HEURISTIC ANALYSIS

Isolation Game Playing Agent

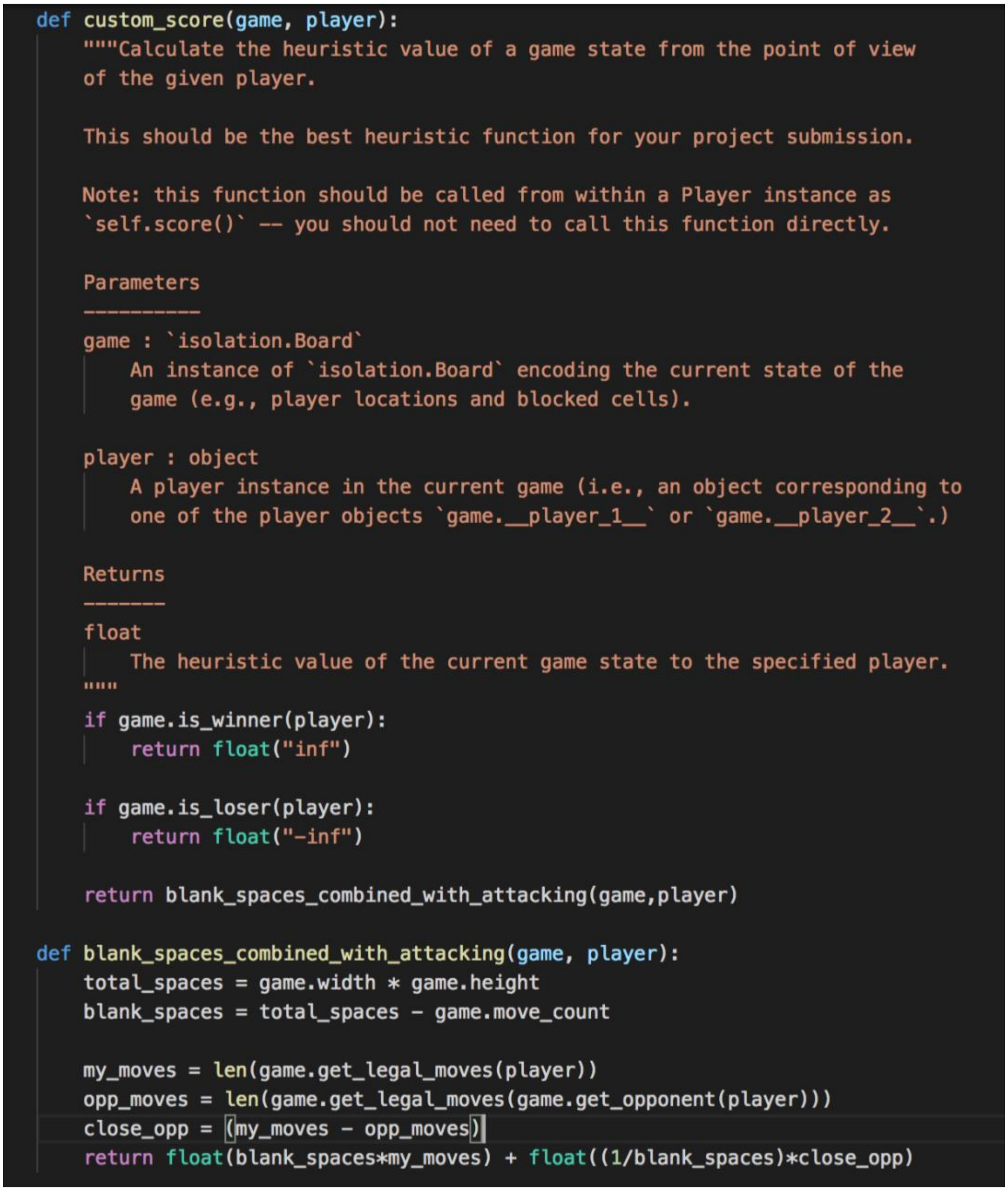
Elikem Kuivi

Heuristic Analysis

Heuristic 1: Combination of open_moves and advantage_score

This function uses a combination of the open moves heuristic and the advantage score heuristic with weights applied. The idea is to create a function that moves its focus from finding the spot with the most available moves to finding the spot that cuts off the opponent's moves, as the game progress.

