Heuristic Analysis

For Udacity AIND Isolation Game Playing Agent Project

In this report we will be evaluating performance of all three evaluation functions, their results and based on comparison we will chose the best heuristic.

Heuristics

Custom_score : This function is a rescaled version in which I tried to get the evaluation value on a scale and hence used the evaluation technique of (value - minValue)/(maxValue - minValue) along with the custom score 3.

Custom_score_2 : This function used the same simplistic approach of difference between the player moves and opponent moves but in this I multiplied opponent moves by 2 and divided the whole value with free spaces. My idea is that as game progress and free spaces decreases the evaluation value should increase if a player has more moves.

Custom_score_3 : This function used the id improved approach for some cases and for the other it used the difference of the player's distance from the center of the game board divided by 10.

Results

Here are the Results of all the three score functions when pitted against the improved function. The screenshot from the *tournament.py* is given below.

```
**************************
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      9  |  1
    2       MM_Open     7  |  3      7  |  3      7  |  3      6  |  4
    3      MM_Center    9  |  1      7  |  3      7  |  3      8  |  2
    4     MM_Improved   7  |  3      6  |  4      6  |  4      7  |  3
    5       AB_Open     6  |  4      5  |  5      7  |  3      7  |  3
    6      AB_Center    5  |  5      6  |  4      6  |  4      6  |  4
    7     AB_Improved   4  |  6      8  |  2      6  |  4      6  |  4
--------------------------------------------------------------------------
          Win Rate:      65.7%        68.6%        67.1%        70.0%
```

As we can see all the three score functions performed better than the improved function. However the custom_score_3 performed best with 70% winning rate and custom_score performed second best with 68.6% winning rate and custom_score_2 with 67.1%. Even though the custom_score function somewhat underperform against AB_Open it still manages to get more winning rate than custom_score_2.

Comparing Heuristics

As we can see in the results that all three score functions have very close winning rate in a span of 3%. This makes us suggest that certain strategies are suitable for certain conditions. And there may not be a strategy which can be suitable for all conditions so rather than having a single evaluation strategy we should have an adaptive strategy which can learn from the opponent's behaviour and adapt dynamically to improve itself.

Best Heuristic

From the data shown we can see that *custom_score_3* is the best heuristic. But in my opinion *custom_score* seems to be the best heuristic. Since

a. It performs far better than all other heuristics against id improved function with best score of **8-2** and performs the same like all other heuristics for remaining scenarios.
b. It is also simpler than other heuristics as it uses a simple technique of re-scaling your values in an interval. This eventually helps in minimizing errors in the evaluation values and get them in a certain range.
c. It uses the approach used in *custom_score_3* if player moves is equal to opponent moves and falls only 1.4% short in winning rate but can be tuned with other heuristic approaches to get better results (this again can depend on certain conditions as mentioned in section above). And since the difference is not that large it is easily possible.

I would also like to work on the custom_score_2 and try to include a horizon effect factor in denominator with free spaces as game progresses. (But time's a constraint and till now I am not able to completely lay it out and would like to get some help if you can give it).