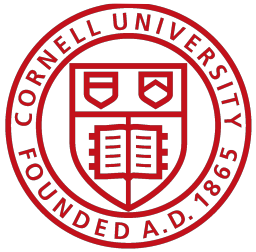


AI Prac - Battleship

Aditya Agashe, Susan Li

The Game Battleship

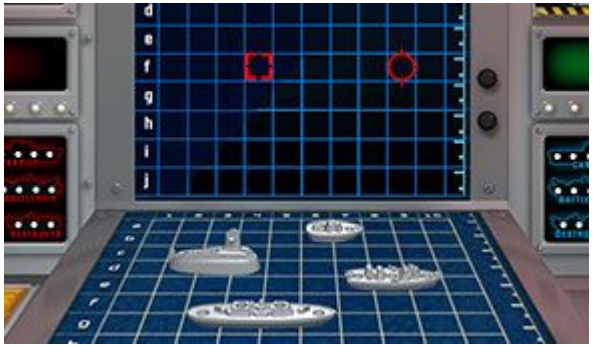
1. Four Ships

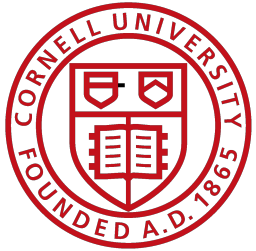
2. Take turns firing shots -- Announce if opponent has hit or miss

3. Sink your enemy's ships before they sink yours

Battleship in AI Terms

Battleship is a two-player, ***nondeterministic*** and ***partially observable*** game where the starting state is determined by each player. Since actions from each side are ***uncertain***, we must pursue a ***strategy*** to reach our goal state of having sunk all our opponents ships while still having a surviving ship.





System Architecture

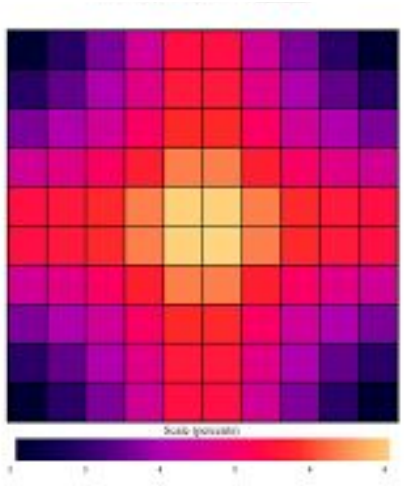
The game consists of 3 main components

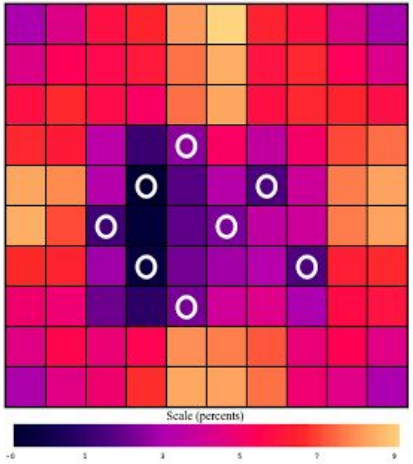
a. **The user** - The user interacts with the Game State module through the REPL text interface.
b. **Game state** - This handles all attributes of game play, from managing the two boards, ships of the user and AI, to getting moves from the AI and calculating hits and misses. It interacts with the AI using the AI interface.
c. **AI** - The AI handles making moves in response to user moves, the information of which is passed to it by the game state. It returns its moves to the game state through the AI interface.

Strategies of Increasing Complexity

1. The most basic battleship strategy is to **select a square in order** (square 1, square 2, square 3).
2. The next most naive approach is to select a **random square**.
3. A heuristic that improves on this method is that you **fire a shot at every nth square**, where n is the length of the largest ship still alive.
4. We can improve the above heuristic by implementing a **"hunt" function** (once a hit has been detected, the AI will fire up, down, left, right to kill that ship before randomly selecting another square).
5. The final improvement is to implement a game tree search: Instead of simply picking the next n-parity square that is available, we will make our selection based on a game tree with one level. We will choose the square that if we miss, eliminates the maximum number of locations that the ship could still be (attack the hottest point on the heat map).

