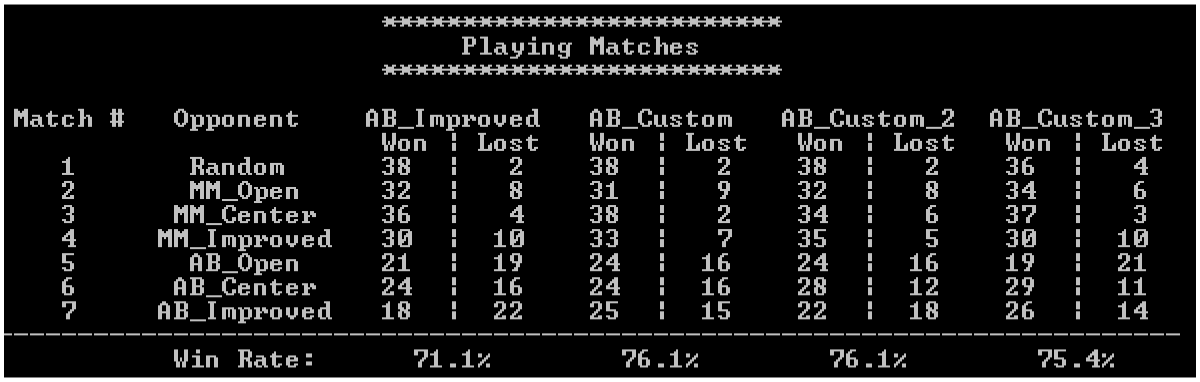
Custom heuristic functions by Rohit Patel

The three custom heuristic functions produced the following result, each doing better than the AB Improved algorithm:



```
                          *************************
                               Playing Matches
                          *************************

Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                       Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1      Random      38  |   2    38  |   2    38  |   2    36  |   4
    2      MM_Open     32  |   8    31  |   9    32  |   8    34  |   6
    3     MM_Center    36  |   4    38  |   2    34  |   6    37  |   3
    4    MM_Improved   30  |  10    33  |   7    35  |   5    30  |  10
    5      AB_Open     21  |  19    24  |  16    24  |  16    19  |  21
    6     AB_Center    24  |  16    24  |  16    28  |  12    29  |  11
    7    AB_Improved   18  |  22    25  |  15    22  |  18    26  |  14
--------------------------------------------------------------------------
          Win Rate:      71.1%        76.1%        76.1%        75.4%
```

custom_score: This score was based on the number of moves available three steps down the line from any game state for each of the players. First, we calculate the available moves from the current position for the player, then all available moves from each of the available moves, and so on for three steps. We count these moves (duplicates are counted more than once), this is called the own_score. Similarly, we calculate the opponent score: all the cells available to the opponent three steps down the line. Please note that this is an imperfect measure, since in reality those moves may not be available to the player three steps down the line, depending on the gameplay. We achieve an improvement over the AB_Improved player. This is despite the fact that the calculation of the three steps is more computationally intensive than not looking forward. However, this is perhaps due to the reason that the algorithm avoids a scenario where many undesirable moves are available to a player in the next move, but none are good.

custom_score_2: This is a modification of the custom_score function with normalized values of the scoring vector (the vector to calculate moves two steps ahead). This vector is essentially a score for each of the cells and how good the prospects are from there going forward. The heuristic adds the scores of self and the opponent and takes a difference. The heuristic does not perform as well as the customer score, however, performs better than the AB_Improved. The deteriorated performance is likely due to the extra normalization step, that appears to add no value.

custom_score_3: This function takes a similar approach as the above two, however, instead of calculating the number of possible moves three steps out, it calculates the number of cells accessible from a position three steps out. If there are multiple ways to reach a cell, they are not counted. This essentially follows the idea that if the board is to be split in partitions, it is worthwhile to be in the bigger of the two partitions. This heuristic also performs better than the improved_score, however, worse than the other two players introduced earlier.

We recommend using the custom_score evaluation function for the following reasons:

1. It remains simple to calculate (only marginally more computationally intensive than improved_score) and is efficient when search is performed in a computation and time bound manner
2. While the improved_score heuristic might be able to search tree to a deeper level due to its simplicity. The custom_score heuristi but considers information 3 moves ahead (albeit imperfectly), and therefore has somewhat of an impact of searching the game tree deeper than the time has allowed
3. It performs better than any of the other more complex heuristics tested, and beats the improved_score heuristic consistently. This suggests that using more complex heuristics, while potentially accurate at a certain depth, do not allow game to be searched deeper when the search is time limited, due to the computational complexity. The custom_score heuristic balances the tradeoff and provides an improvement

Note: Running some tests on the depth of the tree searched with improved_score and custom_score, it appears as though the improved_score consistently searches the tree deeper, sometimes with a difference of more than one depth. However, the custom_score is able to make up for it with the ability to see 3 moves ahead.