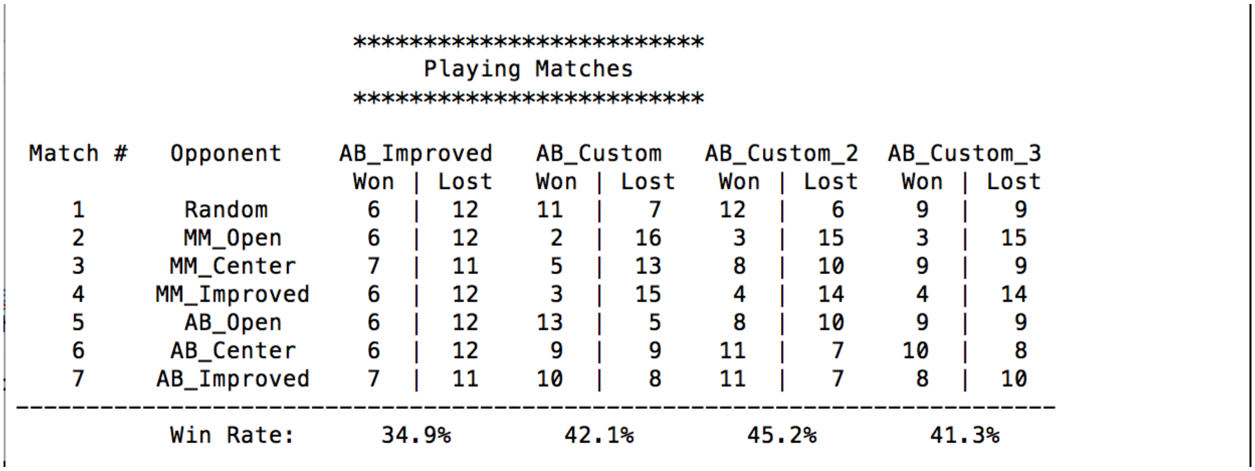
Code:



```
def custom_score(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player.

    This should be the best heuristic function for your project submission.

    Note: this function should be called from within a Player instance as
    `self.score()` -- you should not need to call this function directly.

    Parameters
    ----------
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).

    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)

    Returns
    -------
    float
        The heuristic value of the current game state to the specified player.
    """
    if game.is_winner(player):
        return float("inf")

    if game.is_loser(player):
        return float("-inf")

    return blank_spaces_combined_with_attacking(game,player)

def blank_spaces_combined_with_attacking(game, player):
    total_spaces = game.width * game.height
    blank_spaces = total_spaces - game.move_count

    my_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
    close_opp = (my_moves - opp_moves)
    return float(blank_spaces*my_moves) + float((1/blank_spaces)*close_opp)
```

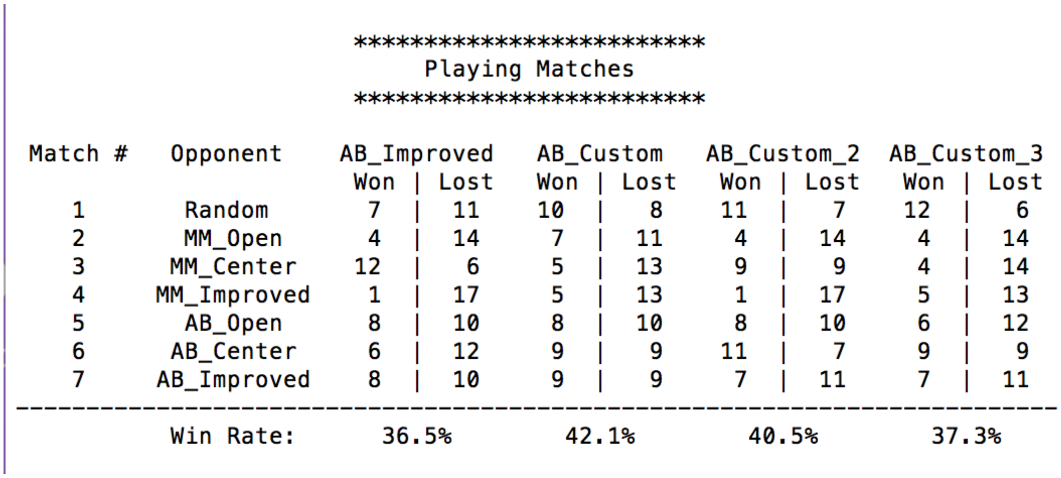
Results:

