Developing An Optimized Hunt/Target Algorithm For The Game Of Battleship

Kim Scicluna
Malta College Of Arts, Science And Technology
Institute Of Information and Communication Technology Kordin Street, Rahal Gdid
Email: kim.scicluna.a106280@mcast.edu.mt

***Abstract*—This paper focuses on the board game known as Battleship. Different algorithms that can win a Battleship will be analysed in terms of efficiency. This will be achieved by using a modified Hunt/Target algorithm that is more efficient than the most commonly found algorithms. The outcome of this research will show if the algorithm developed better by comparing statistics of various simulations. The outcome is that the implemented algorithm fared better than the existing ones. Technologies used to develop the application are C-Sharp for coding and logic and TSQL and Excel for extracting reports tabulating data.**

***Keywords*—Battleship, Hunt/Target, Parity, Game Agent, IDE**

I. Introduction

A. Theme

In this research, the game of Battleship was used to implement an improved algorithm that solved the problem in as little shots as possible. For this research, three algorithms were developed, one taking random shots, another using a methodical approach and another that further improved on the latter.

B. Aim

The aim of this research was to see if the implemented changes on the improved algorithm made any significant difference to the amount of tries it takes to destroy all the ships on a board. This was achieved by recording every move of each game ran and comparing the results from the data gathered.

C. Research Question

The research statement of this paper is ”Developing An Optimized Hunt/Target Algorithm For The Game Of Battleship”. This means that the final algorithm implemented will try to guess the positions where adjacent ship cells might be by taking the orientation and the direction of the ship in consideration. This eliminates different adjacent cells that won't contain a part of a ship. This will lower the amount of target cells that a ship has to be in to as little as one cell.

D. Technical Solution

Three algorithms were implemented to aid this research. The first algorithm implemented shoots at random cells on the board until all ships were sunk. The second algorithm is the famous Hunt/Target algorithm where the algorithm changed shooting states when it hit a new ship on the board. The last algorithm is the improved Hunt/Target algorithm that utilizes orientation and direction of the hit ship to find the most guessing and finding the most probable cells that contains the adjacent part of the ship.

E. Motivation

Solving a problem methodically usually means that the steps taken to solve it can be used on multiple occasions. Although the algorithms mentioned in this paper are far from intelligent, it is a good way to start researching on solving problems using algorithms before exploring the complicated world of artificial intelligence. During my course, I was thought about the importance of optimization of an algorithm. The goal set in this paper is to see if the modified version of an already existing algorithm can perform better.

F. What To Expect

In the literature review, the history of the game will be documented. Since people play the game in different ways, I will describe the rules that the algorithms are programmed to follow. Also an overview of the different variations of the game will be given so offer ideas where the algorithm could be expanded so it would cater to different variations. Different ways of how people tackled the problem that this research focuses on will also be documented. This research paper will include various sources that helped me understand how many ways the game of Battleship can be approached. Tabulated information of gathered data will also be shown to give a more in depth view of how the improved algorithm compares to existing ones.

II. Literature Review

A. The History Of The Board Game

The game of Battleship is a turn based, two player guessing game. It started as a paper and pencil game that predates The Great War. Clifford Von Wickler was its inventor in the early 20th century however Von Wickler never patented the game. This led to various companies publishing their own version of the game. The first ever commercial version of Battleship was released under the name on Salvo in the year 1931. Later on, in the year of 1967, Milton Bradley, which is a board game company in the United States, patented and published a commercial, plastic board game version of Battleship.

B. Rules Of The Game

Battleship has a set of simple rules that should be followed. According to the 1959 patent, the game is usually played on a ten by ten (10x10) grid. There are two stages of the game, the ship placement stage and the shooting stage. When the first player calls out coordinates for his first shot, the opponent needs to say if that shot was a hit, miss or destroyed. The game is won when all the opponents ships are sunk. (Meyer, 1959)

C. Variations Of The Game

As stated in 'The History Of The Board Game', Battleship has a lot of variations. Variations include different grid sizes, different lengths and amounts of ship on the board. However, one of the most famous variation is the Salvos variation, where each turn the player takes multiple shots, however the player wont know if they hit or miss, until the turn is over.

