**Strategy for Minimizing Shots in Battleship**

Nicholas Winter and John Paul Depew

1. INTRODUCTION

Battleship is a game that involves a great degree of chance. Regardless of the random nature of the game, patterns can be found to improve performance and increase the odds of winning a match. Ship placement can be tailored based on expected hit areas of the other player. Ship targeting can be influenced by the number of unexplored cells surrounding each cell, clustering detection, and strafing patterns.

For ship placement, a simple learning algorithm can be used to decrease the chances of an AI player’s ship being hit. At the end of every game, the AI records the locations of the cells that its enemy targeted on its board, and saves it to a frequency table. The table contains the amount of times each cell has been hit. The AI uses this chart, and places ships in the cells with the minimum values. (THIS CAN BE FURTHER ENHANCED TO DO ROWS/COLUMNS WITH MINIMUM VALUE, CAPSING THIS SO I REMEMBER TO CHANGE IT IF IM ABLE TO GET IT WORKING). This strategy can result in the AI taking less hits over a period of time.

The main strategies used for targeting were heatmaps, strafing patterns, and clustering detection. The heatmaps was simple, raising and decreasing cells’ values depending on how many open spots were surrounding each cell. To detect clustering, the algorithm would revisit areas where a ship should have sunk but didn’t, the ship’s coordinates were not aligned, or the length was greater than the largest possible length.

One strong strategy for targeting in Battleship is to use a strafing pattern based on a number of subgrids of the battleship board [1]. Within these subgrids, values can be assigned to each cell based on the frequency of a specific ship appearing there. These values are used to create a strafing pattern for searching for ships, and the pattern is specific to each size of ship being looked for. Once completed, this pattern is guaranteed to find all the ships except the ship of length two, while only searching one third of the board. In addition, the algorithm and use a cleanup pattern to find the smaller ship while minimizing the number of cells to find.

The expected results of this project are that heatmaps, strafing, and clustering detection will each lead to overall less turns.

1. OTHER CONSIDERATIONS

Alpha Beta Pruning is commonly used in two player games and was considered for this project. However, this algorithm has certain limitations. The algorithm is dependent on knowing how one player’s actions will affect the other player’s actions, and how each player will respond to each other. In battleship, a player’s actions have absolutely no effect on the other player’s actions, and one cannot judge what the other player will do based on one’s own actions. Therefore, our project Alpha Beta Pruning did not seem like the ideal algorithm for this project.

1. MAIN RESULTS

Heatmaps

Figure 1. Graph displaying the number of total turns for two AI agents comparing heatmap use versus random selection for strafing and cleanup, only cleanup, only strafing, and neither.

Clustering detection

Figure 2. Graph displaying the number of total turns for two AI agents comparing clustering detection, no clustering detection, and skipping previous hits.

Strafing

Figure 3. Graph displaying the number of total turns for two AI agents with strafing and cleanup, only strafing, random with parity, and purely random shots. Data for each column is based on 1000 games.

1. CONCLUSION

While Battleship is frequently seen as a game of chance, strategies can be used to increase the odds of winning a game.

REFERENCES:

[1] E. Y. Rodin, J. Cowley, K. Huck, S. Payne, and D. Politte, “Developing a strategy for ‘battleship,’” *Mathematical and Computer Modelling*, vol. 10, no. 2, pp. 146–151, 1988.

<http://www.cores2.com/files/FinalResearchPaper.pdf> - Not cited yet, but still reading through and would like to use this