ARTIFICIAL INTELLIGENCE NANODEGREE

Adversarial Search Project Report

Author: Vraj Parikh

Search Strategy

Implemented a simple **Minimax Algorithm** with **Alpha-Beta pruning** and Iterative Deepening.

Heuristic / Evaluation Function

In this project, I have implemented 6 heuristics - one from the course and the other is a variation of the same.

Heuristic 0: Defensive Strategy

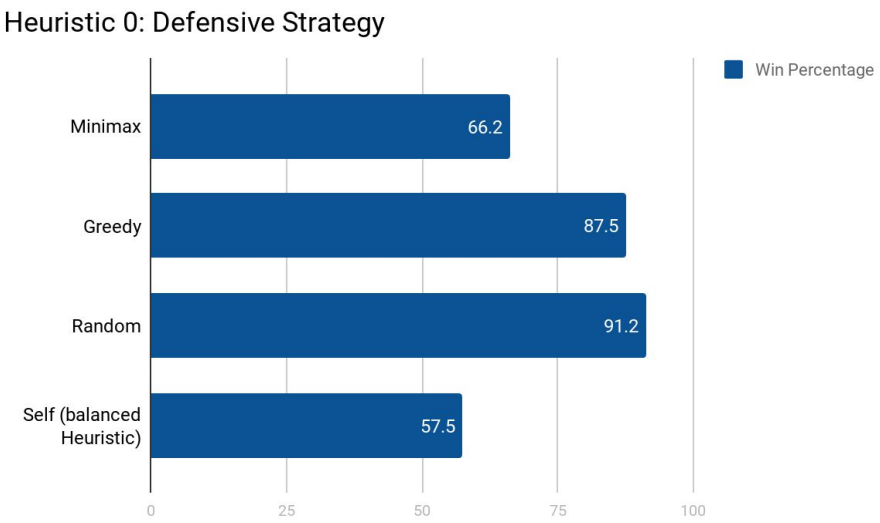
Maximizes the strategy that our player having more moves will always win. Here, we give more impact to the number of player moves available than the number of opponent moves. This leads to the agent playing more defensively.

The factor 2 can be changed and see which is most effective.

Formula

```
2 * number_of_player_moves - number_of_opponent_moves
```

Performance Comparison



Heuristic 1: Offensive Strategy

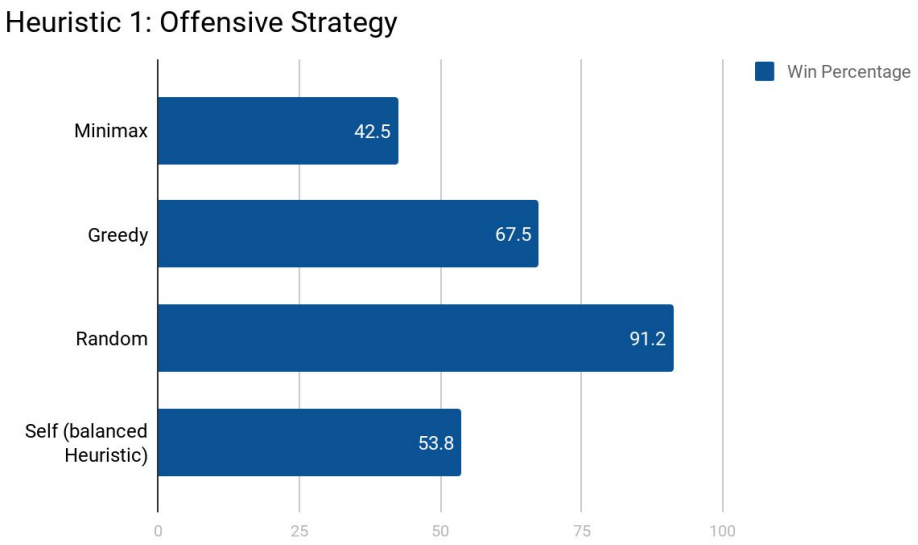
This is totally opposite to the defensive strategy. Here we give more weight to the opponent, hence letting our player play more aggressively to minimize the opponent winning.

The factor 2 can be changed and see which is most effective.