Seeking a Ship

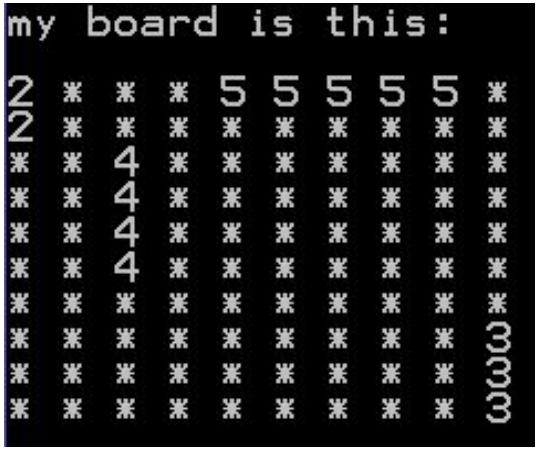


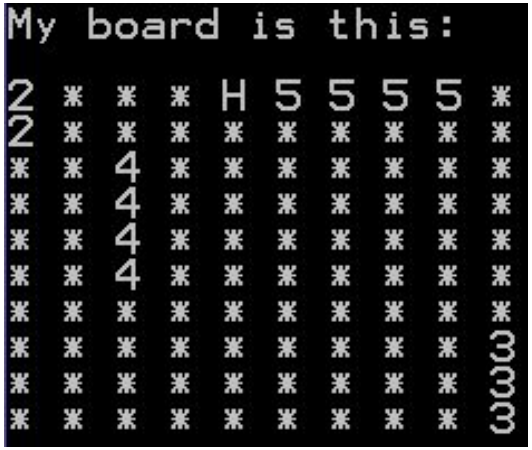


1) Build a probabilistic heat map
2) Attack the square with highest heat (hitting it would eliminate the greatest number of intervals that the largest ship could still be in)

