Build a game playing agent for isolation

Heuristic Analysis

By Mahad Amer Akhter

Overview

The aim of this project is to develop an adversarial search agent to play the game "Isolation". Isolation is a deterministic, two-player game of perfect information in which the players alternate turns moving a single piece from one cell to another on a board. Whenever either player occupies a cell, that cell becomes blocked for the remainder of the game. The first player with no remaining legal moves loses, and the opponent is declared the winner.

This report compares different heuristics and discusses their effectiveness

Custom Scores

Heuristic 1

This heuristic maximizes the number of moves available to the current player. It is mathematically expressed as:

$$3 * (number\ of\ my\ moves) - (number\ of\ opponent's\ moves)$$

Heuristic 2

This heuristic maximizes the ratio of the number of moves available to the current player to the opponent. It is mathematically expressed as:

$$\frac{(number\ of\ my\ moves)}{(number\ of\ opponent's\ moves)}$$

Heuristic 3

This heuristic maximizes the number of moves available to the opponent. It is mathematically expressed as:

$$(number\ of\ my\ moves) - (\ 3 * (number\ of\ opponent's\ moves))$$

Evaluation of Custom Scores

The custom scores were evaluated by competing against the following players:

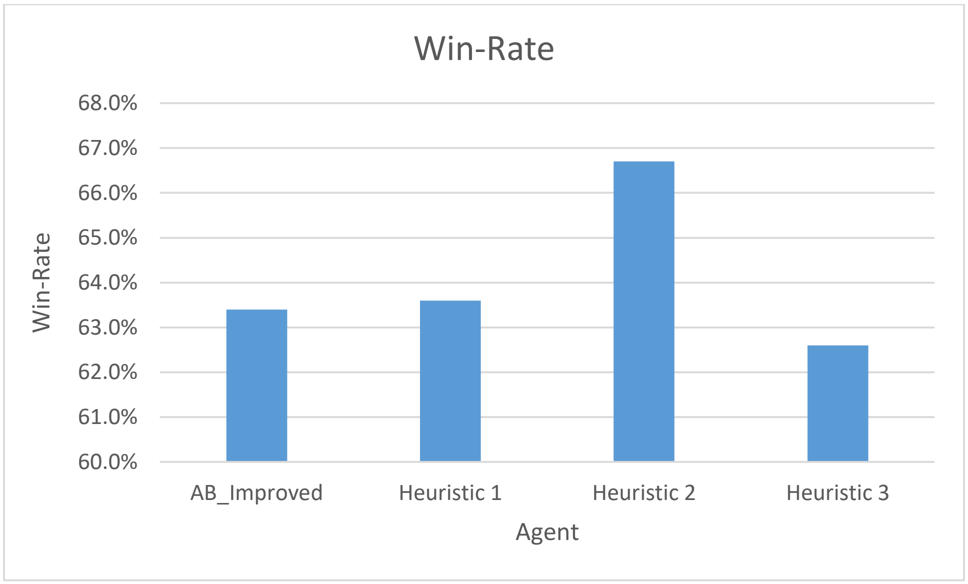
- Random: An agent that randomly chooses a move each turn.
  MinimaxPlayer agent using the open_move_score heuristic with search depth 3
- MM_Center: MinimaxPlayer agent using the center_score heuristic with search depth 3

- MM_Improved: MinimaxPlayer agent using the improved_score heuristic with search depth 3
- AB_Open: AlphaBetaPlayer using iterative deepening alpha-beta search and the open_move_score heuristic
- AB_Center: AlphaBetaPlayer using iterative deepening alpha-beta search and the center_score heuristic
- AB_Improved: AlphaBetaPlayer using iterative deepening alpha-beta search and the improved_score heuristic

Comparison

In order to get a better average, the number of trials against each opponent was increased to 100. The comparison between the Heuristic functions is given in the following table and histogram:

Agent	**Win-Rate**
AB_Improved	63.4%
Heuristic 1	63.6%
Heuristic 2	66.7%
Heuristic 3	62.6%



Heuristic 1 (maximizing the number of moves to the current player) turned out to be the second best option with a win-rate of 63.6%. This is almost equal to the win-rate of AB_Improved

Heuristic 3 (minimizing the number of moves to opponent) turned out to be the worst option with a win-rate of only 62.6%. This performance is even worse than that of AB_Improved

Heuristic 2 (maximizing the ratio of the number of moves available to the current player to that of the opponent) turned out to be the best option with a win-rate of 66.7%. This performance is the best compared to all the other heuristics as well as that of AB_Improved

Conclusion

Heuristic 2 turned out to be best option among all others since it had the highest win ratio. Moreover, it is also very simple to implement in terms of the complexity of the code. Thirdly, the computation of this heuristic is also fast which helps us to iterate deeper into the tree which in turn leads to a better decision. Lastly, in order for the evaluation of this heuristic, no additional memory is required and it is enough to know about the current state of the game.

Hence Heuristic 2 was chosen as the main heuristic for our agent