Build an AI agent for Game-Playing: Heuristic Analysis

This project develops an AI agent using Minimax and Alpha-beta pruning in conjunction with iterative deepening for Isolation Game. The developed AI agent is evaluated by using three different heuristic score functions and the aim is to choose best function which results in higher winning rate in comparison with the baseline agent (AB_improved).

In order to evaluate the board state, an evaluation function or heuristic function has to be defined. Depending on each game and strategy for the game, the definition of the evaluation function could be built accordingly. In the Isolation Game, the objective of our player is being the last player to have a legal move and thus, heuristic strategies have to be toward this end by maximizing the number of legal moves for our player and also minimizing the opponent's number of available moves at each board state. Besides, in designing the heuristic, the trade-off between complexity versus performance has also been carefully taken into account (i.e., the more complex of heuristic function could prevent the depth of search given the limited time). In the lecture, the following heuristic function has been suggested:

number of our player's available moves - number of opponent's available moves

This approach is naive since it assumes the equal importance of the number of our player's moves and the opponent's moves. Based on this naive case, we consider three following variants

1. custom_score3 (Heuristic 3): Aggressive strategy when our player put more emphasis on trying to obstruct the opponent.

   number of our available moves – 2 number of opponent's available moves*

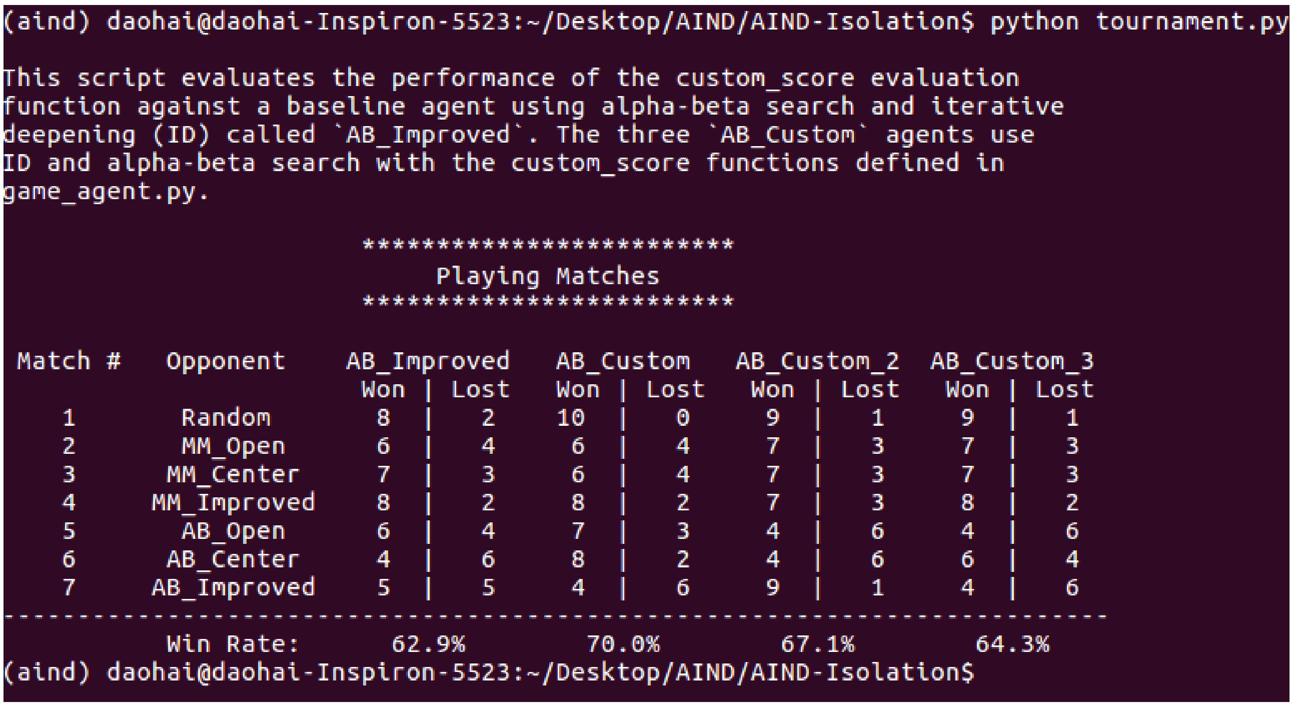
2. custom_score2 (Heuristic 2): Defensive strategy when our player put more emphasis on having as many legal moves as possible

   *2*number of our available moves – number of opponent's available moves*

3. custom_score (Heuristic 1): The selection of aggressive or defensive strategy depending on the information of board occupation. The intuition is that if the board occupation is small, the objective is emphasized on maximizing our player's available move while in the case of large board occupation, the emphasis is on obstructing the opponent's moves.

```
if percent_board_occupied < 30 :
        return float(player_moves - 2*opponent_moves)
 else:
        return float(2*player_moves - opponent_moves)
```

Heuristic Comparison:



```
(aind) daohai@daohai-Inspiron-5523:~/Desktop/AIND/AIND-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       8  |   2    10  |   0     9  |   1     9  |   1
    2       MM_Open      6  |   4     6  |   4     7  |   3     7  |   3
    3      MM_Center     7  |   3     6  |   4     7  |   3     7  |   3
    4     MM_Improved    8  |   2     8  |   2     7  |   3     8  |   2
    5       AB_Open      6  |   4     7  |   3     4  |   6     4  |   6
    6      AB_Center     4  |   6     8  |   2     4  |   6     6  |   4
    7     AB_Improved    5  |   5     4  |   6     9  |   1     4  |   6
--------------------------------------------------------------------------
           Win Rate:      62.9%        70.0%        67.1%        64.3%
(aind) daohai@daohai-Inspiron-5523:~/Desktop/AIND/AIND-Isolation$
```

As it can be seen, AB_Custom heuristic which integrates information on board occupation for deciding strategy (aggressive or defensive) performs better than other heuristics with an overall winning rate of 70%. The aggressive strategy has winning rate of 67.1% while the defensive strategy's winning rate is 64.3%. The baseline AB_improved heuristic has winning rate of 62.9%
Note that the winning rate is dependent on the underlying hardware as the faster computer will be able to perform deeper searches in a given time slot and thus, the figures here makes senses for only comparison.

Our heuristic developed in custom_score satisfies the goal of simplicity and highest performance and thus, being the best choice among three heuristics.