Queen's UNIVERSITY

GRP 21: Battleship - Best Hit Detector

Donovan Bisson (19dcb2)

Matthew Woo (20mw6)

Justin Woo (20jw8)

Marc Bucovy (19meb12)

Course Modelling Project

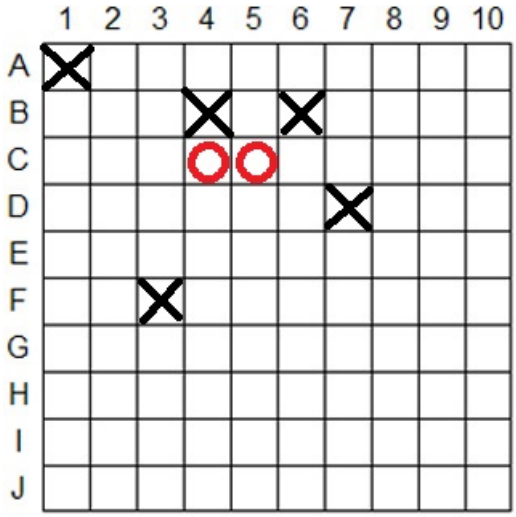
CISC/CMPE 204

Logic for Computing Science

October 29, 2021

Abstract

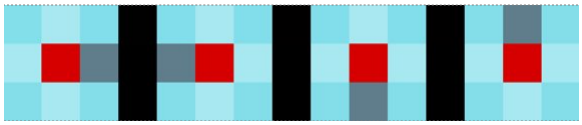
Battleship is a two-player board game about sinking all the opposing player's ships by taking turns with them attacking certain grid coordinates on each other's board. The picture below depicts the grid coordinates that one of the players have attacked. The black X marks attacks that did not hit any ship; a miss. Red circles indicate an attack has hit a ship. Blank spaces are areas of the grid that are yet to have been attacked



An example of a typical battleship game

This project intends to show the likelihood of a ship being hit by attacking a specified tile and an arbitrary board state. The standard Battleship pieces will be used (2 x 1 Destroyer, 3 x 1 Submarine, 3 x 1 Cruiser, 4 x 1 Battleship, 5 x 1 Carrier), as well as other standard Battleship rules (10 x 10 grid, no overlapping ships, no diagonals). The only exception to this will be that ships will not be named on hit; however, they will still be named when sunk.

This is done by examining every possible permutation for every ship that have a part of the ship presiding on the checked tile. Every tile is checked this way, and the tile with the most permutations has the greatest chance of a ship presiding on that checked tile. Thus that tile or tiles will be considered the best places to attack to increase the likelihood of a hit.



The diagram above depicts how the permutations will be checked at a single tile. By following the battleship rules for the ship lengths stated

above, there would be a total of 4 + 6 + 6 + 8 + 10 = 34 permutations that a single tile can be checked for. By analyzing which tiles can cover the most of these permutations, it determines the best tiles to hit.

Propositions

D, S, Cr, B, Ca - Each evaluates to true when the respective vessel (Destroyer, Submarine, Cruiser, Battleship, Carrier) that the proposition represents is active on the board. By being marked as false, the permutations of the ship will not be used during the hit check process.

$D_{i,j,d}, S_{i,j,s}, Cr_{i,j,cr}, B_{i,j,b}, Ca_{i,j,ca}$ - These are the permutations of each ship type relative to (i,j). The third value represents the possible orientations of the ship.

$X_{i,j}$ Evaluates to true when the point (i,j) is a known, unresolved hit. Unresolved means

$O_{i,j}$ Evaluates to true when the point (i,j) is known, but is not an unresolved hit (Miss, Out of Bounds, confirmed sink). This allows the identification of ship permutations that are not possible based on these tiles where something other than an active ship part is located

Constraints

Dimensions of the board are defined as $\{(i,j)|1 \geq i \geq m, 1 \geq j \geq n\}$ where m and n are the width of the board.

$X_{i,j} \rightarrow P_D \vee P_S \vee P_{Cr} \vee P_B \vee P_{Ca}$.
P is shorthand for all permutations. ie. $P_D = (D(i,j,1) \vee D(i,j,2) \vee D(i,j,3) \vee D(i,j,4))$. Unresolved hit must allow a permutation based on any of the 5 ships at the coordinate of (i,j)

For any $D_{i,j,1}, D_{i,j,1} \rightarrow D \wedge X_{i,j} \wedge X_{i+1,j}$

For any (i,j), $\neg(X_{i,j} \wedge O_{i,j})$. Means that a tile cannot be both a unresolved hit and Not an unresolved hit

Model Exploration

We first intended to have separate classes to represent the hits and misses. To simplify this we made the Hit and Miss propositions a subclass of the coordinate class, which holds the coordinate location on the board.

We also wanted to have one class to model all permutations over all ship types. Instead of this, there is a distinct class for each ship type, within which each permutation for this specific type is to be held. In order to not introduce too much complexity, each ship type is a subclass of the ship class.

For each ship we created a function to deal with each hit, or assumed hit. These functions should rule out permutations, in relation to the coordinate being considered. By creating a function that works on each coordinate, we can further break down into a series of cases. Relative to the coordinate we search for instances where a permutation results in both a Hit and Miss proposition overlapping.

To further simplify the construction of the constraints, we decided to put these functions specific to the ships, into the classes specific to the ships. This way we will not have to repeatedly call the functions for each point, and the constraints should be made at the same time as the propositions. In this way we can rely on the coordinate class to do a lot of the work for us.

Jape Proofs

Ideas for what we might want to prove in Jape:

If checking a tile along the edge of the board, then some of the permutations will not exist.

Disprove the validity of a permutation, by showing it falls on an OOB space.

Show that only that only a permutation of a destroyer can exist in the unresolved tiles

First-Order Extension

We could utilize the

Proposition(Predicate):

D(x), S(x), Cr(x), B(x), Ca(x) means x is a destroyer, submarine, cruiser,

battleship, carrier.
X(i), i is a unresolved hit
O(i), i is not a unresolved hit

Constraint:
$D_i = \{1, 2...m\}, D_j = \{1, 2...n\}$ where m and n are arbitrary integers
$\forall i \in D_i, \forall j \in D_j$, i and j act as coordinates for the board and must be a value in D_i and D_j