Heuristic Evaluations

There were three heuristic evaluation functions to be written to compete against the Computer players.

Custom_score_2:

Number of moves is one of the simple and efficient heuristic evaluation function to guess the game state. The difference between the number of moves between the current player and the opponent is the best heuristic implemented by the computer players. Have tweaked that evaluation function to make the current player little aggressive. The evaluation function will return the difference between the own moves and double the time of the opponent moves.

Custom_score_3:

It's same like custom_score_3 except, its multiplying the opponent moves by 3 which make the player more aggressive in finding the optimum result.

Custom_score:

The goal of this function is to beat the AB_Improved player. Hence the following 3 level strategy was implemented. The game moves have been split into 3 categories.

Category 1: Beginning of the game. Determined by number of blank spaces greater than (board height X board width) – 10. This will be true for around the first 10 moves.

Category 2: Last few moves. Its determined by either player's legal moves comes less than 5.

Category 3: Those moves that does not qualify for Category 1 and Category 2 falls under this category.

Strategy 1(Category 1): In the beginning there will be more number of moves available for the player. Hence the aggressive approach of custom_score_2 has been used here. It will return the difference between the own moves and 2 times the opponent moves.

Strategy 2(Category 2): In the last few moves(less than 5), the difference between the number of moves between own moves and opponent moves would be optimal.

Strategy 3(Category 3): In the middle of the game, it will go ply down one level to see the number of legal moves to its children and return the difference between the max no of legal moves of the among the own's and opponent's legal moves.

The following two results depicts the performance against AB_Improved,

RUN 1:

```
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     9  |   1     9  |   1
    2       MM_Open      9  |   1     8  |   2     6  |   4     9  |   1
    3      MM_Center    10  |   0    10  |   0     9  |   1     8  |   2
    4     MM_Improved    6  |   4     9  |   1     7  |   3     8  |   2
    5       AB_Open      5  |   5     5  |   5     6  |   4     6  |   4
    6      AB_Center     5  |   5     6  |   4     4  |   6     3  |   7
    7     AB_Improved    5  |   5     5  |   5     3  |   7     4  |   6
--------------------------------------------------------------------------
          Win Rate:      71.4%        74.3%        62.9%        67.1%
```

RUN 2:

```
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     9  |   1    10  |   0
    2       MM_Open      8  |   2     8  |   2     6  |   4     5  |   5
    3      MM_Center     9  |   1     9  |   1     9  |   1     9  |   1
    4     MM_Improved    6  |   4     7  |   3     7  |   3     8  |   2
    5       AB_Open      6  |   4     6  |   4     6  |   4     6  |   4
    6      AB_Center     6  |   4     5  |   5     7  |   3     7  |   3
    7     AB_Improved    4  |   6     6  |   4     6  |   4     5  |   5
--------------------------------------------------------------------------
          Win Rate:      68.6%        70.0%        71.4%        71.4%
```

Performance:

Custom_score (Student) heuristic consistently out beaten AM_Improved heuristic while running the tournament. AM_Improved heuristic is a simple yet powerful heuristic which evaluates fast but it's the same heuristic in all stages of the game. But Custom_score heuristic evaluates based on the current game state which makes it complex and powerful which resulted in beating AM_Improved consistently (71.4% vs 74.3% and 68.6 % vs 70.0%). As Customer score (Student) goes into an addition depth search for the middle of the game, it increases the chances of Timeouts when the board size is very huge and works seamless with the boards (7X7).

Recommended Heuristic:

Custom_score (Student) heuristic would be the best for the isolation game compared to the other heuristic functions because,

1. It has consistently beat AB_Improved(71.4% vs 74.3% and 68.6 % vs 70.0%) heuristic function
2. It has a 3 level of different strategies to use for the heuristic based on the game state
3. Additional one depth search to decide on the heuristic in the middle state of the game

Other Heuristics tried: Splitting the board positions based on the center of the board and assigning different weights to each position based on its location. Its split into 4 and considering a 7X7 board,

1. Center position (1 place). If the legal moves has center position it get the high value of weight(2)
2. 2 Layers above the center position (8 places). The eight board positions that are around the center move are very good places for a knight to move in any direction and hence gets the second highest weight(1.5)
3. The next two layers (16 places) are given less value of weight (1) as the number of possible moves from those places comes down.
4. Corner board positions (24 places) gets the least weight value (0.5) as the moves are restricted from this place.
   The evaluation function will return the difference between the sum of the weights of the legal moves between the own moves and opponent moves.