Formula

number_of_player_moves -2 * number_of_opponent_moves

Performance Comparison



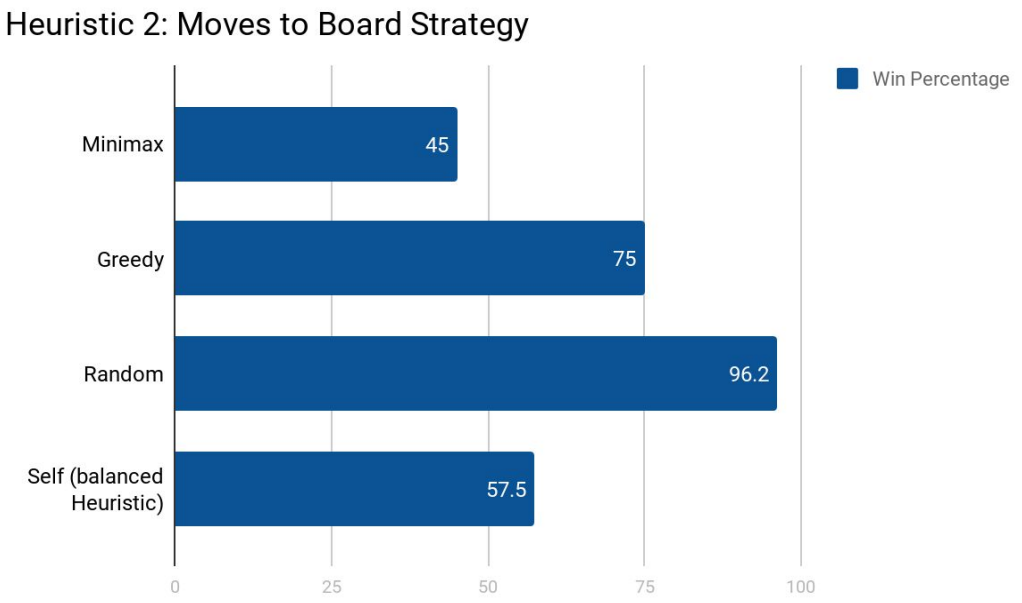
Heuristic 2: Moves to Board Strategy

In this heuristic, we give our agent sense of how much deep down we are in the search. We estimate this by finding the ratio of **(number of rounds / total blocks on the board)**.

Formula

2 * num_of_rounds / board_size * number_of_player_moves - number_of_opponent_moves

Performance Comparison



Heuristic 3: Defensive To Offensive Strategy

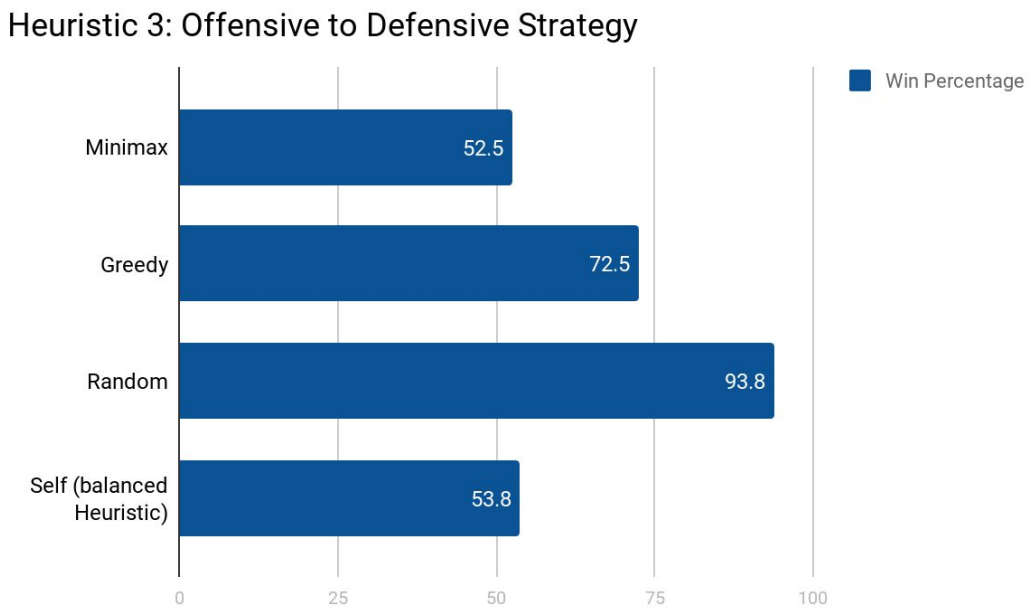
This approach again provides the agent with an idea on how much down the search the agent is. Here at the same time, we tell the agent to first be defensive and when the game is half way done, we ask the agent to get offensive.

