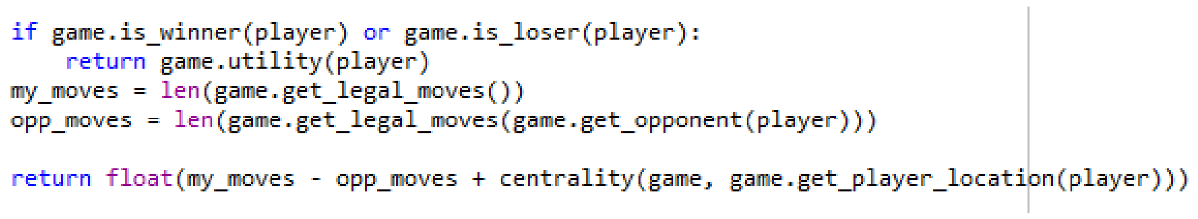
Heuristic analysis

Loay Ashraf

custom_score (heuristic #1)

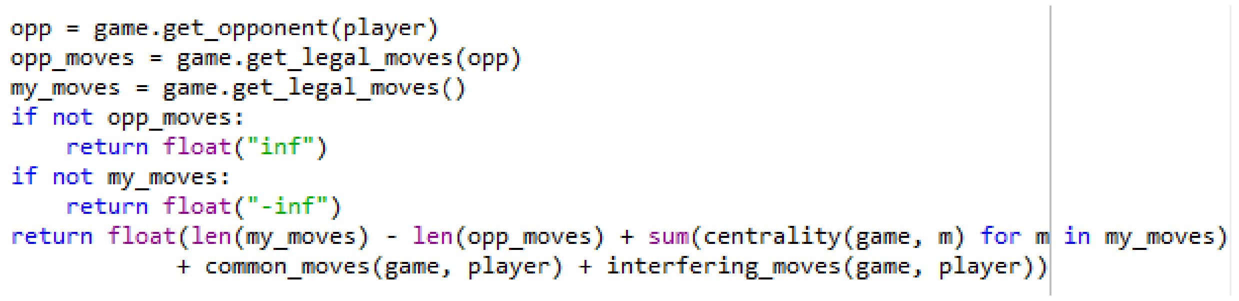


```python
if game.is_winner(player) or game.is_loser(player):
    return game.utility(player)
my_moves = len(game.get_legal_moves())
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))

return float(my_moves - opp_moves + centrality(game, game.get_player_location(player)))
```

The first heuristic returns the number of available moves for the player with a “centrality” value for the move to be made. It can be considered a defensive heuristic because it takes in consideration how near or how far the player will be from the center of the board given that being at the center of the board gives the player the most options for the coming move.

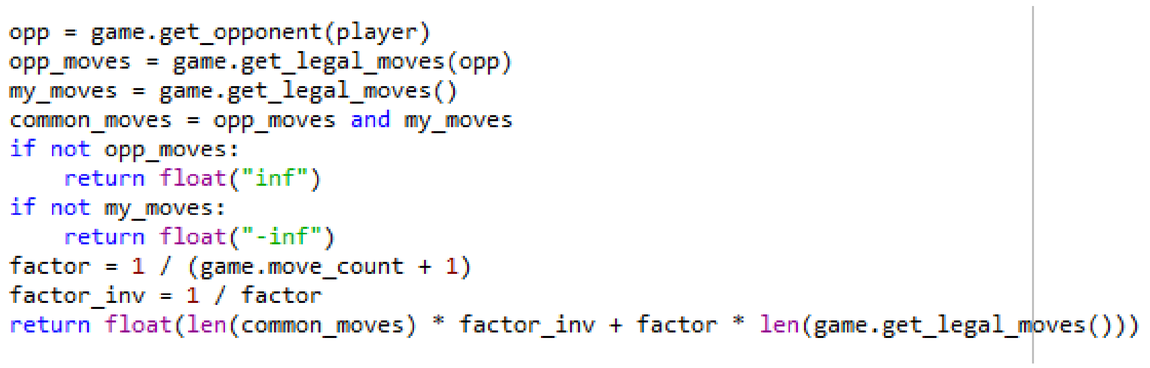
custom_score_2 (heuristic #2)



```python
opp = game.get_opponent(player)
opp_moves = game.get_legal_moves(opp)
my_moves = game.get_legal_moves()
if not opp_moves:
    return float("inf")
if not my_moves:
    return float("-inf")
return float(len(my_moves) - len(opp_moves) + sum(centrality(game, m) for m in my_moves)
            + common_moves(game, player) + interfering_moves(game, player))
```

The second heuristic takes a lot of game state elements into consideration which include the total score of “centrality” of all of the possible moves added to the common moves between player and opponent and finally the “interfering” moves which are the move(s) which has highest centrality score and also common for both players to take (we might be able to steal a very good move from the opponent), as this heuristic takes account for all of those factors it can be considered a balanced heuristic with a bit of defense and also offense.

