Performance Evaluation of Three Heuristic Functions

Introduction and Preliminaries

- The isolation game is fully deterministic. Given hypothetical infinite resources, the game state can be determined a priori by examining and scoring all available options.
- As per problem description, the player can move in L shape if the target square is valid. This indeed simplifies forecasting legitimate moves. Therefore, the following rules apply:

    1. Given the L-shape restriction, the next valid move, cannot be found on either diagonal of the current piece position.
    2. If a player at any of the (4) corners, then he would only have 2 valid moves.
    3. If a player at the center of at least (5x5) board, then he would have the maximum number of moves (8)
    4. If a player is at one of the tiles (border lines), then at maximum he would have (4) valid moves.

- One may be tempted to assign a low score to a player if he is at a corner, and a high one if he was at the center. However, an excellent heuristic should consider the probability of landing on a next better square given the current position. For example, a player should be given a high score if he was at a corner, and his next move would be to the center. On contrary, a move should be given a low score if it was towards the center while the one that follows will force the player to land in the corner!
- Let N be the leading dimension of the board of size $\boldsymbol{N}\times\boldsymbol{N}$. The following results can be easily shown (I already did the analysis on papers and I am only showing the results here)

  Given a free $\boldsymbol{N}\times\boldsymbol{N}$ game of with N at least 5. Then

    1. The total number of (8) possible moves is $(N-4)^2$
    2. The total number of (6) possible moves is $(N-3)\times 4$
    3. The total number of (4) possible moves is $(N-2)\times 4$
    4. The total number of (2) possible moves is at maximum 8.

  For example, and without loss of generality, consider a 7x7 Grid,

    1. The total number of (8) possible moves is $(7-4)^2 = 9$
    2. The total number of (6) possible moves is $(7-3)\times 4 = 16$
    3. The total number of (4) possible moves is $(7-2)\times 4 = 20$

- Moreover, a good heuristic should keep track of the opponent moves, so that we limit his options as he plays.

Results

The following snapshot was taken after running tournament.py file. On average, the best customized scoring function outperforms all other functions against 7 different opponents.

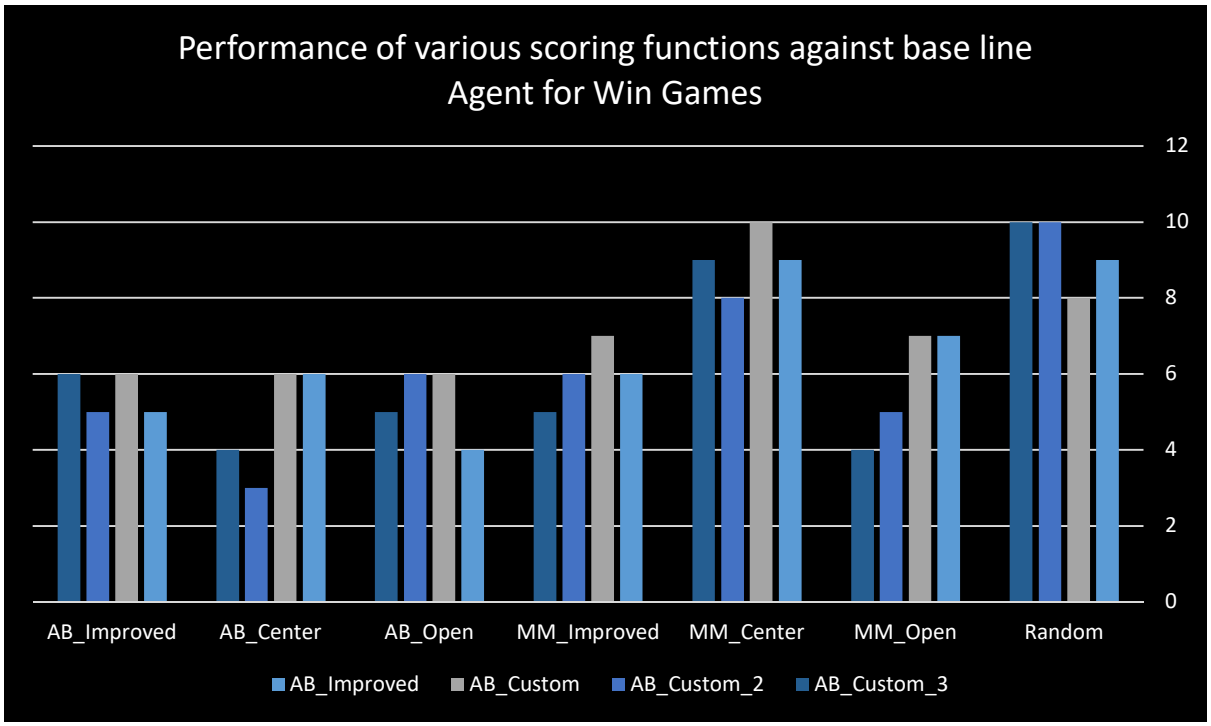
```
(aind) F:\0_Development Folder\AI_ND\P2_v2_Isolation>python tournament.py

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
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       9  |  1     8  |  2     10 |  0     10 |  0
    2       MM_Open      7  |  3     7  |  3      5 |  5      4 |  6
    3      MM_Center     9  |  1     10 |  0      8 |  2      9 |  1
    4     MM_Improved    6  |  4     7  |  3      6 |  4      5 |  5
    5       AB_Open      4  |  6     6  |  4      6 |  4      5 |  5
    6      AB_Center     6  |  4     6  |  4      3 |  7      4 |  6
    7     AB_Improved    5  |  5     6  |  4      5 |  5      6 |  4
--------------------------------------------------------------------------
          Win Rate:      65.7%        71.4%        61.4%        61.4%
```

