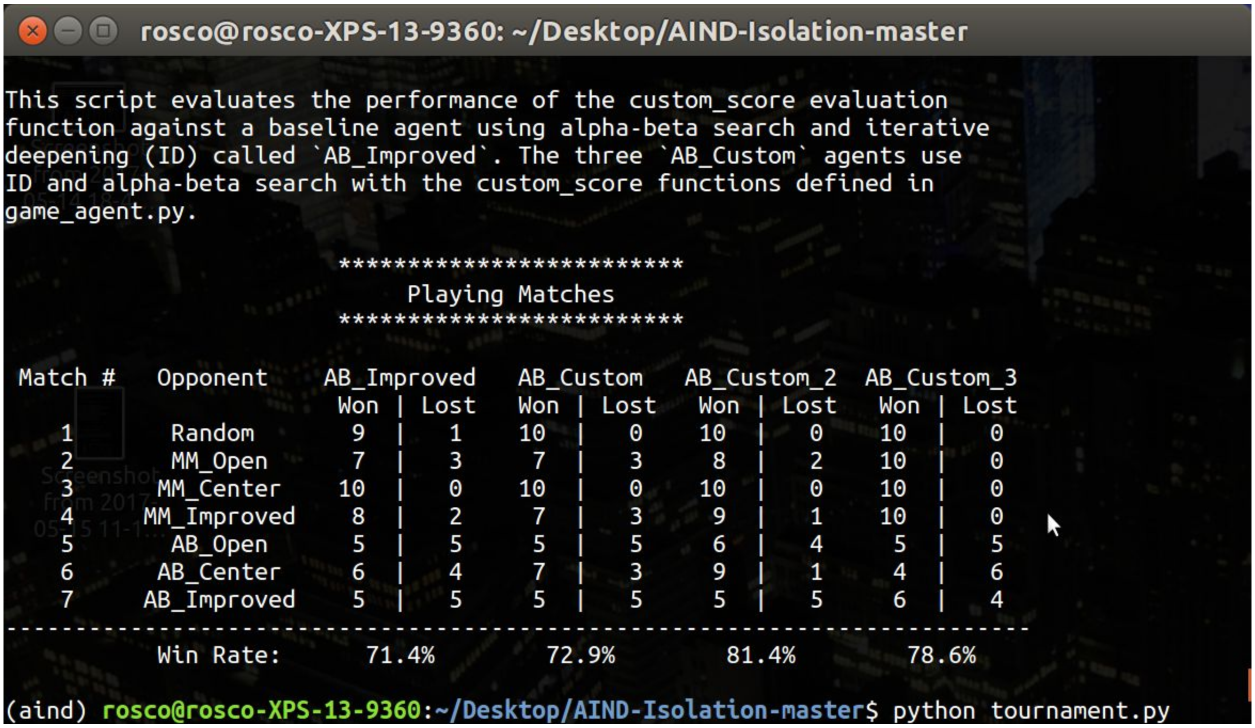
Isolation Heuristic Review

By: Stephen Rawson, Udacity AI Nanodegree, Term 1 (April 2017)

For the Advanced Game Playing project, we were to implement the min/max, alpha-beta pruning, and iterative deepening algorithms. I created three different heuristics that had similar outcomes. Figure 1, provides the win rate of the three heuristic functions vs. AB_improved when running the default tournament.py script.



```
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    10  |   0    10  |   0    10  |   0
    2       MM_Open      7  |   3     7  |   3     8  |   2    10  |   0
    3      MM_Center    10  |   0    10  |   0    10  |   0    10  |   0
    4     MM_Improved    8  |   2     7  |   3     9  |   1    10  |   0
    5       AB_Open      5  |   5     5  |   5     6  |   4     5  |   5
    6      AB_Center     6  |   4     7  |   3     9  |   1     4  |   6
    7     AB_Improved    5  |   5     5  |   5     5  |   5     6  |   4
--------------------------------------------------------------------------
           Win Rate:      71.4%        72.9%        81.4%        78.6%

(aind) rosco@rosco-XPS-13-9360:~/Desktop/AIND-Isolation-master$ python tournament.py
```

Figure 1. Tournament.py results. 10 Matches and 150 ms Time limit

Heuristic Descriptions

Custom_score takes the difference of the percentage of legal_moves a player has vs. the number of remaining the blank spaces. The more legal moves a player versus blank spaces allows the player become The opponent value is doubled to provide more aggressive play.