Images: http://thevirtuosi.blogspot.com/2011/10/linear-theory-of-battleship.html

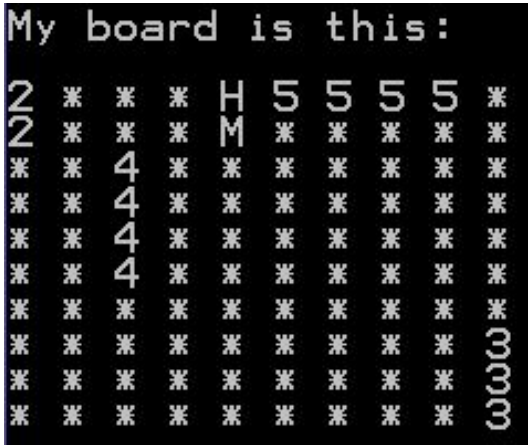
The Hunt - Find

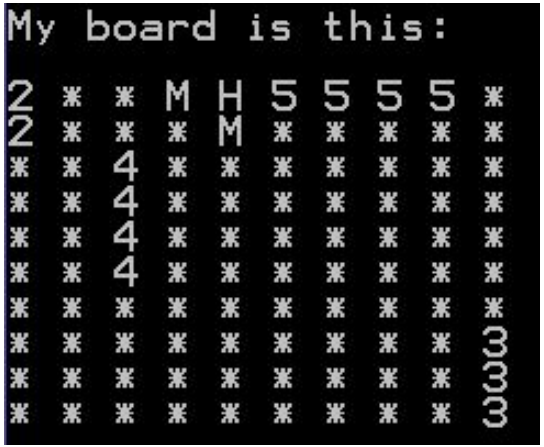




The AI has found a ship

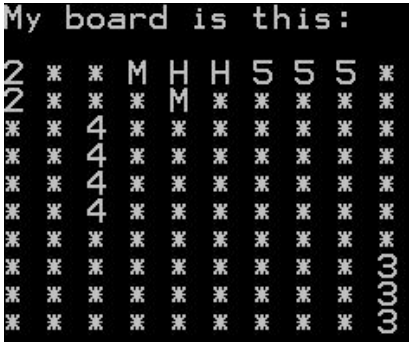
The Hunt - Search

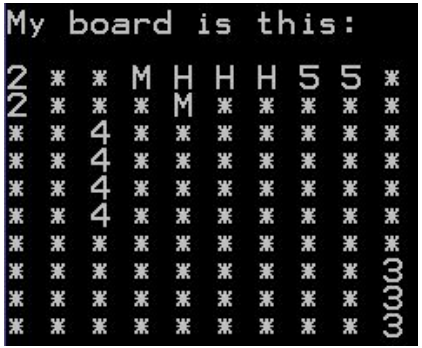


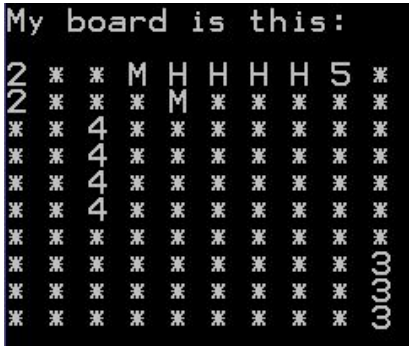


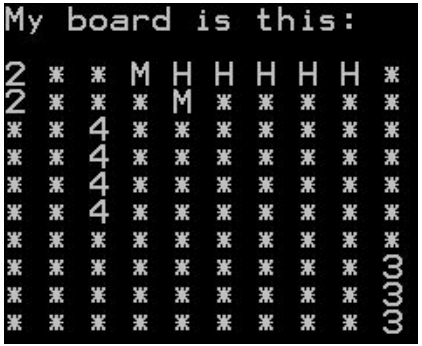
The AI searches the neighborhood of the first hit

The Hunt - Kill

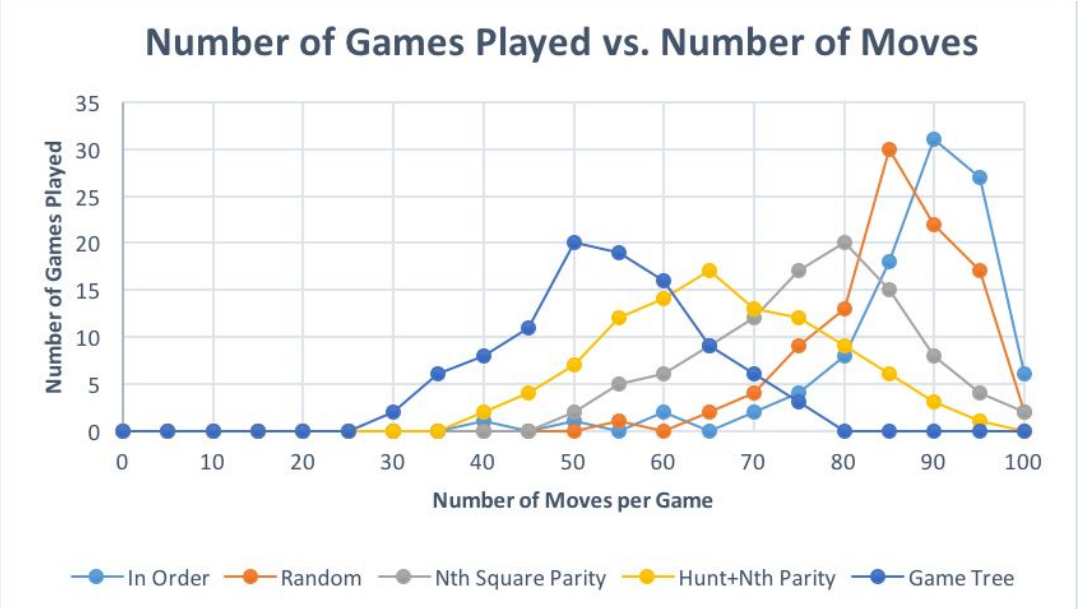




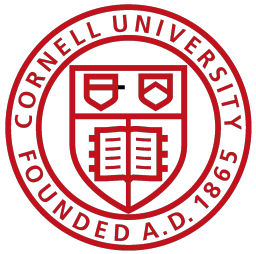




System Evaluation - Quantitative



1) Ran each iterations of AI against 200 randomly generated 10 x 10 battleship boards



