Report

UAlg
UNIVERSIDADE DO ALGARVE

Battleship

Metaheuristic - Solve Problem

Course: Cibersecurity

Teacher: Prf. Dtr. Antonio Lobo

Student: a78830 Helder Oliveira

Student: a78279 Luciano Neves

UALG 2022

Abstract

Battleship Problem (BP) is a mathematical game that involves guessing the position of ships on a checkered board. The objective is to find all ships in a minimum number of attempts using Metaheuristic (M) approach.

Search and optimization algorithms can be used to solve the Battleship problem.

These algorithms use heuristic techniques to guide the search.

Keywords: Battleship problem, heuristic techniques, mathematical gamesearch algorithm , algorithms..., Search and optimization algorithms ...

Contents

Acronyms

BP Battleship Problem. i, 1, 2

GA Genetic algorithms. 1

M Metaheuristic. i, 1–3

1. Introduction

The Battleship Problem (BP) is described on Wikipedia [4], as a mathematical game that consists of guessing the position of ships on a chessboard. The objective is to find all the ships in a minimum number of attempts.

Metaheuristic (M)'s resource for solving BP consists of using a search algorithm that tries to find a satisfactory, but not necessarily optimal, solution in a reasonable period of time. These algorithms usually use heuristic techniques to guide the search and increase the efficiency of the process.

As suggested by the professor of the discipline through the link [5], the group decided to accept the challenge and explore further the game that is considered an NP-Complete problem.

As referenced [5], NP-Complete problems are in NP, the set of all decision problems whose solutions can be verified in polynomial time. NP can be equivalently defined as the set of decision problems that can be solved in polynomial time on a nondeterministic Turing machine. A problem p in NP is NP-Complete if all other problems in NP can be transformed (or reduced) to p in polynomial time.

There are several strategies that can be used to solve the BP efficiently, such as depth-first search, breadth-first search, and A* search. Furthermore, it is possible to use Genetic algorithms (GA) or particle swarm optimization algorithms to find an approximate solution quickly.

In summary, the solution to the BP can be found using search and optimization algorithms that explore the game's characteristics and use heuristic techniques to guide the search and increase the efficiency of the process.

1.1 Problem Details

For this problem, we are playing on a 10x10 grid and there are five different ships. The ships have lengths of 5 4 3 3 and 2. This means that the total number of squares covered by the ship in a 100 square grid is 5 + 4 + 3 + 3 + 2 = 17. 100) x 100 = 17% , there would be an equal chance of hitting each box. This means that success is possible, but after examining the math behind the warships, we found that this is not the case due to the unpredictability of the placement of the ships on the board.

To solve this problem, it is possible to use M algorithms, which are optimization techniques that seek to find optimal solutions quickly and efficiently. Some examples of M that can be used to solve the BP include GA, ant colonization algorithms and particle swarm algorithms.

Some popular M that can be used to solve the Battleship game problem include:

- Genetic algorithms [2]: Genetic Algorithms GA were first introduced by [Holland

1975], these algorithms use techniques inspired by natural selection and reproduction to find optimal solutions. Tabu search algorithms: these algorithms use a list of ”forbidden” solutions (the ”tabu list”) to avoid falling into already explored solutions and find more interesting solutions. Ant colony algorithms: these algorithms use the ant colony metaphor to find optimal solutions, where ants ”explore” the search space and ”deposit” pheromones on the solutions they find interesting.

- Particle swarm-based optimization algorithms: resumed in Particle Swarm Optimization [6] site wikipedia, these algorithms is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formula over the particle’s position and velocity.

Bothe algorithms work iteratively, testing different solutions and adjusting them according to the results obtained. When using a M to solve the BP, it is important to define an evaluation function that allows evaluating the quality of a given solution and establishing stopping criteria for the algorithm, in order to avoid it running for a long time without finding a solution.

It is important to remember that there is no single “perfect” script that can be used to solve the warship problem using M. The choice of algorithm and approach to be used depends on the specific needs of each problem and the context in which it is being solved.

1.2 Math function

The BP is a classic example of a search problem, in which the goal is to find the location of hidden objects (in this case, battleships) on a grid. One way to solve this problem using search algorithms is to use a brute-force search, in which the algorithm checks every possible location on the grid until it finds the battleships. This is not a very efficient solution, however, because it has a time complexity of O(n^2), where n is the size of the grid.