Game Series 1



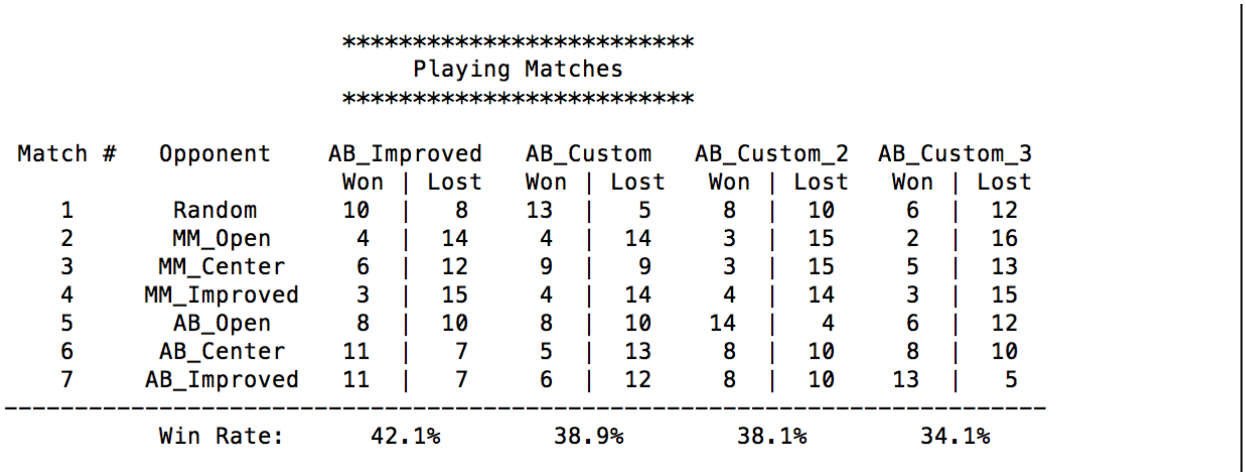
```
                        *************************
                             Playing Matches
                        *************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       6  |  12    11  |   7    12  |   6     9  |   9
    2       MM_Open      6  |  12     2  |  16     3  |  15     3  |  15
    3      MM_Center     7  |  11     5  |  13     8  |  10     9  |   9
    4     MM_Improved    6  |  12     3  |  15     4  |  14     4  |  14
    5       AB_Open      6  |  12    13  |   5     8  |  10     9  |   9
    6      AB_Center     6  |  12     9  |   9    11  |   7    10  |   8
    7     AB_Improved    7  |  11    10  |   8    11  |   7     8  |  10
--------------------------------------------------------------------------
           Win Rate:      34.9%        42.1%        45.2%        41.3%
```

