Heuristic Functions:

For this analysis, the three heurestic functions tested are as follows:

1. Custom_score:

$$when\ move_{count} < 10 : \frac{(h-a)^2 + (w-b)^2}{(h-y)^2 + (w-x)^2}$$

$$when\ move_{count} \geq 10 : \left(\frac{own_{moves}}{opponent_{moves}}\right)$$

   a. (h,w) are the board center's coordinates,
   b. (x,y) is the location of our CPU player
   c. (a,b) is the location of the opponent
   d. Move_count is the number of moves played, as tracked by game class
   e. Own_moves is the number of available moves for player
   f. Opponent_moves is the number of available moves for the opponent

First part of the heuristic is used when the number of moves played is less than 10. It results in the player trying to maximize its own distance from the center, while trying to keep the opponent close to the center. The second part mainly reiles on maximizing the ratio between the player's legal moves and opponent's legal moves. The heuristics are combined such that during the early game a zonal strategy is used. In the mid and late game the own_moves: opponent_moves calculation is used.

The second part of the heuristic is slightly different than Improved_score function as it rewards minimizing the opponent moves more. Consider the scenario below (only for visualization, may not be accurate) where node_1 has player_moves = 5 and opponent_moves = 2; and node_2 has player_moves = 3 and opponent_moves = 1. The Improved_score heuristic will chose node_1 since the difference between the player and opponent moves is greater. However, Custom_score will choose node_2 since the ratio 3:1 is greater than 5:2.

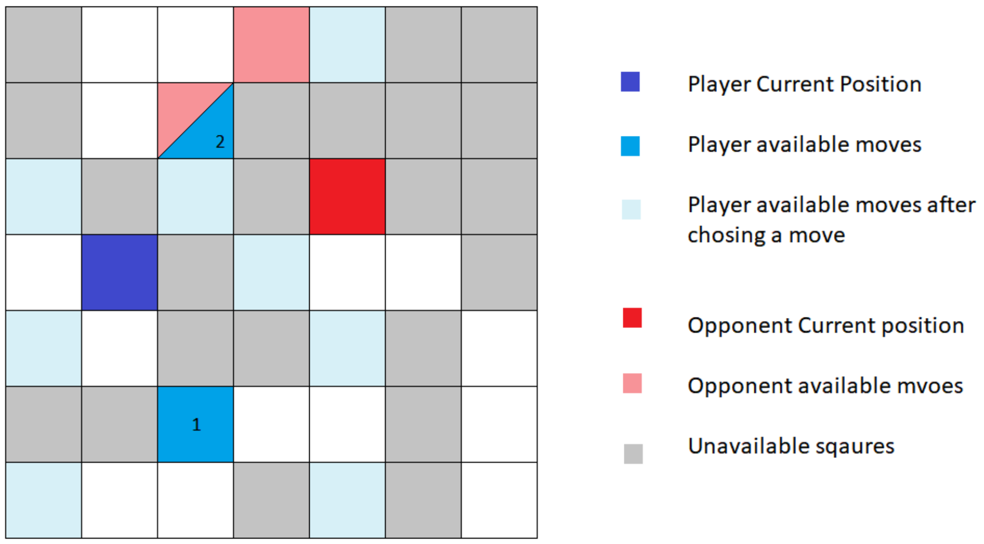


Figure 1: Own moves / Opponent moves scenario

2. Custom_score_2:

$$\frac{(h-a)^2+(w-b)^2}{(h-y)^2+(w-x)^2}$$

This function is similar to the first part of the custom_score.

3. Custom_score_3:

$$\frac{\sum_{(i,j)=blank\_spaces[0]}^{blank\_spaces[n]} (a-i)^2+(b-j)^2}{\sum_{(i,j)=blank\_spaces[0]}^{blank\_spaces[n]} (x-i)^2+(y-j)^2}$$

This heuristic calculates the average distance of the player and opponent from all blank spaces currently left on the board. This will guide the player to try to minimize its distance from the center of blank_spaces and try to maximize the opponent’s distance from the center of blank_spaces. It is similar to the custom_score_2 function but instead of the board center it uses the blank spaces center, which changes as the game progresses.

