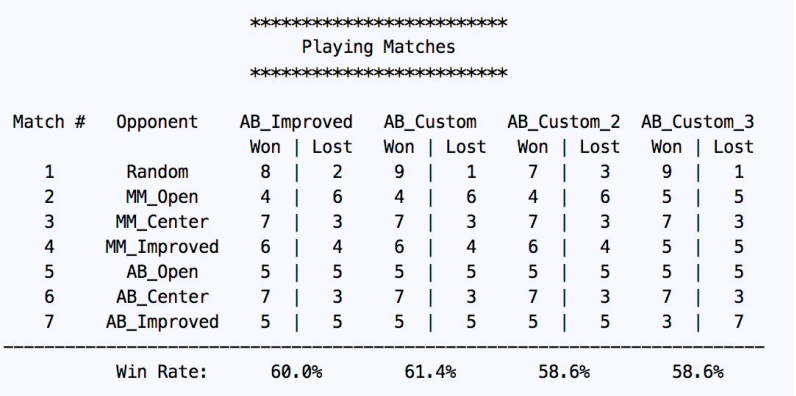
Heuristic Analysis

Matching Results(1)



```
*************************
    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     7  |  3     9  |  1
   2       MM_Open       4  |  6     4  |  6     4  |  6     5  |  5
   3      MM_Center      7  |  3     7  |  3     7  |  3     7  |  3
   4     MM_Improved     6  |  4     6  |  4     6  |  4     5  |  5
   5       AB_Open       5  |  5     5  |  5     5  |  5     5  |  5
   6      AB_Center      7  |  3     7  |  3     7  |  3     7  |  3
   7     AB_Improved     5  |  5     5  |  5     5  |  5     3  |  7
--------------------------------------------------------------------------
           Win Rate:      60.0%        61.4%        58.6%        58.6%
```

Heuristic Functions(1)

Heuristic	AB_Custom	AB_Custom2	AB_Custom3
Player is more likely to win in a game when..	As Custom2, but choose move where opponent has no available moves if possible	There are more moves available for player than opponent's moves	There are more open moves for the player
Formula	as Custom_2, but score * 2 when #legal_moves(opponent) = 0	#legal_moves(player) - #legal_moves(opponent)	#legal_moves()

I have tried some other heuristic strategies like "minimizing opponent's future moves after player move", but all resulted in bad result like ~50% win rate. I found the idea of trying to choose moves which maximize player's available future moves to be a both simple and effective evaluation metric. (58.6% win rate for AB_Custom3)

Further, while trying to maximize the player's moves, I think it may help to also try to minimize the opponent's move at the same time. (AB_Custom2). Although in the tournament above it showed no difference with AB_Custom3.

Finally, slightly modified from AB_Custom2, I make heuristic score twice bigger when there is no available moves for opponent. The intuition is that for two different game state A and game state B, even if the difference between #legal_moves(player) and #legal_moves(opponent) is the same, our player is better off to choose game state B if #legal_moves(opponent) in game stateB is 0, thus the player win immediately.

This result in a slight, though not significantly, improvement on win rate for AB_Custom (61.4%). Thus I will use AB_Custom for my final evaluation function.

[Updated Improvement]

As shown above, since the general heuristic function "#legal_moves(player) - #legal_moves(opponent)" perform well, I made some modification to the function to see whether the win rate can be further improved.

We can reformule the formula
"#legal_moves(player) - #legal_moves(opponent)"
as
"w1 * #legal_moves(player) - w2 * #legal_moves(opponent)" where w1 = w2 = 1.

For AB_Cusom, I leave w1 and w2 unchanged, thus the function is equal to AB_Improved.

For AB_Custom2, I set w1 = 10 and leave w2 unchanged, thus the function reward game states when there are more available moves for the computer player in the game.

For AB_Custom3, I set w2 = 10 and leave w1 unchanged, thus the function penalize game states where there are more moves for the opponent.

Matching Results(2)

```
                    *************************
                        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    10  |   0     7  |   3
   2       MM_Open      5  |   5     5  |   5     6  |   4     3  |   7
   3      MM_Center     8  |   2     8  |   2     8  |   2    10  |   0
   4     MM_Improved    3  |   7     3  |   7     4  |   6     6  |   4
   5       AB_Open      5  |   5     5  |   5     4  |   6     4  |   6
   6      AB_Center     6  |   4     6  |   4     6  |   4     5  |   5
   7     AB_Improved    5  |   5     5  |   5     6  |   4     6  |   4
--------------------------------------------------------------------------
          Win Rate:      60.0%        57.1%        62.9%        58.6%
```

As shown above, both AB_Custom_2 and AB_Custom_3 outperformed the original AB_improve function and AB_Custom_2 got the highest win rate for all the heuristic functions I had tried. (62.9%) thus I will use this AB_Custom_2 as my final function.