Notation for some simple results. The shape of the ship is the finite lattice set S^2 and its number of points is n. The opponent gets the S -p tile by transforming the shape of the unknown vector $-p \in Z^2$. The vector $p \in Z^2$ is the ship’s position. We say that a hit at x is a hit if $x - p$ and a miss otherwise. Assuming (without loss of generality) that the first impact occurs at the origin x = (0 0), then we know that the position of ship p is a point of the form S (i.e. $p \in S$), but we don’t know the actual value to determine p at this point, we can perform membership tests on S on points of the form p x and our goal is to determine p using as few membership tests as possible (called errors).

1.3 Evaluation of the problem

The problem consists of recovering the unknown position p of S with a small number of shots. In the construction of the problem, we created several approaches that we will talk about one by one during the presentation of the obtained results.

1.4 Function fitness

As we can read in the article Fitness Scenario Analysis for M Performance Prediction [3], normalized metrics are proposed to quantify the performance of the algorithm on known problems to generate adequate training data. Performance metrics are tested using a standard particle swarm optimization algorithm and are investigated along with three existing fitness landscape measures. The fitness function was tested in several algorithm optimizations in search of the best result, as we can see later on.

1.5 Problem Steps

To solve the problem, we considered the development of an algorithm that started with the basics using a blind search for the game board, and then we improved to seek the best results. We also use two modes of position boats, fixed and random.

The points of algorithm construction are:

- Blind search - Uninformed random walk: At each step the algorithm moves from a solution s randomly to the next solution in the neighborhood of s.
- Blind search with neighborhood - Stochastic Local Search: Local search algorithms move from solution to solution in the space of candidate solutions (the search space) applying local changes, until a solution considered optimal is found or a time limit be elapsed.

1.5.1 Strategies adopted

We create patterns to improve scam performance and adopt some strategies to optimize the algorithm. Next the strategies there we adopted:

- Split the board 50/50 to increase the search probability;
- The second strategy adopted was a diagonal search;
- The third is begin the search from the center board;
- Fourth, limit the board search;
- Finally, probability density search.

All strategies were added to improve the algorithm's success rate, and we achieved very satisfactory results.

The 50/50 board division technique allows the algorithm to search, in this example of 100 squares, only 50, greatly reducing the search time and the number of missed shots.

With the introduction of the diagonal search technique, the hit rate increased considerably, as ships can only be placed vertically and horizontally, so the diagonal is a good strategy to intercept them.

Alemi (2011) [1] presented the long linear vessels theory which states that near the center of a 10×10 grid there is a greater concentration of incidents. He also said that the turns have fewer collisions. Using the technique of starting the search in the center of the board is risky, however, it is better than having the algorithm start randomly, because there is a high probability that the ships are placed in the center and with the diagonal search they quickly intercept.

Limiting the search space on the board, we assume that the ship with 4 squares has already been reached, so the algorithm will not search in spaces that have less than four squares available.

As described in probability theory in [7], a probability density function (PDF), or density of a continuous random variable, is a function whose value at any sample (or point) in the sample space (the set of values possible values taken by the random variable) can be interpreted as giving a relative probability that the value of the random variable is equal to that sample.
For example, if a ship has 5 houses, the search for the remaining houses is done in percentage until the chance of success is reduced.

2. Approach to the problem

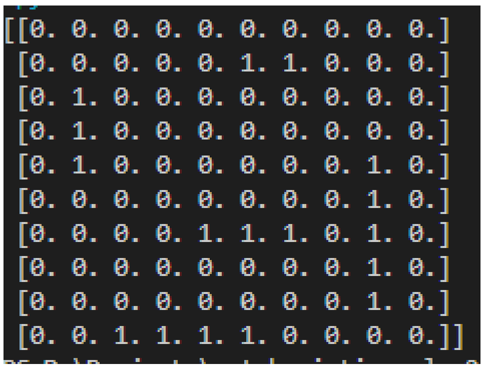
2.1 Phase One

2.1.1 First approach - fixed ships

In the first approach to this problem using a search algorithm, we use a brute force search where the algorithm runs through all possible positions on the grid until it finds a ship. This is not the most efficient solution, but it has O(n2) complexity and can be expensive if the grid search increases or the number of iterations increases as the grid size increases best success rate.

