Heuristic Analysis

Advance Game playing Agent

Introduction :

Advance Game Playing Agent project is to crate game agent to play Isolation game by implementing two types of agents: Minimax Agent - use minimax algorithm to apply depth-first search for possible moves in each isolation game state. Alpha-Beta Agent - apply alpha-beta pruning and treatise deepening to improve performance of minimax agent. Agents goal is to win the game. To do so, they need evaluation function which indicates score of players in each game state and assigns value according to situation at the time of evaluation. Score in isolation is defined according to game's rules. Players take turn and move in L-shaped from current position. Players with no legal move left lose the game.

Evaluation functions:

1. Available Moves Difference (Customer score 1):

Eval_fn = #my_legal_moves - # opponent_legal_moves

Available Moves Difference gets number of legal moves of each player then find the different between legal moves of active player and inactive player. Return that value as game score.

This function returns higher value when the player has more legal moves than opponent's, As it return the difference between them.

Output of this function is simple and well represent states that player win the game. Range of this function is [-49, 49]

2. Customer score 2:

Eval_fn = #my_legal_moves – 2 * # opponent_legal_moves

It's the difference between the number of legal moves of the active player and inactive

player multiply by 2 (as lectures).

3. Ratio of Available Moves (customer score 3) :

Eval_fn = #my_legal_moves / # opponent_legal_moves

Ratio of Available Moves gets number of legal moves of each player then divides legal moves of active player with legal moves of inactive player. Return that value as game score.

Ration of Available Moves return more drastic change comparing with Available Moves Difference. Range of this function is [0, inf).

Since denominator could be zero and could lead to error, if-loop is implemented before calculate score to check if opponent's move is zero, player is the winner.

Performance and Analysis:

```
 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       9  |   1    10  |   0    10  |   0     4  |   6
    2       MM_Open      6  |   4     8  |   2     8  |   2     0  |  10
    3      MM_Center     8  |   2    10  |   0     7  |   3     0  |  10
    4     MM_Improved    5  |   5     8  |   2     9  |   1     1  |   9
    5       AB_Open      5  |   5     6  |   4     5  |   5     0  |  10
    6      AB_Center     4  |   6     2  |   8     5  |   5     0  |  10
    7     AB_Improved    6  |   4     5  |   5     4  |   6     0  |  10
--------------------------------------------------------------------------
           Win Rate:      61.4%        70.0%        68.6%        7.1%

Your agents forfeited 160.0 games while there were still legal moves available to play.
```

The result show that customer score 1 performed better than 2 and 3 .

The customer score 3 doesn't perform well.

Recommendation Evaluation Function :

I would suggest Different of Available Moves is best evaluation function. This also could be a result of single pre-defined move I used as opening book in the get_move() function. Improve opening book could improve agent's performance, and it's easy to be implemented .