Heuristic Analysis - Jasper Liu

Abstract:

In this advance game playing project, I implemented the minmax search algorithm and an updated version of minmax search with alpha beta pruning. Also, I created three different scoring heuristic to drive the game agent to decide which moves will be more likely to lead to a better end result. In tournament.py, a game play scenario is prepared, where two set of players will compete with each other with different heuristic, and the three heuristic will be labeled as AB_Custom, AB_Custom2, and AB_Custom3. The first gaming heuristic is based on the amount of legal moves from both player, the second heuristic is solely based on each move's position against the center of the board, and the third one is a twisted version of first heuristic with applying logarithmmic function.

Heuristic #1 (AB_Custom)

score = alpha * # of my legal moves + beta * #of the opponent's legal moves, where alpha =1 and beta = -1.8

The motivation of this scoring system is driven from the winning condition of the isolation game. A player wins the game if the opponent has no more legal moves. Therefore, we can describe the probability of winning the game with a linear function of the number of my legal moves - the number of my opponent's legal moves. With this heuristic, the game agent will favor the board state where its own legal moves are maximized and the opponent's moves are minimized. Also, I added different weights to the two variables, where the number of opponent's moves will cause a larger impact to the final score, to make the game agent more aggressive to isolate the opponent into a board where its moves will be minimized.

Heuristic #2 (AB_Custom_2)

score = distance of the current move to center of the board

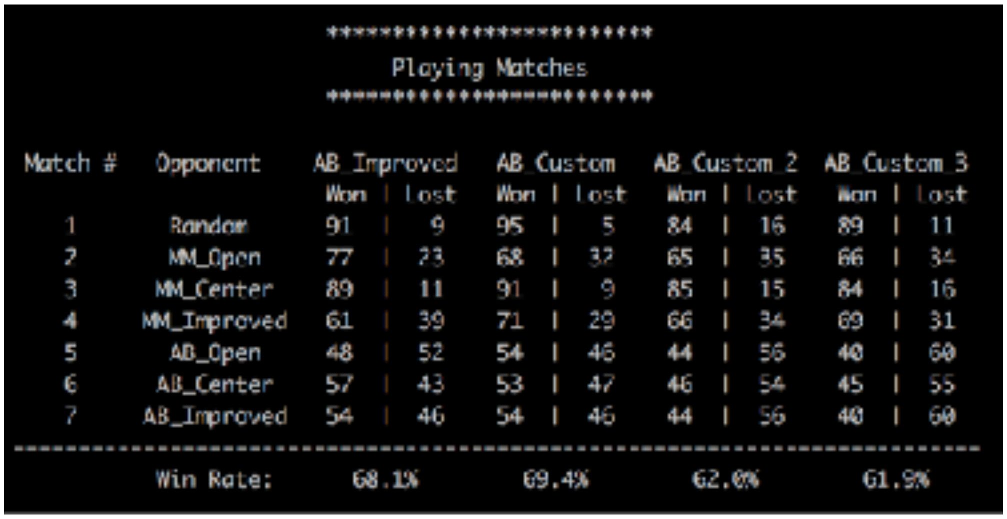
This heuristic only focus on the location of the current move and ignores all other variables. Therefore, it will make the game agent always prefer the moves that is further away from the center, which will make it occupies the position from the outer ring and cycle inward. The main motivation behind is to close off the opponent from outside and make the opponent unable to move eventually.

Heuristic #3 (AB_Custom_3)

score = alpha * log (# of my legal moves / 8) + beta * log (8 / #of the opponent's legal moves), where alpha = 100 and beta = 140

The third heuristic is very similar to the first heuristic in term of control variable. However, it utilizes the property of the log function, so when the number of the legal moves are closer to 0, the score will be significant higher versus when number of moves are at the higher end (6 to 8 , noted that 8 is a maximum moves where a player can have). With the equation above, game agent will contribute 0 if it has a 8 moves available and negative value if it has less than 8 moves. On the other hand, the score will be drastically improved if the opponent have small number of available moves.

<u>Data:</u>



```
                         *************************
                              Playing Matches
                         *************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       91 |   9    95  |   5    84  |  16    89  |  11
    2       MM_Open      77 |  23    68  |  32    65  |  35    66  |  34
    3      MM_Center     89 |  11    91  |   9    85  |  15    84  |  16
    4     MM_Improved    61 |  39    71  |  29    66  |  34    69  |  31
    5       AB_Open      48 |  52    54  |  46    44  |  56    40  |  60
    6      AB_Center     57 |  43    53  |  47    46  |  54    45  |  55
    7     AB_Improved    54 |  46    54  |  46    44  |  56    40  |  60
--------------------------------------------------------------------------
           Win Rate:      68.1%        69.4%        62.0%        61.9%
```

(Figure 1)

The table above shown the results of the game between two different game agent. They competed each other for 100 rounds and each one will have 50 rounds to begin the game. The heuristic that considered both itself and the opponent's status turned out to be the best heuristic out of the three. It was interesting to find that heuristic two (distance to center) and heuristic three (logarithmic function) had about the similar performance and that heuristic one performs much better than heuristic three. I believed the reason that heuristic three did not perform as well as expected is due to the small variation of the variable since it only range between 1-8, so game agent was not able to predict the game outcome as accurate as the heuristic without logarithmic function. Also, one interesting finding from the matches above is that alpha beta pruning performs much better than minmax search since it was able to perform the search more efficiently and to cover more depth before making a decision.

Since the first heuristic seems to be having the best win rate, I decided to further adjust the weighting parameter to see if I can increase the winning rate.

	Alpha = 1, Beta = -1.8	**Alpha = 1.5, Beta = -1.8**	**Alpha = 1, Beta = -2.5**	**Alpha = 1, Beta = -1**
Win Rate	67.9%	65.0%	66.4%	64.3%

(Table 1: results of first heuristic with NUM_MATCHES = 10)

<u>Recommendation:</u>

Out of the three heuristic, I would recommend heuristic #1 (AB_Custom) with alpha = and beta = , based on the following reasons.

score = alpha * # of my legal moves + beta * #of the opponent's legal moves, where alpha = 1 and beta = -1.8

1. Win rate: Based on the win rate, this heuristic is obvious the best out of the comparing algorithms. It was able to defeat the AB_Improved, provided by the Udacity, and all other my heuristic.

2. Correlation to the final game outcome: it turns out the simple number of legal moves is highly correlated to the game results. Considering this scenario: player A has 5 moves and player B has only 1 moves, in general, player A would have a higher win rate since it has 5 different branch of tree for it to figure out the best board that will give him the best outcome. On the other hand, player B only has one moves, and if that move results in a terminal state, which means there are no more moves, it will lose regardless how good its algorithm is. This is the reason this heuristic is recommended since it is highly correlated to a player's win rate.

3. Simplicity: This scoring system is fairly simple, which means you can easily calculate the score without traversing the tree further or making intense computation. Since the scoring function is invoked when we we are at the leaf nodes based on our search depth level, it is executed multiple time depending on the number of branches we have. If we have a less computational intense function, it will allow us to return the search faster, and search deeper in the tree, which often helps the game agent to make a better decision.