D. Rules Used By The Algorithm

The rules used by the implemented algorithms are the following:

1) Played on a ten by ten grid
2) Only one shot per turn
3) The algorithm is only notified if the shot is missed, hit or destroy
4) Ships must have at least one empty cell between them

E. Randomly Shooting Algorithm

The first and easiest algorithm is to just shoot randomly. As it is stated in the paper Battleship: A Hit or Miss Affair , this can be used as a benchmark of a non-optimal way to play the game. This means that its results can be used as a control to compare other algorithms efficiency. This algorithm is not efficient at all as its based on pure luck. (Compton, Stanzione and Liu, 2014)

F. Hunt/Target Algorithm

The Hunt/Target algorithm is an optimized version of the random algorithm. This algorithm has two states, hunting and targeting. In the hunting stage, random coordinates are shot each turn. When the first hit is registered, the algorithm goes into a targeting stage. This stage is a more complicated as it calculates possible adjacent targets around the initial hit cell. When the targeted ship is destroyed, the algorithm goes back to the hunting stage. This drastically reduces the amount of shots when compared with the random algorithm. (Compton, Stanzione and Liu, 2014)

G. Adaptive Shooting And Targeting Algorithm

An adaptive shooting and targeting algorithms improves the more times it plays. This is done by generating heat maps of possible ship placements. It is a heavily modified version of the Hunt/Target algorithm, focusing on optimizing the hunting part of the algorithm. In the paper An Artificially Intelligent Battleship Player Utilizing Adaptive Firing and Placement Strategies, it is stated that Pre-training the solution is important since an initial "random" solution might perform too poorly, giving few hits on the opponents ships and little data for training the targeting algorithm Pre-training the solution is important since an initial "random" solution might perform too poorly, giving few hits on the opponents ships and little data for training the targeting algorithm. This means that the algorithm learns and adapts depending on the games it plays. At first it will perform poorly, however as the player plays more game against the AI, the AI will become better. Player profiles would ideally be saved so that there will be different profiles for different players.

III. Research Methodology

A. Researching The Rules Of The Game

The first step of this research was to understand the rules of the game of Battleship. Since there are many variations of the rules, the chosen rules were the most commonly used. Although these are not the official use of the patented game, the developed algorithms have been developed in a way that they can be modified to different set of rules.

B. The Chosen Rules

The chosen rules were the following:

1) Played on a ten by ten (10x10) grid, having 100 cells in total.
2) Only one shot can be taken in each turn.
3) The only data that the algorithm can use are:
    a) Shot Missed
    b) Shot Hit
    c) Ship Destroyed
    d) Amount Of Enemy Ships Deployed
    e) Type/length Of Enemy Ships Deployed
4) Ships can only be deployed horizontally or vertically.
5) Ships cannot be deployed adjacent to each other. A space of at least 1 cell must be present.

C. Technology Used

The technology used for the development of the algorithm was:

1) C-Sharp, using Visual Studio as an IDE. C-Sharp was used because of its wide variety of available libraries that made developing the application easier. These libraries include but are not limited to:
    a) Framework Entity to connect to a database
    b) Z Entity Framework Extensions to have an optimized way to insert bulk data into the database
2) TSQL, using Microsoft SQL Server as an IDE. This was used to create a database to store the data of the algorithms. This data includes but is not limited to, number of shots per game per simulation, positions of shots etc.
3) Microsoft Office Excel was used to tabulate the data to extract information.

D. Technique Used

The research methodology used is quantitative research. The technique used is statistical analysis. This was decided because a large amount of data would be collected. This data would need to be analysed. This data contains every shot of every game the algorithm makes. This made is easy to extract and tabulate statistics to compare the implemented algorithms. This will be explained more in detail in the coming sub sections. This was done by querying the data, using SQL statements, exporting the reports to CSV files and then tabulating it on a spreadsheet software.

E. Dataset Used

The dataset used for the algorithm was the grid and ship placement. Since the prototype focused on completing the game, the ship placement had to be done manually. This was achieved by creating a text document and building a grid using the following legend:

1) # - Sea
2) 1 - Carrier (Length 5)
3) 2 - Battleship (Length 4)
4) 3 - Cruiser (Length 3)
5) 4 - Submarine (Length 2)
6) 5 - Destroyer (Length 2)

A parser was developed to convert the dataset into objects. This step is done in the initialization part of the algorithm.

```
333#444###
##########
######1###
######1###
######1###
######1###
######1###
##2222####
##########
```

Fig. 1: Dataset Used

This raw text will be parsed into a board object and five ship objects.

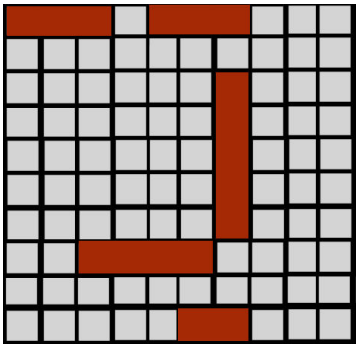


Fig. 2: Graphical Representation Of The Dataset

F. The Random Algorithm

The first algorithm implemented was the Random Algorithm. This implemented algorithm works by only shooting a random cells, irrespective if it was a hit or a miss. This algorithm was used as a benchmark and is not an optimal way of efficiently winning the game.

