CSE 202 Project - Battleships - Implementation

Soham Pachpande (A59001664)
Raghav Prasad (A59012163)
Grishma Gurbani (A59019740)
Sreyas Ravichandran (A59019749)
Mayank Jain (A59019799)

March 17 2023

1 Introduction

We have implemented the algorithms described in our previous submission and varied the parameters to create multiple experiments. We tun multiple trials of these experiments and present the results of each of these experiments in this document.

Here is a description of the system we used to conduct our experiments:

1. **Programming language**: Python
2. **Processor**: Intel Xeon @2.2GHz
3. **Memory**: 12 GiB

2 Experiments

2.1 Parameters

1. **Board size**: 10×10
2. **Ships**: 4 ships with lengths $[2, 3, 3, 5]$
3. **Number of games**: 50
4. **Number of Monte Carlo samples**: 200

2.2 Algorithm 1: Random Selection

As a baseline, we played the game naively, randomly selecting the next move to play without applying any strategy.

2.3 Algorithm 2: BFS

This strategy places emphasis on sinking a ship as soon as we **HIT** it. This is done by performing a breadth-first search on the cells neighboring the **HIT** cell. In the event of a **MISS**, it goes back to the random strategy described in section 2.2.

2.4 Algorithm 3: Monte Carlo

Here, we generate a probability map of the board, the value in each cell denoting the likelihood of a ship lying in that cell. This probability map is generated using Monte Carlo sampling to generate ships placements for a fixed number of times (n).

3 Results

3.1 Expected Number of Moves to Win

The following table 1 gives the average number of moves required by each algorithm to win a game. The Monte Carlo Sampling method takes nearly half the number of moves compared to the baseline random strategy.

	Random Strategy	BFS	Monte Carlo Sampling
Expected Moves to Win	92.3	60.78	47.12

Table 1: Expected number of Moves to win a game

3.2 Cumulative Distribution for Moves to Win

The following plot 1 displays the performance of each of the three methods. On x axis is the number of moves to win and on y axis is a cumulative count of games won. As seen in figure 1, the Monte Carlo method far outperforms the BFS and the Random methods. The random algorithm takes 88, BFS takes 52, and Monte Carlo takes 41 moves to complete 25% of the games.

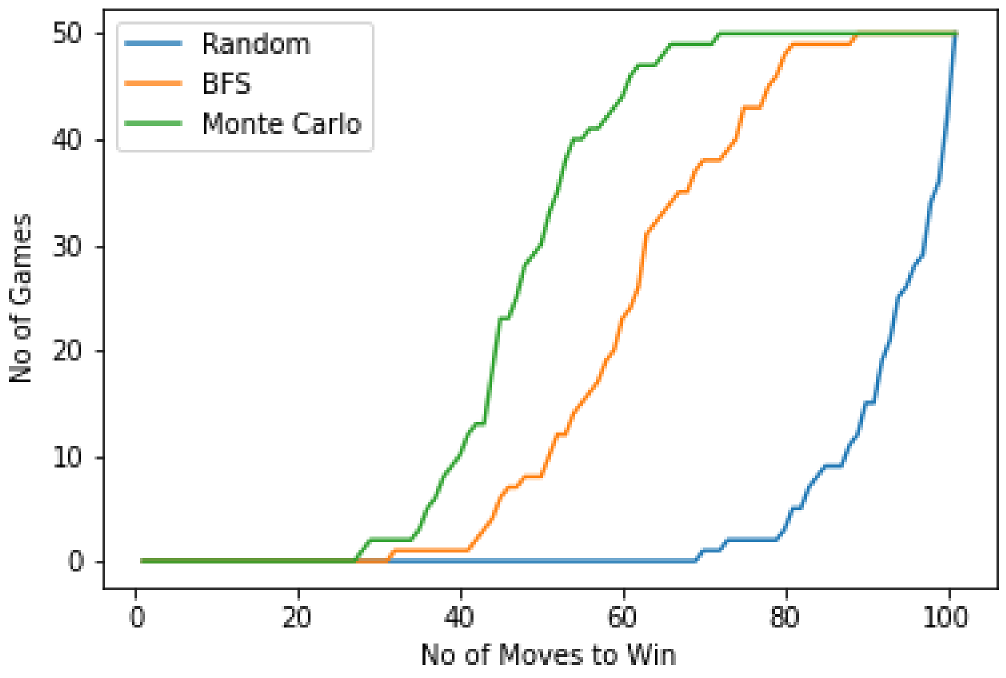

Figure 1: Cumulative Distribution for Moves to Win

4 Code

The code can be found at github.com/sohampachpande/battleships-solver