Testing Strategy

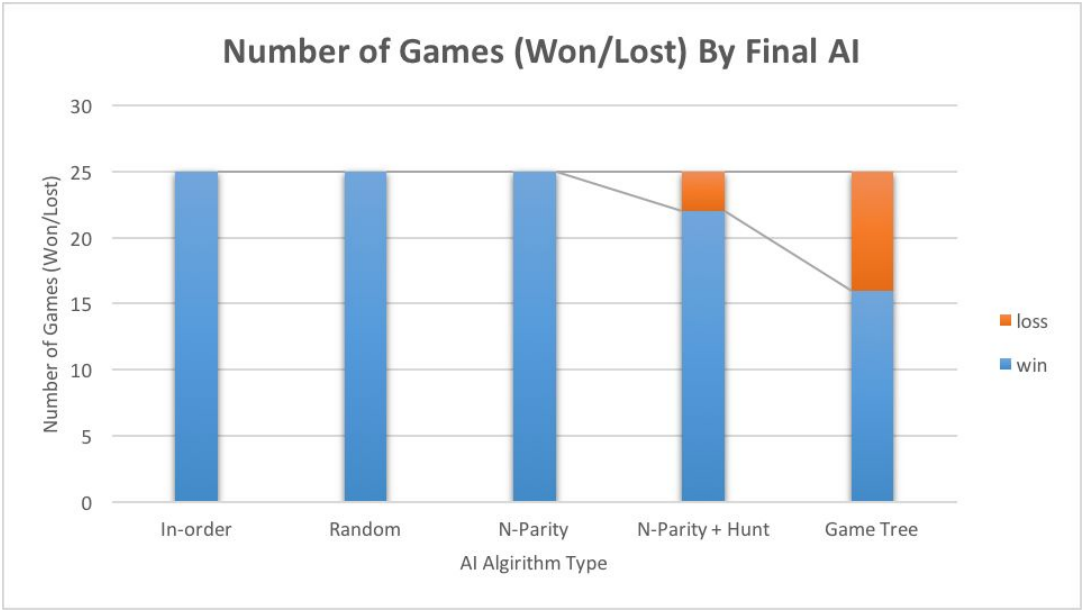
Qualitative Evaluation

- How many times each iteration of the AI could beat a smart human player
- We tested the effectiveness of our Battleship AI iterations by pitting them against a human opponent (Adi and Susan)

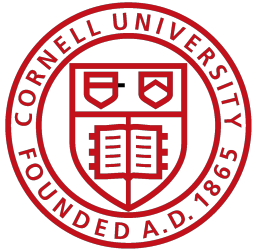
Quantitative Evaluation

- The chances of the AI using the random firing strategy to play a perfect game are roughly 1 out of 355,687,430,000,000
- We ran the AI against 100 randomly generated moves and counted the moves it took for the AI to hit all of the ships

System Evaluation - Qualitative



1) Tested Final AI against all iterations of the AI
2) Algorithm consistently won majority of the time against all 5 adversaries



Conclusion

- Most Valuable: Hunt Heuristic (once you've found a ship, kill it)
- Game tree was immensely valuable in finding a "first hit," much better than simply attacking the first interval where the ship could fit
- Thoughts:
  - Hunt Heuristic which is intuitive to humans was hard to implement
  - Heat Map, which humans can not intuit as accurately, really gives AI an edge

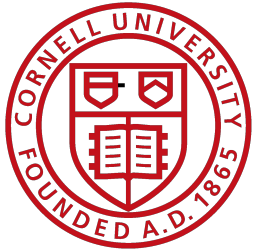
What next?

Deeper game tree?

How smart of a human can we beat?

Reinforced Learning?

The singularity?



QUESTIONS?