Oh, Hell! Card Playing AI Bot

Jack Amend and Cameron Zach
CIS5603 Final Project
Spring 2021

Outline

- Rules for the game *Oh, Hell!*
- Methodology
  - Use of state searching algorithms
- Web Application
- Experiments and Results

Oh, Hell! Gameplay

1. Players are dealt their hands.
2. Trump card revealed.
3. Players bid number of tricks they will take.
4. Play for all tricks.
5. Record points.
6. Shift dealer over and begin again.

Can an AI play this game sufficiently well?

Methodology

1. Implement game logic in Python
2. Implement two different AI algorithms
    a. MCTS and STS
3. Compare algorithms against each other
4. Create a web app to interact with the game
5. Test agents against human players

Simple Tree Search

- Uses a probabilistic method to evaluate game state
  - Each node on the tree represents one trick
  - The value of the node is the probability of winning the trick
  - The AI takes the path down the tree with the highest expected score (most tricks taken)
- Always bids maximum hand size

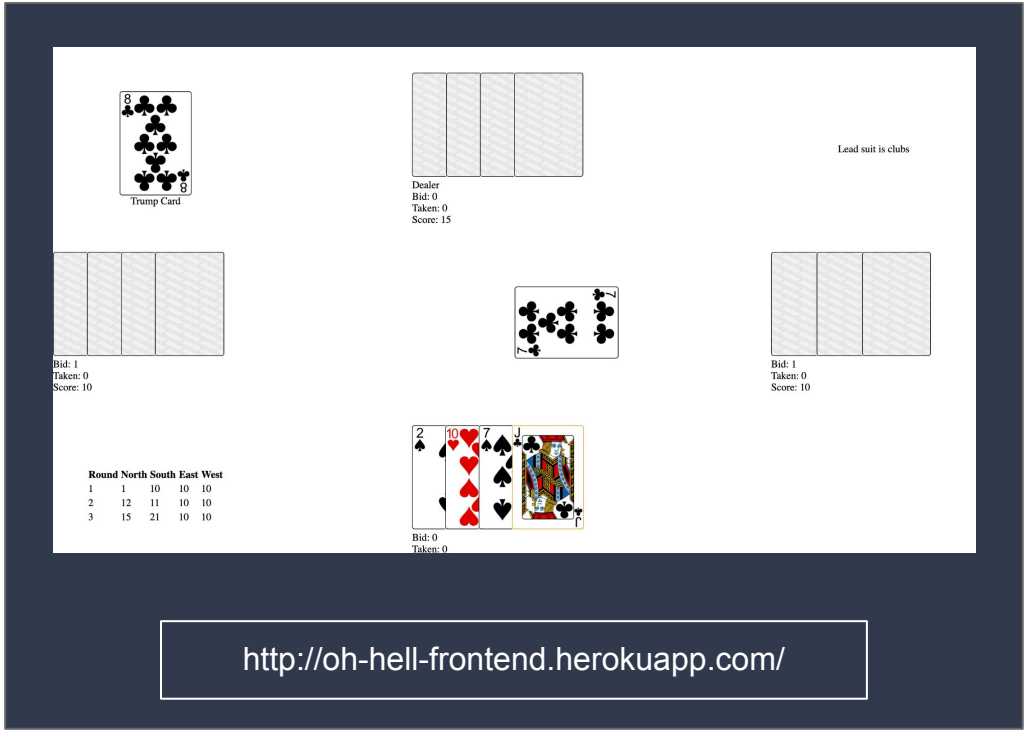
Monte Carlo Tree Search

4 stages:

- Selection
  - Chose best node
- Expansion
  - Add child state
- Simulation
  - Find end state
- Backpropagation
  - Update values

Web Application

- Frontend written with React
- Backend written with python and socket-io
- Allows configuration to decide which AI agent to use



http://oh-hell-frontend.herokuapp.com/

Experimental Results

Simple Tree Search

- Found diminishing results as depth of search increased
  - Simulating further hands gives too much weight to outcomes that can’t happen (cards already played, etc.)
- Easily beaten by the random player
  - Scored an average of 18 points in a 9-round game
  - Random player averaged 44 points

Monte Carlo Tree Search

- Better performance than the STS algorithm
- Similar decrease in performance as search time increased
- Strangely, the presence of the MCTS AI improved the performance of the Random player
    - MCTS averaged 46 points per game
    - Random averaged 42 points, compared to 13 points in an all-random control game
    - Likely because the random player could score 0 tricks more easily

MCTS vs STS

- MCTS completely outperformed STS
  - MCTS averaged 47 points per game
  - STS only averaged 17 points
- The diminishing returns of search depth held true in this test as well

Human experiments - Cameron

- The Random player performed best in manual testing
  - When playing against more competent players, it's easy for the random AI to successfully make a bid of 0
  - This was confirmed by manually playing the random strategy, easily beating all 3 AI players
- MCTS was able to beat me slightly in the first game

Human experiments - Jack

- Random and MCTS performed best
  - STS performed very poorly
  - Both nearly won during second trial
- MCTS only rewards for states that result in bids equaling tricks taken
  - Can begin in state where goal is impossible

Conclusions

1. Hidden information is much harder to account for than we initially thought
2. The ability for the random strategy to win so effectively is hard to account for with simple AI
3. These algorithms can easily be enhanced with smarter heuristics, but to fully understand the game we would want to use reinforcement learning

Thank you!