2.1.1.1 Result

As expected, this technique gives very poor results, however it was important to develop the algorithm to see in practice how the search is carried out in a random way.
Here are the results obtained:



```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1. 1. 1. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 1. 1. 1. 1. 0. 0. 0. 0.]]
```

Figure 1: Planning - Stage 0

First interaction blind search:

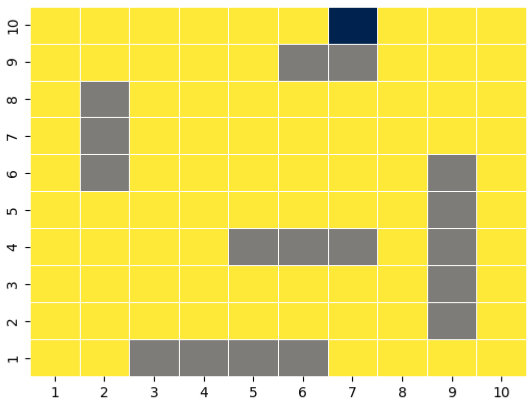


Figure 2: Blind search - Stage 1

In the first approach, as we can see, the results are 99 rounds, for the expected 17 correct shots.

2.1.1.2 Limitations and Challenges

It is a very high value, and the aim of the study is to significantly reduce the number of turns.

2.1.2 Second approach - fixed ships

In the second approach, we chose to add the concept of neighborhood search to the algorithm, and the objective is each time a boat is detected, it will go through the adjacent houses in an attempt to sink the boat completely.

2.1.2.1 Result

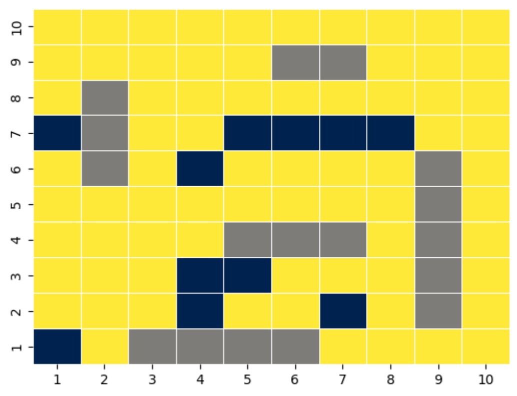


Figure 3: Blind search with neighborhood - Stage 2

In the second approach, as we can see, the results are 89 rounds, for the expected 17 correct shots.

2.1.2.2 Limitations and Challenges

In this approach, we achieved better results compared to the first one, because although the algorithm makes a blind search for all the houses from the moment it detects a ship, it explores all the surrounding points in search of more points of interest.
The identified problems are: if a ship with 5 squares, if the shot is on the 4th square and it advances to the 5th square and after the 5th, squares 1, 2, 3 are still to be hit.

2.2 Phase Two

2.2.1 Third approach - fixed ships

From the third approach, we remake the algorithm and implemented some changes, in order to explore better hypotheses of the problem. An interaction simulation option was added for n times chosen by the user, to observe the results obtained from an average of actions that number of rounds of several consecutive games.
We added a plot to help you understand the results obtained, which are automatically saved in a CSV file.
In this simulation we will use stationary ships and neighborhood search, but we will use 100 interactions in a row to validate what differences exist. The algorithm is also being started at the center of the matrix.
The previously identified strategies were also added to optimize the algorithm in order to improve the results obtained.

2.2.1.1 Result

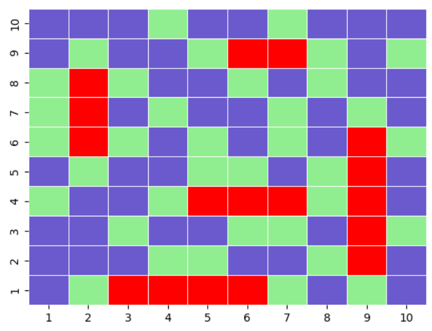


Figure 4: Improved neighborhood algorithm - Stage 3

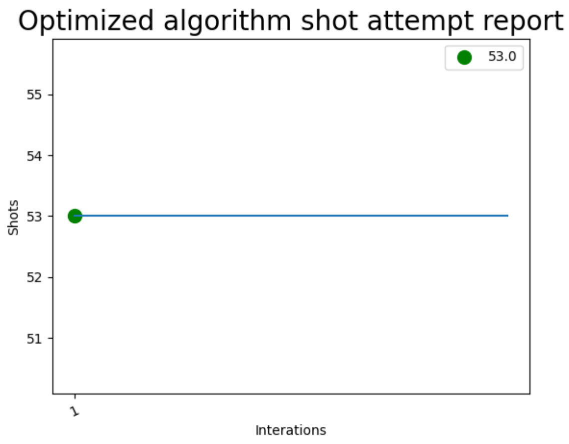


Figure 5: Plot results - Stage 2

In the third approach, the results are 53 turns, 36 shots missed and 17 correct shots with an average of 53%.