Formula

If (rounds/board_size < 0.5) -> Defensive

else -> Offensive

Performance Comparison



Heuristic 4: Offensive To Defensive Strategy

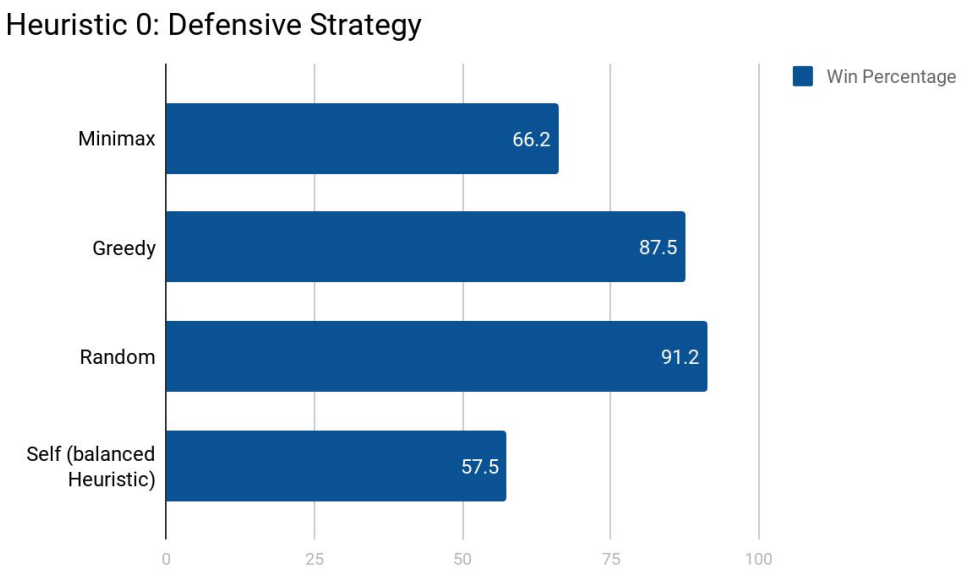
It is the inverse of the above approach. This approach again provides the agent with an idea on how much down the search the agent is. Here at the same time, we tell the agent to first be offensive and when the game is half way done, we ask the agent to get defensive.

Formula

If (rounds/board_size < 0.5) -> Offensive

else -> Defensive

Performance Comparison



Heuristic 5: Blank Space Strategy

This heuristic is a variation of the above Heuristic, which also takes into consideration the number of empty spaces.

Intuition: In a scenario where the game runs till all the block wears out, it can be considered that if there are an odd number of empty spaces left, then the current player will always win.

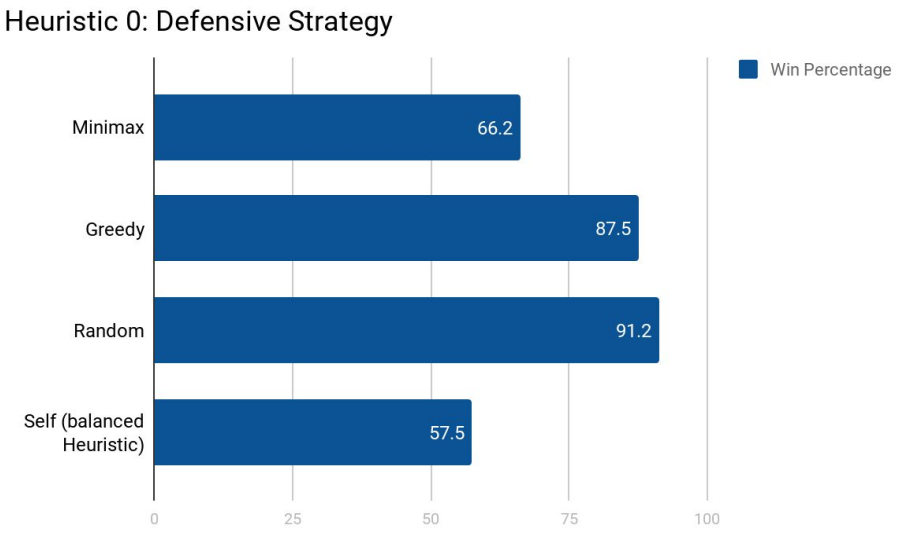
Considering this, we can assume that we give more weight to odd number solutions when it is the current player's turn. Again, as the round passes, the number of blocks decreases, but it is still odd.

So we will give more weight to odd numbers first and then even numbers.

Formula

number_of_player_moves - **2** * number_of_opponent_moves - (-1.1**num_of_blank_spaces)

Performance Comparison



Questions

- **What features of the game does your heuristic incorporate, and why do you think those features matter in evaluating states during search?**

  Explained above

- **Does search speed matter more or less than accuracy to the performance of your heuristic?**

  Yes, I feel that speed matters more as more the speed, more we will be able to explore. If the agents keep on running, then that is not helpful at all. Again these are exponentially difficult problems. So to solve it, we need to give more weight to speed rather than accuracy.

References

http://ajulio.com/assets/documents/Adversarial_Game.pdf