Heuristic Analysis for the Project -2 AIND

By: Abuhanif Bhuiyan, Feb 2018

Introduction:

This project includes the application of AI in game of "Isolation" by developing an adversarial search agent to play the game. Isolation is a two-player deterministic game, in which the players alternate turns moving a single piece from one cell to another on a board. Whenever either player occupies a cell, that cell becomes blocked for the remainder of the game. The first player with no remaining legal moves loses. There were three heuristic functions developed in this game agent to compare the relative performance of those functions. Description of the function application their related performance is explained below.

Heuristic Functions and Analysis:

1st Function: This function calculates the total number of available moves each player has and compare the numbers. I have used a weight factor by multiplying the player move by 2. The player got more moves is considered in advantages position.

Code:

```
player_moves_no = len(game.get_legal_moves(player))
oppont_moves_no = len(game.get_legal_moves(game.get_opponent(player)))
return float(player_moves_no *2 - oppont_moves_no)
```

2nd Function: This function calculates the distance from the center location for each player. The player got distance than the opponent's move from the center location is considered in advantages position.

Code:

```
player_dist_fr_cent = rela_dist_fr_cent(game, game.get_player_location(player))
oppont_dist_fr_cent =
rela_dist_fr_cent(game,game.get_player_location(game.get_opponent(player)))
return float(player_dist_fr_cent - oppont_dist_fr_cent)
```

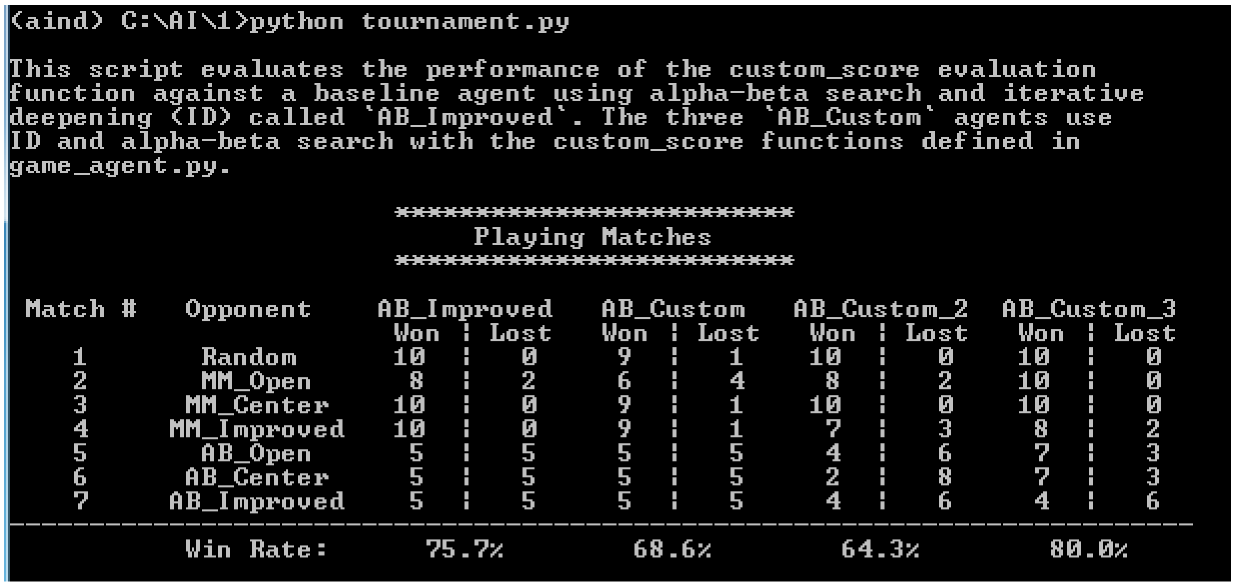
3rd Function: From the class lecture, I got the impression that staying close to the center can increase the chances of winning. So, wanted to take the advantages to keep the number of available move close the center for my player. And got the best result from this function.

Code:

avail_centr_player = sum(rela_dist_fr_cent(game, i_move) for i_move in player_moves)

avail_centr_oppont = sum(rela_dist_fr_cent(game, i_move) for i_move in oppont_moves)

return float(avail_centr_player - avail_centr_oppont)



```
(aind) C:\AI\1>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      10  |   0     9  |   1    10  |   0    10  |   0
    2       MM_Open      8  |   2     6  |   4     8  |   2    10  |   0
    3      MM_Center    10  |   0     9  |   1    10  |   0    10  |   0
    4     MM_Improved   10  |   0     9  |   1     7  |   3     8  |   2
    5       AB_Open      5  |   5     5  |   5     4  |   6     7  |   3
    6      AB_Center     5  |   5     5  |   5     2  |   8     7  |   3
    7     AB_Improved    5  |   5     5  |   5     4  |   6     4  |   6
--------------------------------------------------------------------------
           Win Rate:      75.7%        68.6%        64.3%        80.0%
```

Fig1: Snapshot of the match results. AB_Custom3 got the best winning result.

Conclusion:

It is observed that custom heuristic function performed differently over the others. From my result it is found that the 3rd function performed better than the other two. In conclusion, I can say that within this limited capability of the game agent, the player got more move available near the central point will have better changes of winning. So the player need to find the next move which will keep its number of central move more than opponent, that player's wining changes are more.