2.2.1.2 Limitations and Challenges

The results obtained were quite satisfactory, as we increased the hit rate by about 30% compared to the previous phase.
The idea of starting the algorithm in the center, however, was not a good ideia, because the fixed ships, the algorithm always starts at the same point and starts the search from there.
The results did not evolve and the number of simulations always maintained the same average that we can see in the graph.

2.2.2 Fourth approach - random ships

In this simulation, we performed two different tests. First test with one simulation in the algorithm. In the second test we ran a battery of simulations of 10, 100, 1000 turns to see the average success rate of the algorithm.

2.2.2.1 Test one

We have made two different simulations on first test, to see the results approximation.

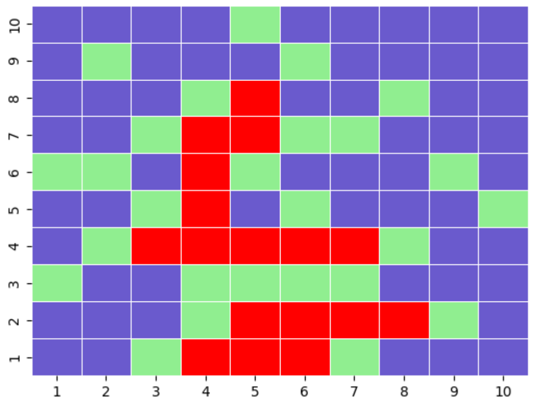


Figure 6: Random improved neighborhood algorithm - Stage 4 - test 1a

In the first test, the results are 43 turns, 26 shots missed and 17 correct shots with an average of 43%.

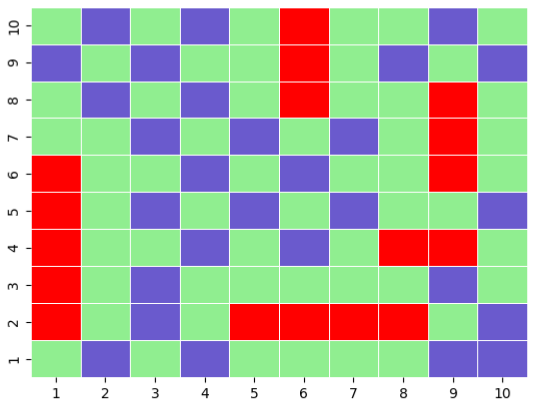


Figure 7: Random improved neighborhood algorithm - Stage 4 - test 1b

In the first test, the results are 72 turns, 55 shots missed and 17 correct shots with an average of 72%.
The results obtained from the two simulations are quite contradictory, they are quite dispersed results. We then decided to carry out a test with several simulations to validate the effectiveness of the algorithm.

2.2.2.2 Test two

In this test, as mentioned above, we are going to run several simulations in search of the best average score.