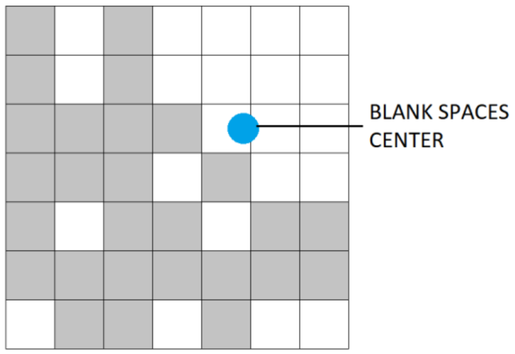
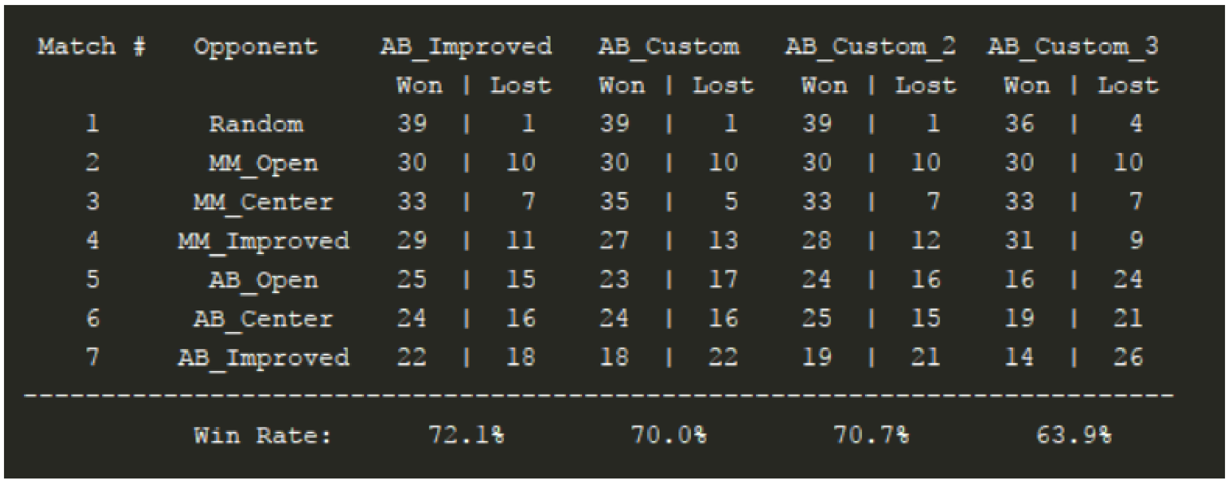


Figure 2: Custom_Score_3 function visualization

Results:

To improve our analysis the number of matches was increased from 5 to 20.

Table 1: Tournament.py results (# matches = 30)



```
 Match #    Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                         Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1        Random      39  |   1    39  |   1    39  |   1    36  |   4
    2        MM_Open     30  |  10    30  |  10    30  |  10    30  |  10
    3       MM_Center    33  |   7    35  |   5    33  |   7    33  |   7
    4      MM_Improved   29  |  11    27  |  13    28  |  12    31  |   9
    5        AB_Open     25  |  15    23  |  17    24  |  16    16  |  24
    6       AB_Center    24  |  16    24  |  16    25  |  15    19  |  21
    7      AB_Improved   22  |  18    18  |  22    19  |  21    14  |  26
--------------------------------------------------------------------------
          Win Rate:      72.1%        70.0%        70.7%        63.9%
```

In the results, we see that our custom_score heuristic could not outperform the Improved_score heuristic but only with a very small margin. It would not be advisable to suggest that either of the heuristic was significantly better. Custom_score_2 was similar to custom_score in win rate. Custom_score_3 was worse off compared to the other 2 custom heuristics and Improved_score.

Alpha-Beta pruning seems to provide a huge advantage when designing an isolation player. All functions did much better against the test agents when using Alpha-Beta pruning, compared to using only the minimax algorithm.

Recommendation:

Looking at the data above the Improved_score heuristic should be recommended for use, due to the following reasons:

1. The win rate for this function is the highest
2. The function did better against other functions that were also using Alpha-Beta Pruning
3. It is a simple heuristic and hence would allow the CPU player to reach deeper levels more quickly, than if the function was more complex and thus required more time for computation