Isolation Heuristic Analysis

# Daniel Meechan

## Udacity Artificial Intelligence Nanodegree

## Term 1, Project 2 (October 2017)

# Introduction

As part of the Adversarial Search Agent project, I implemented a minimax algorithm with alpha-beta pruning and iterative deepening to create an AI capable of competing at the board game Isolation. Minimax evaluates possible future game states and attempts to maximise the player’s score while assuming that the opponent is trying to minimise the player’s score. Alpha-beta pruning optimises the minimax process by eliminating nodes in the search tree which won’t affect the final decision. Iterative deepening ensures the algorithm will have a result ready when the move timer expires, while still going as deep as possible by searching at increasingly further depths.

With branching factor *b* and search depth *d*, minimax with alpha-beta pruning performs just 2b­­d/2 evaluations (in an optimal situation), which is far fewer than the bd evaluations performed with minimax.

I created three heuristics for scoring a board state. The scoring heuristics help the AI evaluate different moves.

# Heuristic Descriptions

All three heuristics calculate how many legal moves the AI and opponent have available, applies weights to those values, and then subtract the AI’s value from the opponent’s value.

AB\_Custom weights the opponent’s moves much more by doing:

*my\_moves – opponent’s\_moves1.5*

Using exponents leads the AI to be much more avoidant of nodes where the opponent will have many moves available.

AB\_Custom\_2 weights both players’ moves exponentially, while giving the opponent’s value more weighting:

*my\_moves2 – opponent’s\_move2.5*

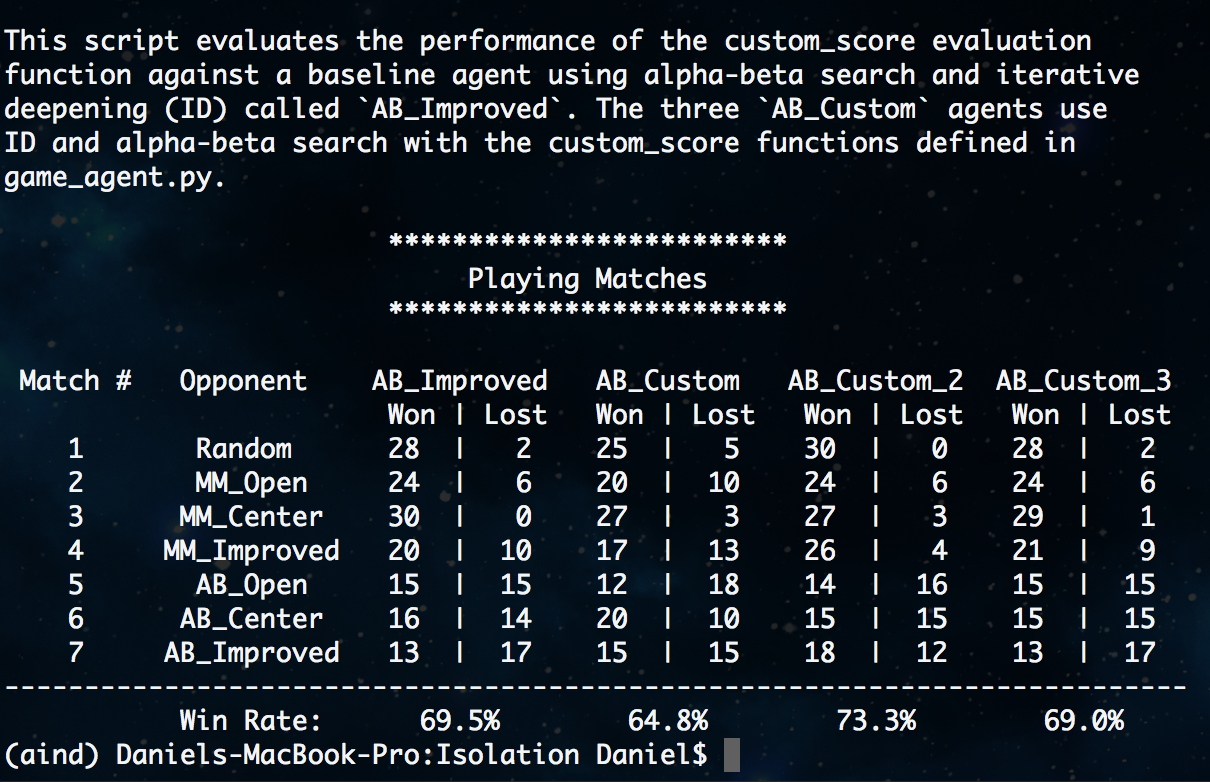
The idea is that the AI will value nodes much more if they give it more moves, but will be incredibly wary of nodes which gives the opponent many moves.

AB\_Custom\_3 is the most aggressive heuristic because it gives very little weighting to the AI’s moves, while giving a lot of weighting to the opponent’s moves:

*my\_moves1/2 – opponent’s\_move2*

# Results and Recommendation

Heuristic AB\_Custom\_2 scored better than the other two heuristics, with AB\_Custom scoring worst (see figure 1).



**Figure 1**: Tournament.py results for 30 matches with a time limit of 150ms

I recommend using AB\_Custom\_2 because it achieved the greatest win rate and scored higher than the AB\_Impoved heuristic (which scored 69.5%). This is likely due to the weightings giving nodes with more moves exponentially higher values, while giving slightly more weighting to the opponent’s moves.

|  |  |  |  |
| --- | --- | --- | --- |
| Heuristic | AB\_Custom | AB\_Custom\_2 | AB\_Custom\_3 |
| Win Rate | 64.8% | 73.3% | 69.0% |