Game Series 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       7  |  11    10  |   8    11  |   7    12  |   6
    2       MM_Open      4  |  14     7  |  11     4  |  14     4  |  14
    3      MM_Center    12  |   6     5  |  13     9  |   9     4  |  14
    4     MM_Improved    1  |  17     5  |  13     1  |  17     5  |  13
    5       AB_Open      8  |  10     8  |  10     8  |  10     6  |  12
    6      AB_Center     6  |  12     9  |   9    11  |   7     9  |   9
    7     AB_Improved    8  |  10     9  |   9     7  |  11     7  |  11
--------------------------------------------------------------------------
           Win Rate:      36.5%        42.1%        40.5%        37.3%
```

Game Series 3



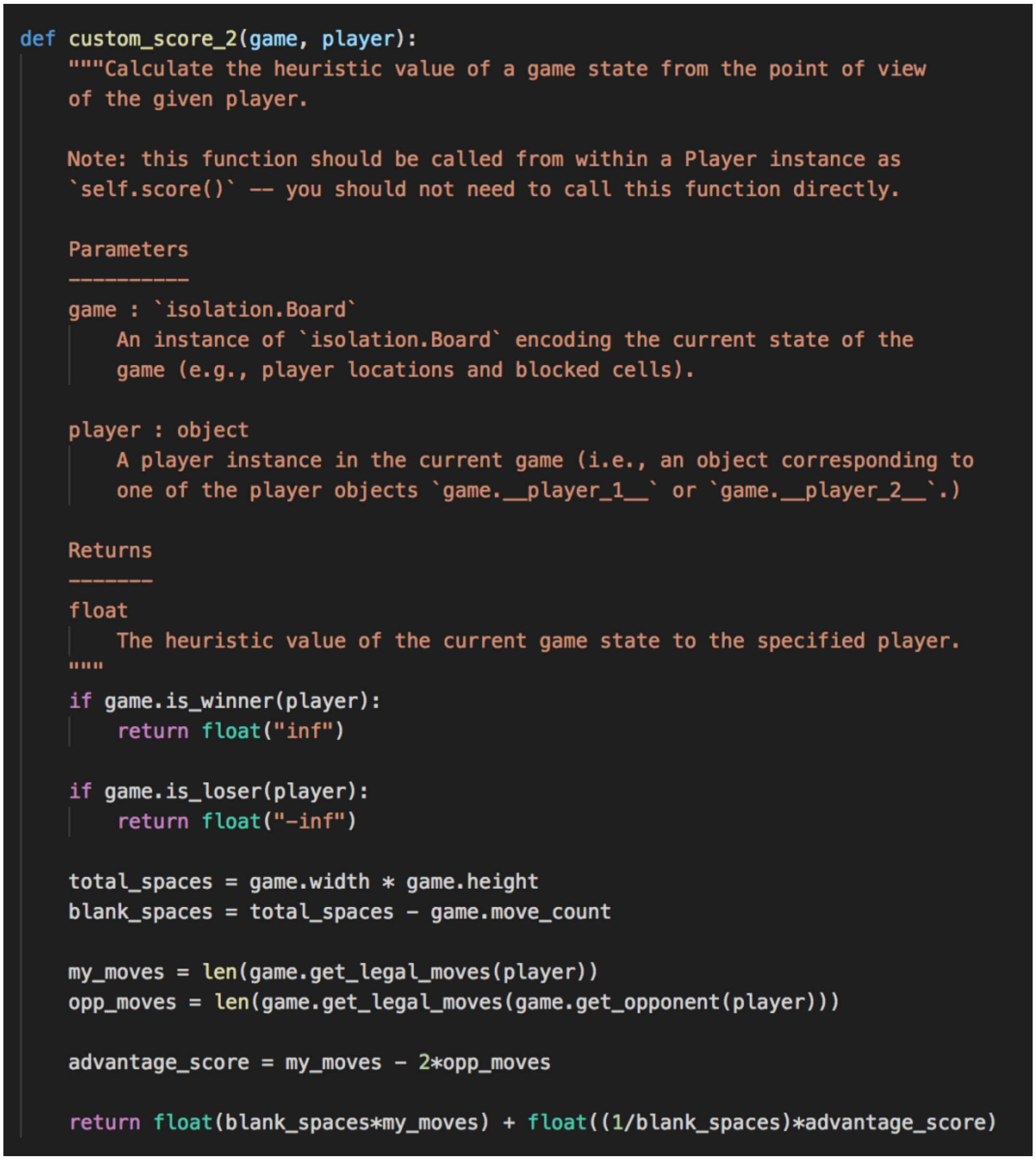
```
                        *************************
                             Playing Matches
                        *************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      10  |   8    13  |   5     8  |  10     6  |  12
    2       MM_Open      4  |  14     4  |  14     3  |  15     2  |  16
    3      MM_Center     6  |  12     9  |   9     3  |  15     5  |  13
    4     MM_Improved    3  |  15     4  |  14     4  |  14     3  |  15
    5       AB_Open      8  |  10     8  |  10    14  |   4     6  |  12
    6      AB_Center    11  |   7     5  |  13     8  |  10     8  |  10
    7     AB_Improved   11  |   7     6  |  12     8  |  10    13  |   5
--------------------------------------------------------------------------
           Win Rate:      42.1%        38.9%        38.1%        34.1%
```

Over the 3 game series, AB_Improved had a win rate of 37.84%, while this custom heuristic (AB_Custom) had an average win rate of 41.03%.

Heuristic 2: Combination of open_moves and aggressive advantage_score

This function is similar to the one above. The only difference is that in calculating the advantage score (difference between my available moves and opponent's moves), the opponent's moves are given more weight. Through this, the algorithm will select moves that are clearly more advantageous than an opponent's moves.

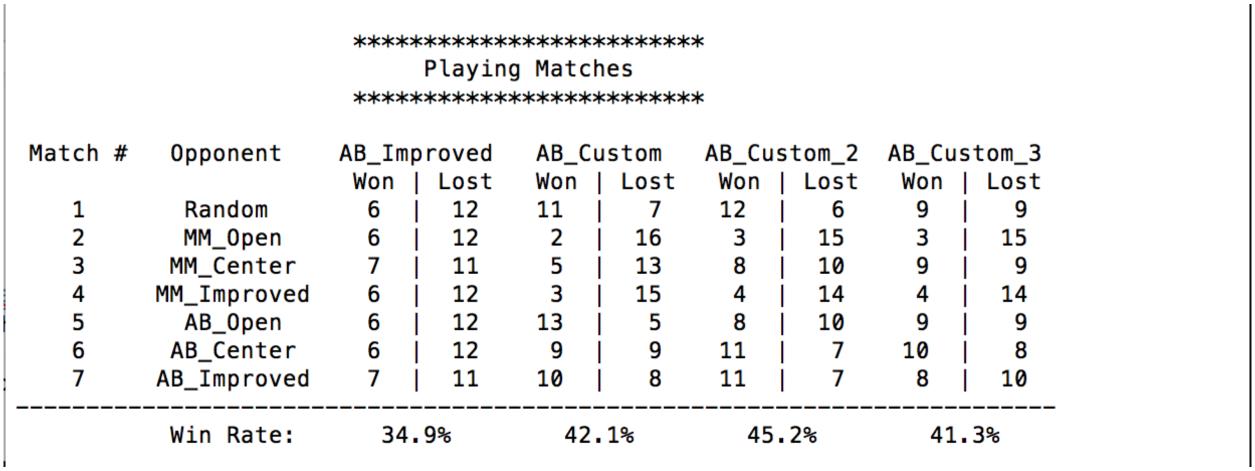
Code:



```
def custom_score_2(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player.

    Note: this function should be called from within a Player instance as
    `self.score()` -- you should not need to call this function directly.

    Parameters
    ----------
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).

    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)

    Returns
    -------
    float
        The heuristic value of the current game state to the specified player.
    """
    if game.is_winner(player):
        return float("inf")

    if game.is_loser(player):
        return float("-inf")

    total_spaces = game.width * game.height
    blank_spaces = total_spaces - game.move_count

    my_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))

    advantage_score = my_moves - 2*opp_moves

    return float(blank_spaces*my_moves) + float((1/blank_spaces)*advantage_score)
```

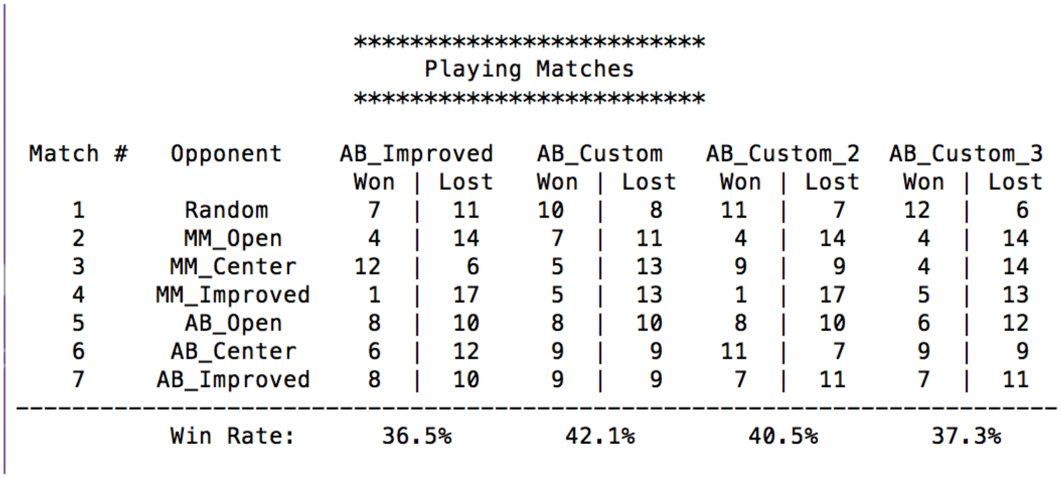
Results:

Game Series 1



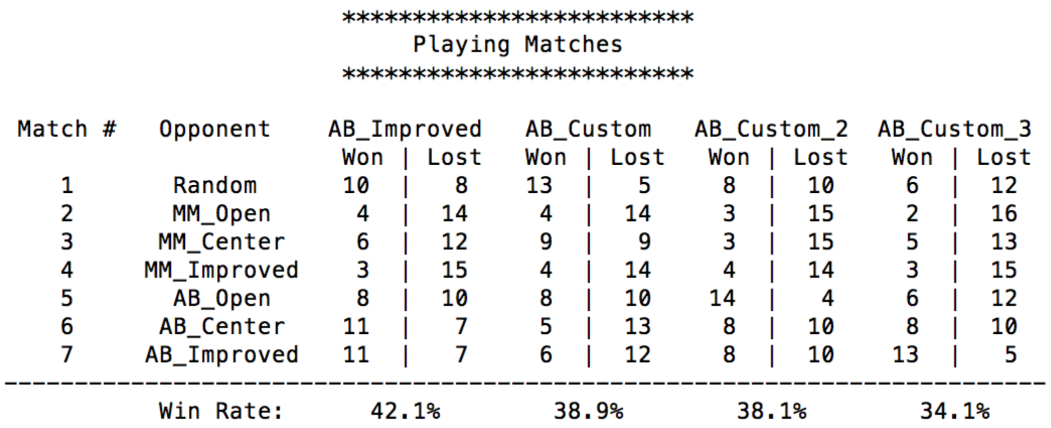
```
                          *************************
                               Playing Matches
                          *************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       6  |  12    11  |   7    12  |   6     9  |   9
    2       MM_Open      6  |  12     2  |  16     3  |  15     3  |  15
    3      MM_Center     7  |  11     5  |  13     8  |  10     9  |   9
    4     MM_Improved    6  |  12     3  |  15     4  |  14     4  |  14
    5       AB_Open      6  |  12    13  |   5     8  |  10     9  |   9
    6      AB_Center     6  |  12     9  |   9    11  |   7    10  |   8
    7     AB_Improved    7  |  11    10  |   8    11  |   7     8  |  10
--------------------------------------------------------------------------
           Win Rate:      34.9%        42.1%        45.2%        41.3%
```

Game Series 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       7  |  11    10  |   8    11  |   7    12  |   6
    2       MM_Open      4  |  14     7  |  11     4  |  14     4  |  14
    3      MM_Center    12  |   6     5  |  13     9  |   9     4  |  14
    4     MM_Improved    1  |  17     5  |  13     1  |  17     5  |  13
    5       AB_Open      8  |  10     8  |  10     8  |  10     6  |  12
    6      AB_Center     6  |  12     9  |   9    11  |   7     9  |   9
    7     AB_Improved    8  |  10     9  |   9     7  |  11     7  |  11
--------------------------------------------------------------------------
           Win Rate:      36.5%        42.1%        40.5%        37.3%
```