The following figure, considers only the wining instances of all the players.



As can be clearly seen, the performance of our customized score is either comparable or out-perform the one provided by AB_Improved.

Discussion and Analysis

1. custom_score_3

This heuristic relates the probability of available moves given their current position to both the player and the opponent next moves. The number of next valid moves for both players is weighted by the returned probability. If the opponent score is greater than student score, the final return score is downgraded. On the other hand, if student score is higher, the returned score is emphasized by multiplying it by another weight. The cutting threshold and both weights are experimentally determined and tuned.

This heuristic surely outperforms any random scoring but may fail if the opponent took advantage of other spatial information.

2. custom_score_2

This heuristic considers solely spatial information. The following applies.

1. It detects the location of the opponent, and assign a high score if my next position is in his next valid moves list. By this we aim at limiting his options and forcing him to go in a certain direction.
2. If my opponent is at either an edge or a corner and many squares are occupied already, then this is a good sign, the returned score should be high
3. If on the other hand, I was at an edge or a corner towards the end of the game, I should penalize my move severely in order not to be trapped with limited options.

This heuristic assumes the opponent is not so intelligent as he eventually will be trapped by his own decisions. No wonder then, this functions outperforms the Random Player and performs well with others.

3. custom_score (Best Heuristic)

Combines the lessons learnt from the previous two heuristics by merging both probabilistic forecasting and spatial information.

1. The best move at the beginning of the game would at the center. It gives the maximum number of options for the next move. If the board is 5x5 or more, then MAX player will have at least 7 more moves for his next turn. The player who controls the center of the game, is more likely to win. Therefore, any move towards the center should be highly rewarded.
2. The final score is formed by combining various portions of the previous scores and this new established weight.

No wonder that on average, this heuristic outperformed all others. It be improved dramatically. In the following recommendation section I highlighted some of the suggested improvements.

General Recommendations

1. Establishing more robust methodology for calculating the probabilities in ***custom_score_3(…)*** by taking into considerations overlapping squares. Also, coming up with better factors to derive the joint probability for a piece given its opponent, location and surrounding.
2. Establishing a better and more accurate way for relating current to opponent positions in ***custom_score_2(..).*** One suggested way is to choose a square within the moving area of the opponent after gathering enough information about the game state, the occupied squares and the probability that the opponent is trapped.
3. A better way to choose parameters alpha, beta, and gamma at the end of ***custom_score(..)*** is range of values for those variables and then running various instances of the game to choose the best ones. One can also utilize some machine learning techniques to accomplish the previous task.
4. For a more stable results' reporting, multiple instances of the game should be run for all players and scores. Various statistics should be eventually returned about the performance.
   The following is a result of running another instance:

My Recommendation

For this project and despite its complexity, I would recommend using the last custorm_score(…) that combines both probabilistic and spatial information for the following reasons:

- It provides the best score for various runs.
- Easy to implement and comprehend. Therefore, easy to maintain and upgrade.
- Its flexibility of capturing different features and its adaptability & suitability to various environments and board sizes.