Custom_score_2 is the difference of legal moves multiplied by a in probability value whose sum equals one. The opponent weight is .95 and your own equals .05. This provides a more conservative game play deeper into the the game. Reversing the probability values resulted in a higher loss rate. Ultimately, I decided use .95/.05 combo because it seemed to have more consistent results.

Custom_score_3 is the difference of legal moves. The opponents moves are doubled then increased by 100%. My rationale was to guarantee that the opponent player always had a commanding lead. Thus, your own player would be ultra aggressive.

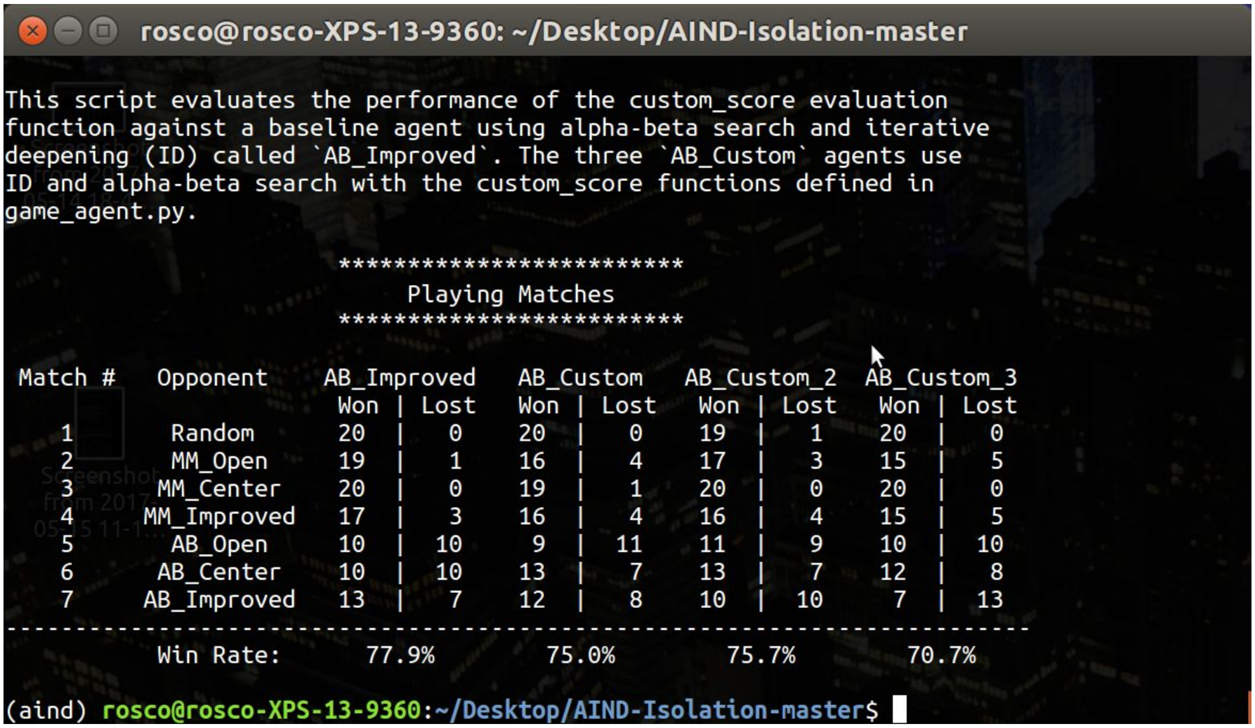
Results

All Three seem to win around 70% of the time. Adjusting Time seemed to negatively impact my alphabeta pruning techniques. It is possible that I might need to make my weights less or compare against neighboring moves differently.

Recommendation

I would recommend custom_score_2 because it had the most consistent win rate. The heuristic also weighs the opponent's number of moves more heavily even if equal. This possibly provides deeper searches prior to pruning potential leafs. It is also incredible fast to compute, O(1), and easily interpretable. I got the idea for .05 from the “Game Tree Searching by Min / Max Approximation” article.