Game Series 3



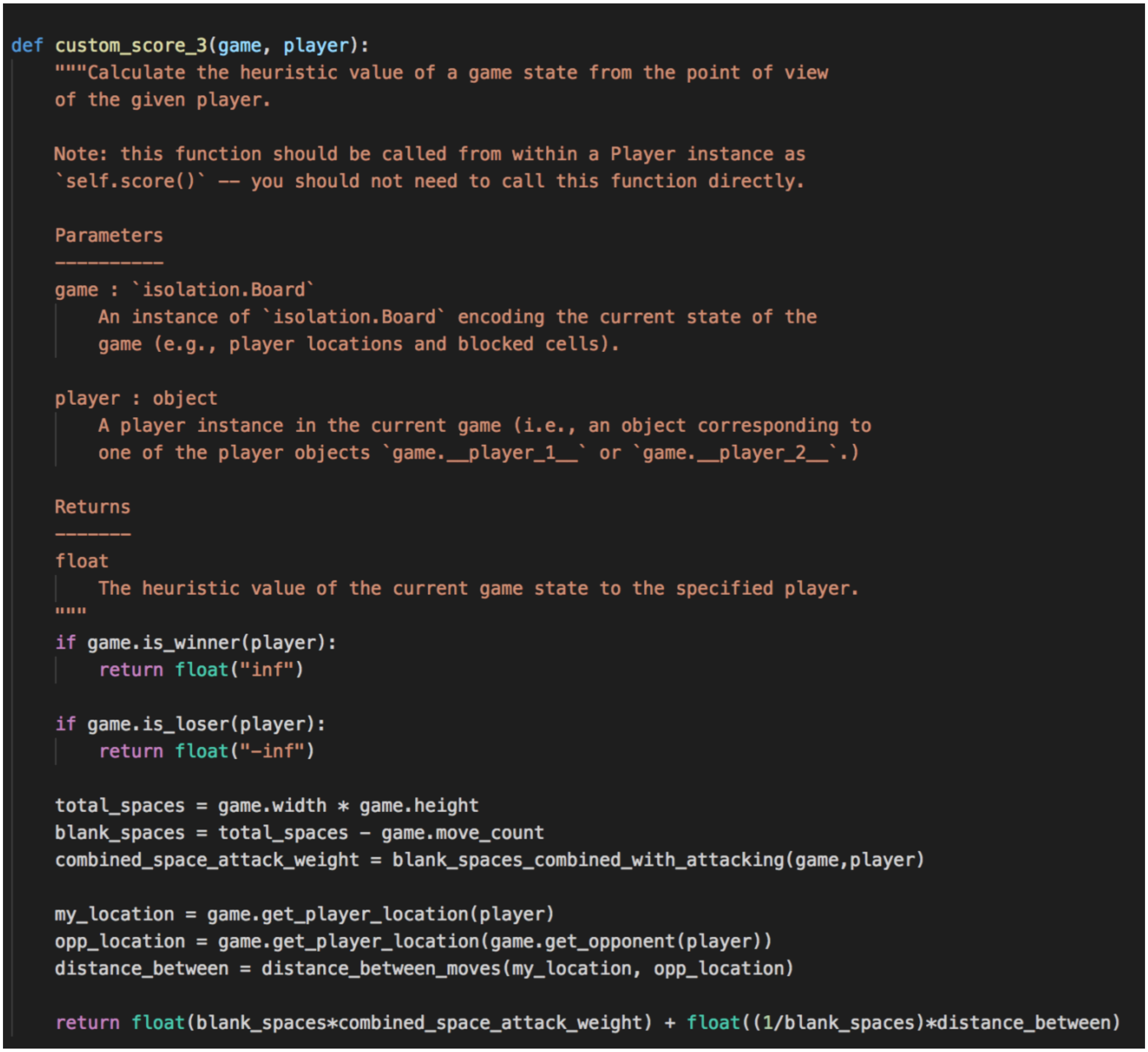
```
                                   *************************
                                        Playing Matches
                                   *************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      10  |   8    13  |   5     8  |  10     6  |  12
    2       MM_Open      4  |  14     4  |  14     3  |  15     2  |  16
    3      MM_Center     6  |  12     9  |   9     3  |  15     5  |  13
    4     MM_Improved    3  |  15     4  |  14     4  |  14     3  |  15
    5       AB_Open      8  |  10     8  |  10    14  |   4     6  |  12
    6      AB_Center    11  |   7     5  |  13     8  |  10     8  |  10
    7     AB_Improved   11  |   7     6  |  12     8  |  10    13  |   5
--------------------------------------------------------------------------
           Win Rate:      42.1%        38.9%        38.1%        34.1%
```

