode,1,true);
  var winningNodeMove = undefined;
  console.log(theResult);
  for(var i = 0; i < myMoveNodes.length;i++){
    if(myMoveNodes[i].result == theResult){
      winningNodeMove = myMoveNodes[i];
      break;
    }
  }
```

Onde "theWinner" é o vencedor entre o minimax do ataque e troca. A imagem ilustra o algoritmo de ataque, mas o de switch é bem semelhante.

Heurística

- A heurística implementada para calcular os resultados das possíveis escolhas é bem simples e baseia-se em três funções: uma para o ataque e outras duas para o caso de switch, onde uma trata as fraquezas de cada Pokémon e a outra as vantagens de cada movimento do mesmo em relação aos tipos do inimigo.
- Como se pode ver, a heurística valoriza muito mais os ataques do Pokémon pois trocar quase nunca é vantajoso, visto que permite que o inimigo utilize um golpe "de graça". Os valores foram modificados algumas vezes e foi constatado que os resultados obtidos pelos atuais são bem satisfatórios, possibilitando vencer contra o bot "Stabby" várias vezes (>70%)

```
evaluatePkmnMove(damage){
    var heuristic = 0.25;
    var result = (damage * heuristic);
    return result;
}

evaluatePkmnWeakness(value){
    var multiplier = -2;
    if(value == 2)
      multiplier *= 2;
    var result = (value * multiplier);
    return result;
}

evaluatePkmnStrength(value, movesCompared){
    var multiplier = 2 * (movesCompared/4);
    if(value == 2)
      multiplier *= 2;

    var result = (value * multiplier);
    return result;
}
```

Conclusões finais

- O problema foi abordado de duas maneiras:
  - Primeiramente, fiz o processo de criar uma finite state machine bem simples, baseada apenas em 'attack' e 'switch', mas as funções ficaram muito extensas e não eram muito satisfatórias.
  - Ao implementar o algoritmo de minimax (algoritmo final), o processo se tornou bem mais fácil e prático, retornando valores muito mais fáceis de se medir e controlar. Além disso, não foi mais preciso modificar funções em si, somente trabalhando com a heurística já foi possível constatar como ela altera o comportamento do bot.

  - Obs.: deu trabalho mas valeu a pena no final. Minimas ftw!