

Spencer Shimko  
Doug Ingalls  
Brian Williams  
Team name: Core-Dump  
Battleship 471  
Project Proposal

Team Core-Dump will make a learning agent for this project. In an effort to maximize our utility points and minimize the opponents utility points our agent will be capable of learning through several methods.

We will create a simulation gameboard given the dimensions from the board generator. This board structure will consist of a series of nodes. Each node on the board will contain: a ship name (or nil), damage to a ship (if we shot it), and a probability (possibly two) .

The ship name in the node will be initialized to nil and then change as we learn more about any possible ship in that node. The damage will track the percentage of ship that has been destroyed by our own shots and possibly a guess at the damage inflicted by the other player (if he shoots multiple times for instance).

The probabilities will represent the likelihood of a ship existing at the node. When the board is initialized each nodes probability will assume the default value of (total squares consumed by ships / total squares). This is the probability of a ship existing in any one square. This initial probability will also be adjusted based on simulations from the game generator. This aspect will be explained in further detail below. As the game progresses the probability will be adjusted to track the likelihood of a ship being found in that node. For instance, if we peek(5) and find a ship, the probability on that node and possibly the adjoining nodes will be increased accordingly. If our opponent peeks and then continues without shooting we will slightly lower the probability. Basically this will be a compilation of what we know about the board and what we have learned from our opponents moves. The second probability, if implemented, will track only our opponents moves and will be used mainly for blocking. This probability will show the likelihood of an intelligent agent firing at a given square and will indicate which squares we should block. These probabilities will perform better against smarter agents since they will learn from the other agents as well.

We will also attempt to confuse opponents agents that might be capable of learning by procrastinating shooting after peeking. If our peek indicates a ship at a given location we will place that location in a buffer. We will later pull items off this buffer randomly to pursue as possible ships. This should slightly “confuse” agents who only learn through tracking direct peeks followed by shots.

The game generator should be a great help in adjusting our initial probabilities described above. If we run a significant number of simulations we should be able to track the likelihood of a ship being placed on any one square. The more biased the generator is towards a certain square, the higher the initial probability for that node.

Our uninformed search algorithm will start at (0,0) and proceed in

increments of two throughout the board. So the second peek operation will be performed on node (0,3). We will iterate through the board using a two-space gap in between nodes. This will minimize the number of operations and maximize our coverage. If it becomes necessary to repeat the search we will start at (0,1) and continue in the same fashion. A simple diagram:

```

-----
| X | - | - | X | - | - | X |
| - | X | - | - | X | - | - |
| - | - | X | - | - | X | - |
| X | - | - | X | - | - | X |
| - | X | - | - | X | - | - |
| - | - | X | - | - | X | - |
| X | - | - | X | - | - | X |
| - | X | - | - | X | - | - |
| - | - | X | - | - | X | - |
| X | - | - | X | - | - | X |
-----

```

Large boards will be broken down into quadrants and each quadrant will be searched using the method described above.

Our strategy for choosing operations will be based on our probability board and on the utility cost of any of the possible operations given the current game state.