Over the 3 game series, AB_Improved had a win rate of 37.84%, while this custom heuristic (AB_Custom_2) had an average win rate of 41.27%.

Heuristic 3: Combined open_ moves and advantage score with proximity score

The idea here is to use a combination of the open moves and advantage score heuristic early in the game. However, as it gets closer to end game, the agent focuses on getting closer to its opponent. The assumption behind being close to the opponent is that it will close the opponent's moves, and back the opponent off; thus restricting the opponent's mobility.

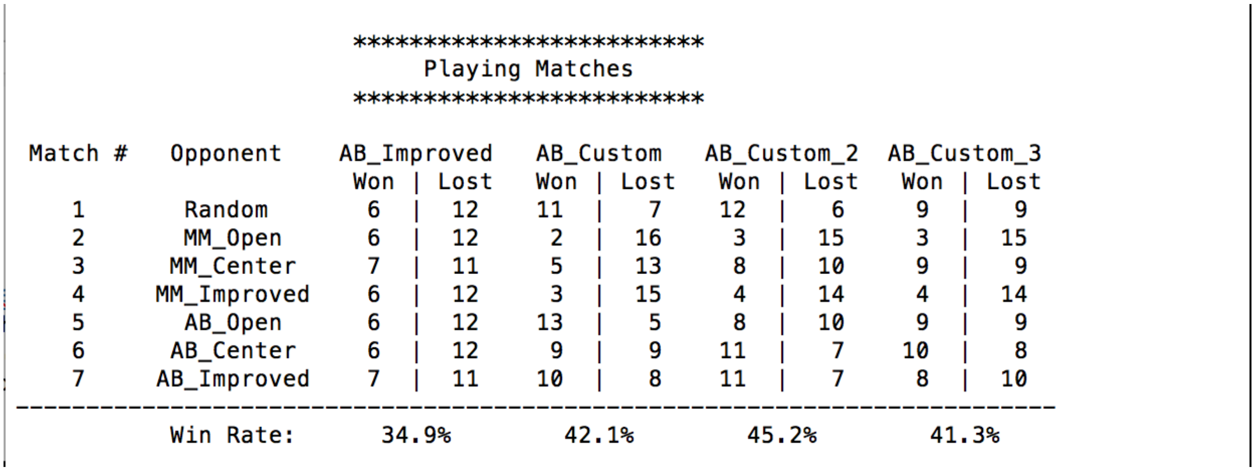
Code:



```python
def custom_score_3(game, player):
    """Calculate the heuristic value of a game state from the point of view
    of the given player.

    Note: this function should be called from within a Player instance as
    `self.score()` -- you should not need to call this function directly.

    Parameters
    ----------
    game : `isolation.Board`
        An instance of `isolation.Board` encoding the current state of the
        game (e.g., player locations and blocked cells).

    player : object
        A player instance in the current game (i.e., an object corresponding to
        one of the player objects `game.__player_1__` or `game.__player_2__`.)

    Returns
    -------
    float
        The heuristic value of the current game state to the specified player.
    """
    if game.is_winner(player):
        return float("inf")

    if game.is_loser(player):
        return float("-inf")

    total_spaces = game.width * game.height
    blank_spaces = total_spaces - game.move_count
    combined_space_attack_weight = blank_spaces_combined_with_attacking(game,player)

    my_location = game.get_player_location(player)
    opp_location = game.get_player_location(game.get_opponent(player))
    distance_between = distance_between_moves(my_location, opp_location)

    return float(blank_spaces*combined_space_attack_weight) + float((1/blank_spaces)*distance_between)
```