G. The Hunt/Target Algorithm

The second algorithm implemented is a common algorithm for the Battleship game. It's called the Hunt/Target Algorithm. The Algorithm has two states, hunting and targeting. The algorithm starts in the hunting stage, where it shoots randomly until its first hit. When the first hit is recorded, the algorithm goes into the targeting stage. In this stage, all adjacent cells of the are added to a list as possible targets. A random target in this list is chosen and shot. This is repeated until the ship is destroyed. The problem with this algorithm is that it doesn't take in consideration the orientation and direction it should target. These two points were taken in consideration when it came to implement the Improved Hunt/Target Algorithm.

H. The Improved Hunt/Target Algorithm

The improved hunt target algorithm woks the same as the Hunt/Target Algorithm but with a few significant modifications that made it more efficient. The changes are the following

1) Parity system for the hunting stage
2) Finds the orientation of the ship
3) Finds the direction it should target

The parity system only works if the shortest ship has a length of two or more. This is because it only targets half of the cells, more specifically it targets only those cells who's x and y coordinates are either both even or both odd, leading to the following hunting pattern.

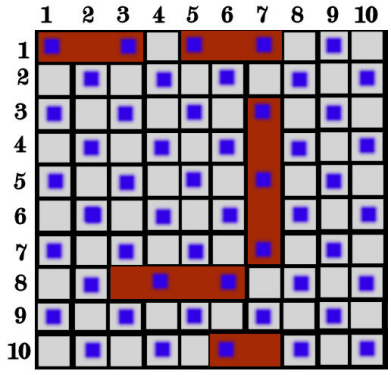
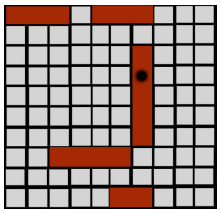


Fig. 3: Available Cells In Hunting Stage Shown In Blue

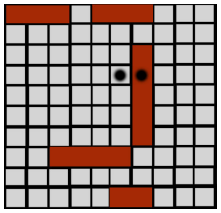
Since every ship is longer than 1 cell, every ship should have a targeted cell.

The second improvement is the orientation. Since no ships can be adjacent to each other, a ship either has empty neighboring cells or other cells belonging to the same ship. This means that on the second hit, the algorithm can determine the orientation of the ship, being horizontal or vertical. This

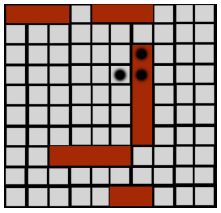
process is shown in Figure 4. This is an example of how the algorithm finds the orientation.



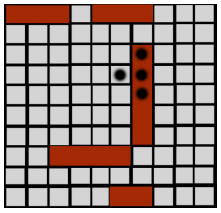
(a) N=1 First hit that triggers the targeting state



(b) N=2 Second shot is a miss



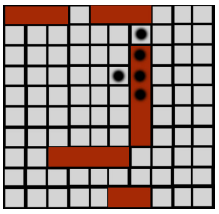
(c) N=3 Second hit shows that the orientation is vertical



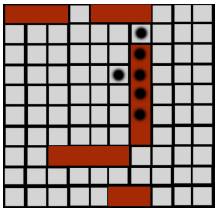
(d) N=4 Third hit is found by knowing the orientation

Fig. 4: Shots showing how the orientation is found

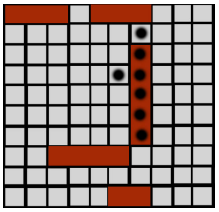
The next improvement done is finding the direction that it the targets should be picked. As shown in Figure 7, at that point, the algorithm only knows that it should choose targets that are vertical. The direction is found after the orientation is found and after the first miss. This technique is shown in the Figure 5 that are a continuation to Figure 4.



(a) N=5 First hit that triggers the targeting state



(b) N=6 Second shot is a miss



(c) N=7 Second hit shows that the orientation is vertical

Fig. 5: Shots showing how the direction is found

IV. Findings And Results

A simulation of 5000 games using each algorithm were executed. Each shot was recorded in a database and queried for analysis.

A. Prototype

The prototype algorithms worked as expected. However, the algorithms were not tested well for different grid sizes. There were other ideas for different algorithms that could have been more efficient, however there was no time to implement them in the prototype developed. Initially it was planned to record the average time it took to take a shot but this idea was not implemented in the version of the prototype.

B. Data

All the data was queried so that the number of all the shots calculated. The results gathered were the following.

Simulation	No. Of Shots In 5000 Games
Random Algorithm	476763
Hunt/Target Algorithm	331255
Improved Hunt/Target Algorithm	249210

TABLE I: Number Of Shots In 5000 Games Per Algorithm

The average number of shots of each game per simulation was also calculated. The results gathered were the following.