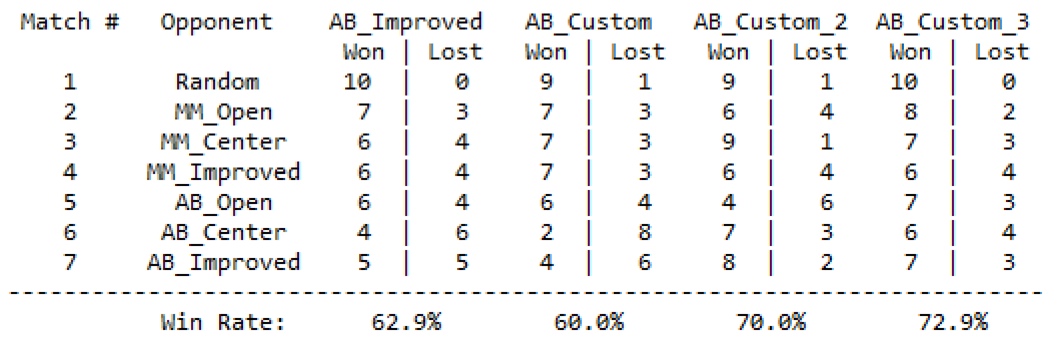
custom_score_3 (heuristic #3)



```
opp = game.get_opponent(player)
opp_moves = game.get_legal_moves(opp)
my_moves = game.get_legal_moves()
common_moves = opp_moves and my_moves
if not opp_moves:
    return float("inf")
if not my_moves:
    return float("-inf")
factor = 1 / (game.move_count + 1)
factor_inv = 1 / factor
return float(len(common_moves) * factor_inv + factor * len(game.get_legal_moves()))
```

The third heuristic's importance arises at the end of the game, usually as the game nears its end the number of available moves decreases significantly, as each player may have 2 or 3 legal moves at best so number of legal move becomes less and less important and what becomes more important is the number of common moves which gives the player better chance of cutting off the number of legal moves for the opponent, so this heuristic is considered an offensive one.

Results



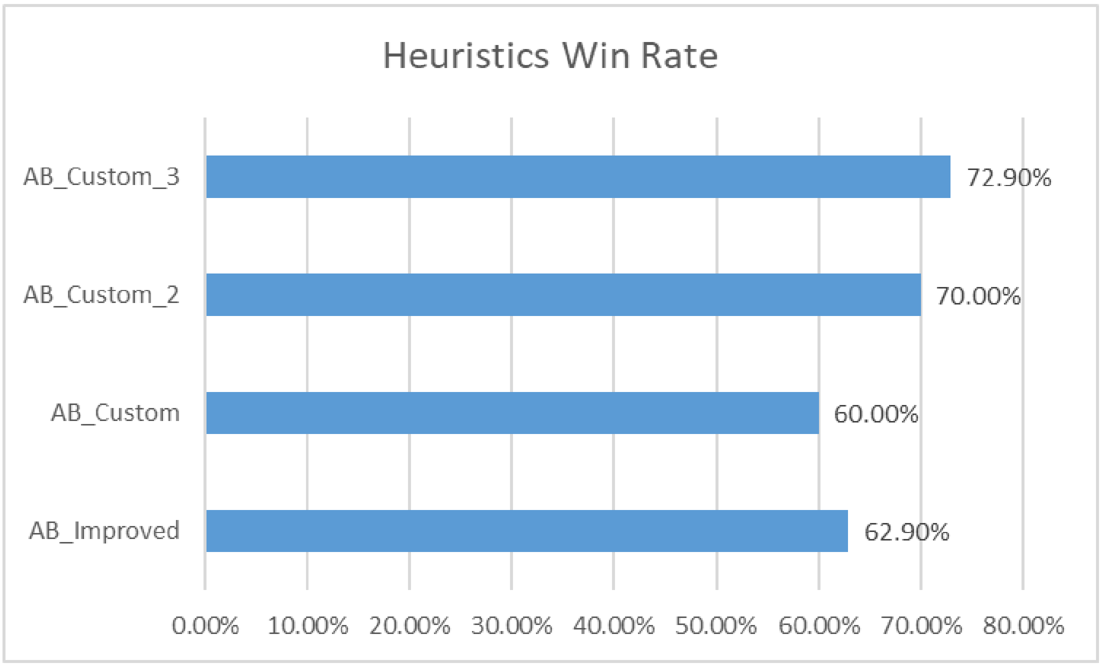
```
 Match #   Opponent     AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                         Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       10  |   0     9  |   1     9  |   1    10  |   0
    2       MM_Open       7  |   3     7  |   3     6  |   4     8  |   2
    3      MM_Center      6  |   4     7  |   3     9  |   1     7  |   3
    4     MM_Improved     6  |   4     7  |   3     6  |   4     6  |   4
    5       AB_Open       6  |   4     6  |   4     4  |   6     7  |   3
    6      AB_Center      4  |   6     2  |   8     7  |   3     6  |   4
    7     AB_Improved     5  |   5     4  |   6     8  |   2     7  |   3
--------------------------------------------------------------------------
           Win Rate:      62.9%        60.0%        70.0%        72.9%
```

For the tournament results the third heuristic comes with the highest winning rate of 72.9% the other two had 60.0% and 70.0% win rates respectively and as we can see all the heuristics (including AB_Improved heuristic) were superior against a random player but as the matches go on against MM and AB heuristics win rates begin to vary and the worst case was for the first heuristic against AB_Center heuristic as it won only two matches, while the win rates for the second and the third heuristics were very similar.

Heuristic Recommendation

My **recommendation** for the heuristic to use would be the third heuristic and not only because of the high win rate but also because of the fact that it surpassed the AB_Improved heuristic and also has the least demand for computation resources among the three heuristics by cutting down the time spent searching the tree in great depths as it ignores part of the possible legal moves and focuses on the common moves between the two players which is usually less in number, and also being aggressive near the end of the game is a very good strategy.

Here’s a comparison chart for the final results:


Heuristics Win Rate
AB_Custom_3
AB_Custom_2
AB_Custom
AB_Improved