Heuristic Evaluation

Summary of methods

custom_score(): $\boldsymbol{x = a^2 - kb + c}$

I started off with the simple function $x = a - b$ where a = my_moves and b = opp_moves. I then introduced a tuning parameter (lambda k) and a constant (c being the distance from the center – to reward center control) so that I had: $x = a - kb + c$. I did a few runs with k = 1.5 … 10. I noticed that I got the best results at k = 7. I then base-lined this. I executed a few more runs and eventually I decided I wanted to reward best values of a. I then introduced a square to get custom_score() so that $x = a^2 - kb + c$.

custom_score_2():$\boldsymbol{x = a^2 - kb + e^c}$

Still I thought the number of moves I have (a) and the number of moves opponent has (b) did not quite represent the status of the game completely. Intuitively one imagines that control of the center of the board gives a player a lot more advantage than just having many moves available.

So I moved from just worrying about the number of moves and I thought getting the total sum of all distances of the available legal moves to the center of the board would be more representative of the true game state – assuming that center control was important. So I changed the variable a to $a = \sum_0^{moves} di$ where d_i is the distance of the i^{th} move to the center. I modified b also to have be the sum of all legal moves for the opponent to the center. So this became my custom_score_2() as given by $x = a^2 - kb + e^c$ where c is the difference between my moves and opponent's moves. For custom score_score_3() I wanted to have a predator-prey model.

customer_score_3():$\boldsymbol{x = a * (c - d * b) ** 2 - k * b * (d - c * a) + e^m}$

I didn't quite achieve this but in the end I had $x = a * (c - d * b) ** 2 - k * b * (d - c * a) + e^m$ where a is the distances to the center for player, b is the distances to the center for opponent, c is the distance of the current position of player to the center, d is the distance to the center of the current position of opponent and m is the difference between player moves and opponent moves. This formula takes into account all the factors I considered important (distances to the center, distance of the current positions to the center as well as the difference between the number of moves available.

Analysis of Results

I executed multiple run while changing the different parameters. In the end k=3 seemed to produce stable results across the three custom functions. The best result I got for the custom_score() heuristic function was 70%. This also performed relatively well against AB:

```
Run #8
custom_score()
k=3.0, a = my_legal_moves, b = opp_moves, g_n = a**2 - k*b + distance_to_center
custom_score_2()
k=3.0, a = my_distances_to_center, b = opp_distances_to_center, g_n = a**2 - k*b + my_moves-opp_moves
custom_score_3()
k=3.0, a = my_distances_to_center, b = opp_distances_to_center,
g_n = a*(c-d*b)**2-k*b*(d-c*a) + c

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     9  |   1     8  |   2    10  |   0
    2       MM_Open      7  |   3     9  |   1     7  |   3     5  |   5
    3      MM_Center     9  |   1     9  |   1     8  |   2     7  |   3
    4     MM_Improved    6  |   4     5  |   5     5  |   5     2  |   8
    5       AB_Open      6  |   4     5  |   5     6  |   4     6  |   4
    6      AB_Center     6  |   4     7  |   3     3  |   7     4  |   6
    7     AB_Improved    5  |   5     5  |   5     5  |   5     4  |   6
--------------------------------------------------------------------------
           Win Rate:      67.1%        70.0%        60.0%        54.3%

There were 1.0 timeouts during the tournament -- make sure your agent handles search timeout correctly,
and consider increasing the timeout margin for your agent.


Your ID search forfeited 244.0 games while there were still legal moves available to play.
```

Figure 1 - Best Score for custom_score() heuristic

The worst result I got for the custom heuristic was 47% win rate and this was largely due to poor performance against AB. Custom_score() also performed consistently well against Minimax but just above average against Alphabeta.

The best score I got for my custom_score_2() was a win rate of 73%. The rest of the runs were between 60% and 70%. This suggests that a policy of computing total distances to the center is good in the long run. There were only two results below 60% out of over 12 runs. Again suggesting distances to the center are key in Isolation.

```
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     7  |   3    10  |   0     8  |   2
    2       MM_Open      7  |   3     5  |   5     8  |   2     3  |   7
    3      MM_Center     6  |   4     8  |   2    10  |   0     9  |   1
    4     MM_Improved    9  |   1     4  |   6     8  |   2     5  |   5
    5       AB_Open      5  |   5     2  |   8     5  |   5     6  |   4
    6      AB_Center     5  |   5     4  |   6     4  |   6     5  |   5
    7     AB_Improved    3  |   7     3  |   7     6  |   4     4  |   6
--------------------------------------------------------------------------
           Win Rate:      61.4%        47.1%        72.9%        57.1%

There were 1.0 timeouts during the tournament -- make sure your agent handles
search timeout correctly, and consider increasing the timeout margin for your agent.


Your ID search forfeited 246.0 games while there were still legal moves available to play.
```

Figure 2 - custom_score_2() best result

The worst result I got for the custom_score_2() heuristic was a 51.4%. For this result custom_score_2() was demolished by Alphabeta.

custom_score_3() had the best result of 67%. It did well in that run against MM but struggled with AB. Strangely I also observed that where I didn't do so well against MM, my custom heuristics performed better against AB.

```
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     8 |   2     9 |   1     10 |   0
    2       MM_Open       5 |   5     8 |   2     5 |   5      9 |   1
    3       MM_Center     6 |   4     7 |   3     7 |   3      8 |   2
    4       MM_Improved   5 |   5     6 |   4     8 |   2      8 |   2
    5       AB_Open       8 |   2     4 |   6     5 |   5      4 |   6
    6       AB_Center     6 |   4     2 |   8     5 |   5      4 |   6
    7       AB_Improved   3 |   7     5 |   5     6 |   4      4 |   6
--------------------------------------------------------------------------
            Win Rate:      61.4%        57.1%        64.3%        67.1%
```

Figure 3 custom_score_3() best result

The worst result obtained by custom_score_3() was 55.7%.

In conclusion, all my heuristics performed well against Random and MM but struggled a bit with AB. My heuristics handle center control, availability of legal moves and mobility quite well. I need to look into dealing partitioned boards, reflection and blocking potentially dangerous moves by the opponent to gain those marginal improvements in performance.

I would recommend the use of custom_score_2() because it produced the highest score of 73%. It also produced the highest runs with an above 60% win rate. Lastly, it handles, key board factors such as distances to the center (which is the overall board state) and the number of legal moves between the two players.