When this is calculated as a percentage, the Hunt/Target is 30.53% more efficient than the random algorithm. The

Simulation	Average No. Of Shots Per Game
Random Algorithm	95
Hunt/Target Algorithm	66
Improved Hunt/Target Algorithm	49

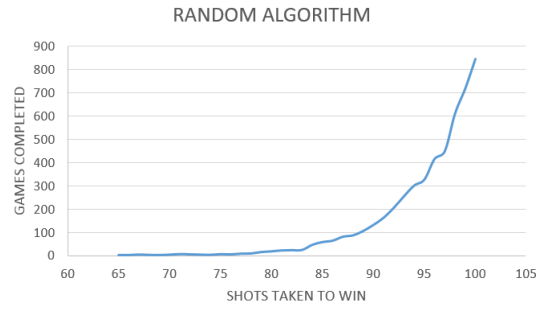
TABLE II: Average Number Of Shots Per Simulation

Algorithm	Least Shots	Most Shots	Range
Random Algorithm	65	100	35
Hunt/Target Algorithm	30	99	69
Improved Hunt/Target Algorithm	24	66	42

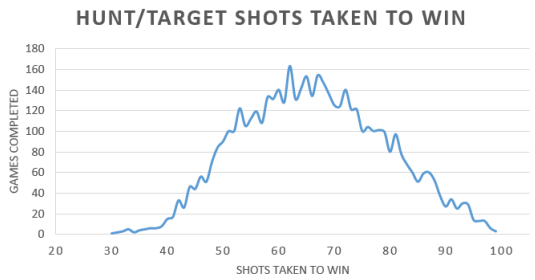
TABLE III: Least Shots, Most Shots and Range

improved hunt/target algorithm is 48.42% more efficient than the random algorithm. This makes the improved version 25.76% more efficient than the hunt/target algorithm.

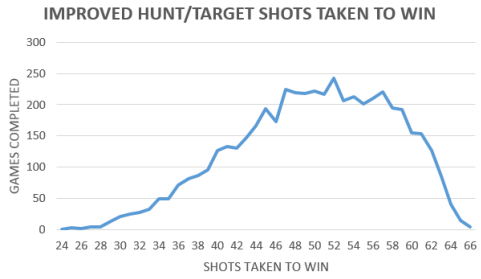
The following graphs in Figure 6 shows the amount of games completed (y-axis) with an amount of shots taken (x-axis) of the different algorithms.



(a) Random Algorithm



(b) Hunt/Target Algorithm



(c) Improved Hunt/Target Algorithm

Fig. 6: Graphs showing the performance of each algorithm

From these graphs, more information was gathered and tabulated, mainly the range of shots, the least number of shots it took to win at least one game, and the most number of shots it took to win at least one game.

C. Project

This project gave me an insight of how to structure a dissertation. It also helped me to learn and understand new techniques in programming. The only con I had with this project is the criteria of the internship. The criteria set for this project about the internship didn't really match up to the amount of effort that had to be put in. I ended up using more than two references for the literature review anyway, and if I wanted to, I could have used more.

D. Discussion Of Results

From the results gathered we can see that the improved algorithm is 48.42% more efficient than the benchmark random algorithm. This shows that simply finding the orientation and direction of the ship can have a big effect on performance since it was 25.76% more efficient than the normal Hunt/Target algorithm as shown in Table II.

Table III shows more in depth data of the performance. The random algorithm had the least range but the highest amount of shots per game. This shows that it is efficient at all. The Hunt/Target algorithm has a far wider range, however it has a small least number of shots taken to win a game. Even though it was better, the results were still inconsistent as there were a significant amount of games that took more than 70 shots to win a game. Compared to its improved counterpart, this shows that it still performs poorly. The improved algorithm had a range of 42 and it had the lowest least and most shots taken to win at least one game. With a range of 42 and a minimum and maximum of 24 and 66 respectively, this shows that it was more consistent and more efficient. This leads to the conclusion that an optimized algorithm was successfully developed. The random algorithm was a good choice to develop and use as a benchmark algorithm as stated in Section II.

V. Conclusion

The project's goal was achieved. The algorithm was improved appropriately. Enough data was gathered for analysis, however there was more room for analyzing, from the average amount of shots it took to get the first hit to the average amount of shots it took to destroy a ship. Further research could be done to develop a new algorithm and that could use a neural network or even probability using heat maps. This would need a dataset that would make the algorithm learn techniques. The existing algorithms could be modified to work on different sizes of grids.

References

[1] Compton, C., Stanzione, N. and Liu, J. (2014). Battleship: A Hit or Miss Affair. University of Massachusetts.

[2] Meyer, R. (1959). Battleship game. US 2898108 A.

[3] Bridon, J., Correll, Z., Dubler, C. and Gotsch, Z. (n.d.). An Artificially Intelligent Battleship Player Utilizing Adaptive Firing and Placement Strategies. The Pennsylvania State University.