Custom_score	75.3
Custom_score_2	**76.9**
Custom_score_3	75.1



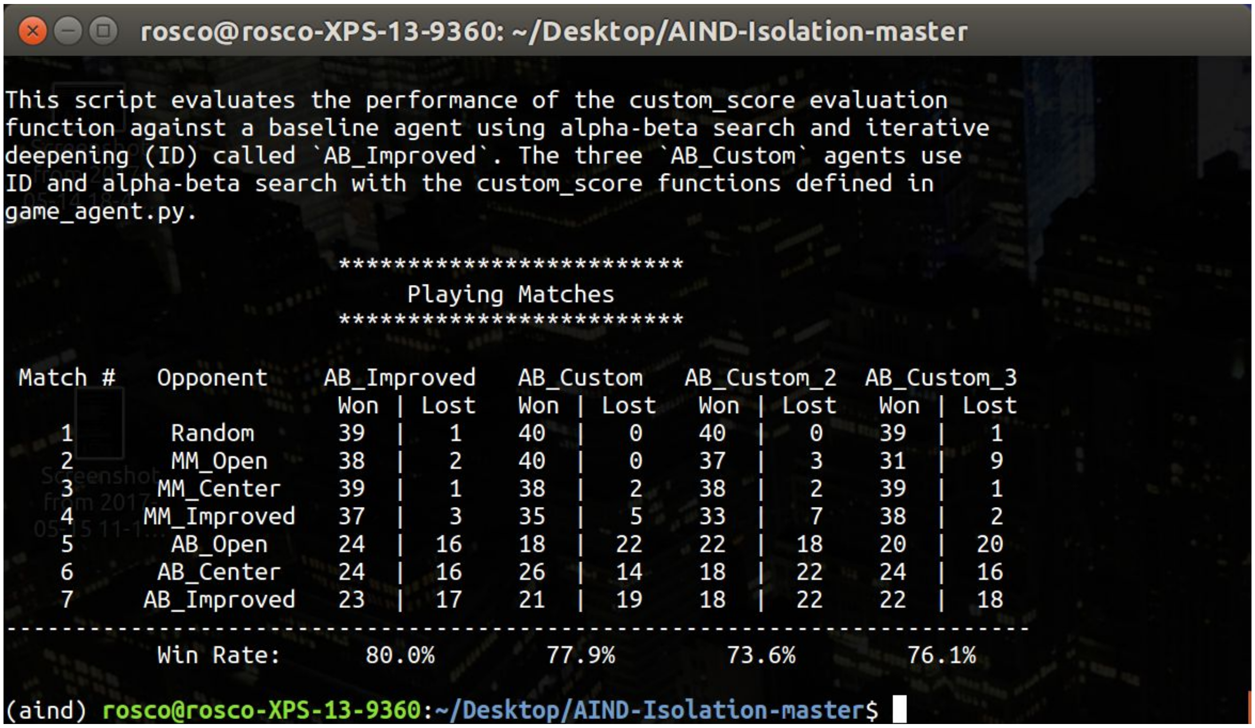
```
This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                        *************************
                             Playing Matches
                        *************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      20  |   0    20  |   0    19  |   1    20  |   0
    2       MM_Open     19  |   1    16  |   4    17  |   3    15  |   5
    3      MM_Center    20  |   0    19  |   1    20  |   0    20  |   0
    4     MM_Improved   17  |   3    16  |   4    16  |   4    15  |   5
    5       AB_Open     10  |  10     9  |  11    11  |   9    10  |  10
    6      AB_Center    10  |  10    13  |   7    13  |   7    12  |   8
    7     AB_Improved   13  |   7    12  |   8    10  |  10     7  |  13
--------------------------------------------------------------------------
           Win Rate:      77.9%        75.0%        75.7%        70.7%

(aind) rosco@rosco-XPS-13-9360:~/Desktop/AIND-Isolation-master$
```

Figure 2. Tournament.py results. 20 Matches and 150 ms Time limit.



```
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      39  |   1    40  |   0    40  |   0    39  |   1
    2       MM_Open     38  |   2    40  |   0    37  |   3    31  |   9
    3      MM_Center    39  |   1    38  |   2    38  |   2    39  |   1
    4     MM_Improved   37  |   3    35  |   5    33  |   7    38  |   2
    5       AB_Open     24  |  16    18  |  22    22  |  18    20  |  20
    6      AB_Center    24  |  16    26  |  14    18  |  22    24  |  16
    7     AB_Improved   23  |  17    21  |  19    18  |  22    22  |  18
--------------------------------------------------------------------------
           Win Rate:      80.0%        77.9%        73.6%        76.1%

(aind) rosco@rosco-XPS-13-9360:~/Desktop/AIND-Isolation-master$
```

Figure 3. Tournament.py results. 40 Matches and 250 ms Time limit.