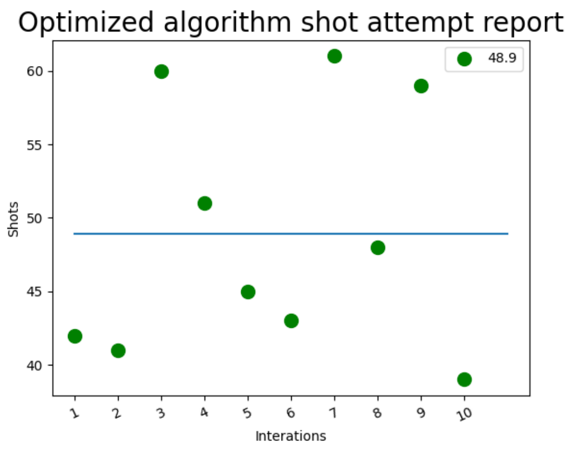


Figure 8: Random improved neighborhood algorithm - Stage 4 - teste 2a

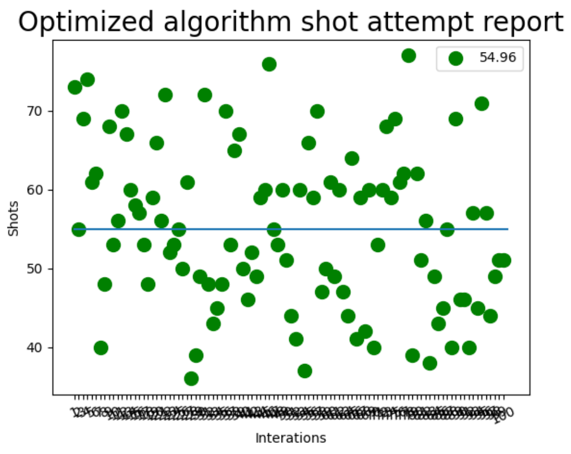


Figure 9: Random improved neighborhood algorithm - Stage 4 - teste 2b

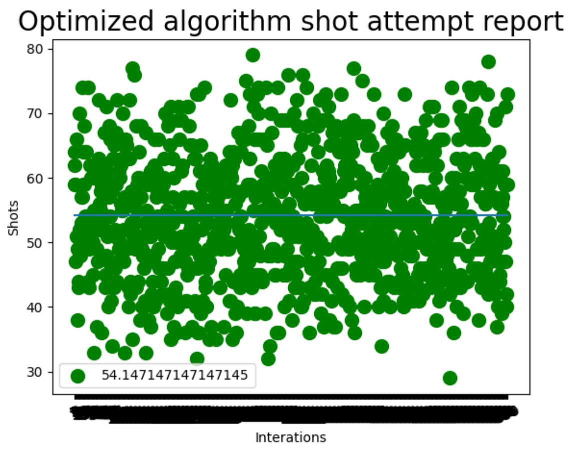


Figure 10: Random improved neighborhood algorithm - Stage 4 - teste 2c

The average of results obtained in the various simulations is around 50%, which leads us to want that the search for values, even with the optimized algorithm, does not show evolutionary results.
Only in the first simulation did we obtain satisfactory results, below 50%.

3. Discussion

After all the tests carried out, we can say that there was a significant improvement in the results obtained. The difference between fixed and random ship tests is also important to understand the best approach to take.
The problem is that it is a real challenge to achieve results below those obtained, even with all the strategies and methods adopted.
We think that among all the tests done, we could also have done a test on fixed boats but with random search, without starting from the center of the board.
The results we proposed with the candidate function of gradually improving the results obtained were obtained, improving the initial results by about 40%.
It is for this reason that the problem is considered an problem, any one of a class of computational problems for which no efficient solution algorithm has been found. The feeling that remains is one of satisfaction with the results obtained and with the strategies adopted to improve results.

4. Conclusion

We conclude that the results we proposed were obtained with the use of research and classes, which served as support. We found that there are several types of approach to the problem and that the adopted solutions helped to obtain better results than the initial ones.

Bibliography

[1] Alemi. The Linear Theory of Battleship. https://link.springer.com/chapter/10.1007/978-3-642-41888-4_4, 2011. [Online; accessed 27-12-2022].

[2] University of Skövde. What is the genetic algorithm?, 2022. URL https://www.mathworks.com/help/gads/what-is-the-genetic-algorithm.html. [Online; accessed 02-01-2023].

[3] SpringerLink. Fitness Landscape Analysis for Metaheuristic Performance Prediction. https://link.springer.com/chapter/10.1007/978-3-642-41888-4_4, 2023. [Online; accessed 06-01-2023].

[4] Contributors to Wikimedia projects. Battleship Game. https://en.wikipedia.org/wiki/Battleship_(puzzle), 2022. [Online; accessed 01-11-2022].

[5] Contributors to Wikimedia projects. List of NP Complete Problems. https://en.wikipedia.org/wiki/List_of_NP-complete_problems, 2022. [Online; accessed 23-12-2022].

[6] Contributors to Wikimedia projects. Particle swarm optimization. https://en.wikipedia.org/wiki/Particle_swarm_optimization, 2022. [Online; accessed 01-12-2022].

[7] Contributors to Wikimedia projects. Probability density function. https://link.springer.com/chapter/10.1007/978-3-642-41888-4_4, 2023. [Online; accessed 06-01-2023].