Isolation Game Heuristic Analysis

Arshdeep Tinna

August 6, 2017

Abstract

The objective of this project was to come up with a few different heuristic functions and compare their performance to *AB_IMPROVED* heuristic.

1 Introduction

For this project, I tried four different heuristics. Even though the heuristics explained here performed well, I noticed lot of variation between tournaments mainly because the starting moves are selected randomly.

2 Heuristics

2.1 Heuristic 1

For the first heuristic, I chose to simply enhance *AB_IMPROVED* by adding a value based on position of the two players. To get that value I compared the distance of the two players to the centre and awarded a number ranging between -1 to 1 depending on whether the player is farther or closer to the centre.

2.2 Heuristic 2

This heuristic follows an aggressive strategy and favours moves where

$$my\_moves - 2 * opp\_moves \tag{1}$$

2.3 Heuristic 3

Unlike the last heuristic, I chose to assign a variable coefficient for *my_moves* and *opp_moves* based on ratio of blanks moves to total moves.

$$(1 - blank\_ratio) * my\_moves - blank\_ratio * opp\_moves \tag{2}$$

As you can observe, the heuristic for the first half of the game prefers moves that aggressively try to block the opponents. In the second half, moves that result in larger number of possible moves for the player are preferred

2.4 Heuristic 4

For the last heuristic, I selected an evaluation strategy based on blank to total moves ratio such that

$$h(s,a) = \begin{cases} my\_moves - 2 * opp\_moves, & \text{if } blank\_ratio > 0.65 \\ (1 - blank) * my\_moves - blank * opp\_moves, & \text{if } blank\_ratio > 0.35 \\ my\_moves - opp\_moves & \text{otherwise} \end{cases}$$

Function transitions from aggressively chasing the opponent at the begging to selecting moves that result in maximum moves for the player.

3 Observation

```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                    ************************
                         Playing Matches
                    ************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3  AB_Custom_4
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       9  |  1     10  |  0     9  |  1     8  |  2     8  |  2
    2       MM_Open      7  |  3      6  |  4     6  |  4     7  |  3     7  |  3
    3     MM_Center      7  |  3      9  |  1     9  |  1     7  |  3     8  |  2
    4    MM_Improved     6  |  4      4  |  6     6  |  4     4  |  6     5  |  5
    5       AB_Open      6  |  4      8  |  2     3  |  7     7  |  3     6  |  4
    6     AB_Center      5  |  5      7  |  3     6  |  4     6  |  4     8  |  2
    7    AB_Improved     5  |  5      3  |  7     6  |  4     6  |  4     6  |  4
--------------------------------------------------------------------------------
          Win Rate:      64.3%        67.1%        64.3%        64.3%        68.6%

Your ID search forfeited 208.0 games while there were still legal moves available to play.
```

As already noted earlier that although random opening moves introduced variations across different runs of the tournament most of the time all heuristics performed better or as well as *AB_IMPROVED* function. However the

best performing heuristic came out to be H4 outperforming *AB_IMPROVED* by 4.3%.

Reason for H4 performing well is that it is based on the intuition that using just one strategy thorough out the game ignores information about the overall environment.

By choosing from the three heuristic mentioned above based on the overall game state the player transitions from opponent blocking aggressive one to a defensive one.

While H3 also accounts for overall board state, it range of variation is much larger. H3 goes from depending heavily on open moves for opponent in the beginning to open moves for the player by the end. H4 puts on uses sensible upper and lower bounds.

Another thing to note about the function is it's not slower than other functions because it uses only one strategy with little additional computation for finding blank_ratio.