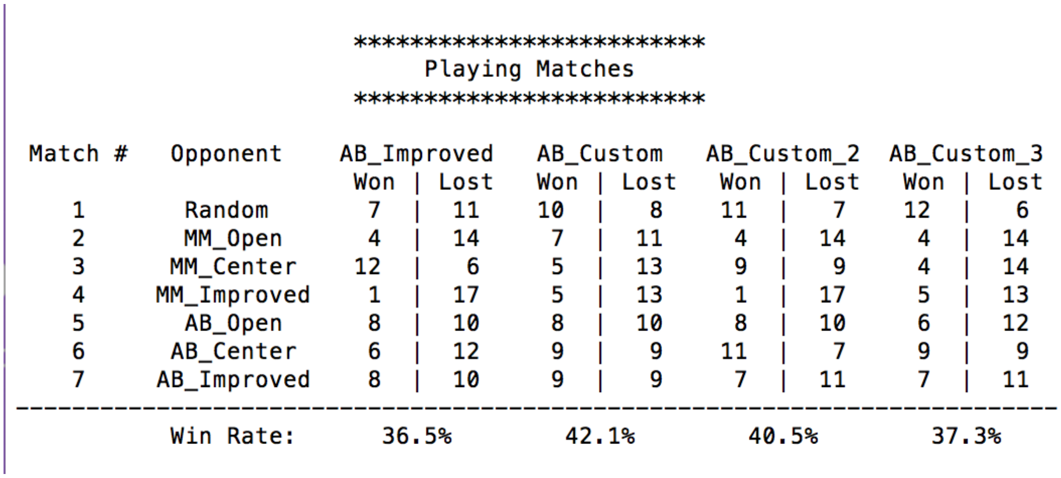
Results:

Game Series 1



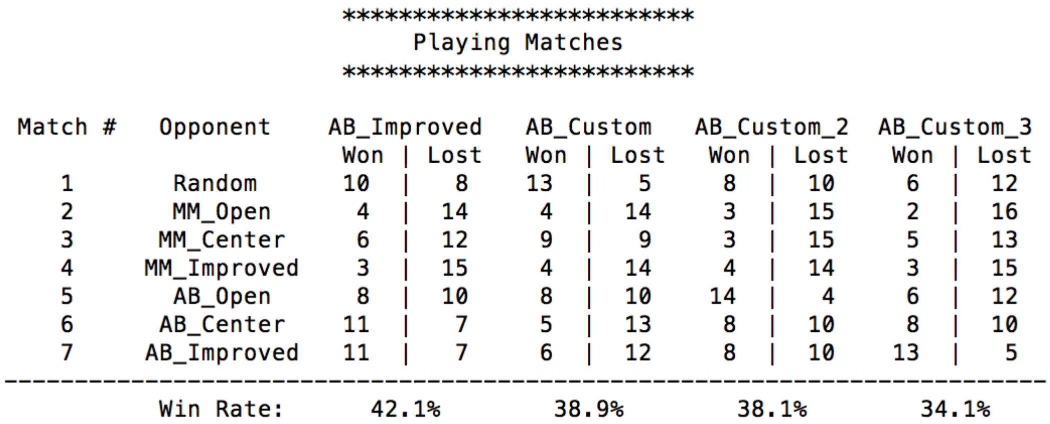
```
                         *************************
                              Playing Matches
                         *************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       6  |  12    11  |   7    12  |   6     9  |   9
    2       MM_Open      6  |  12     2  |  16     3  |  15     3  |  15
    3      MM_Center     7  |  11     5  |  13     8  |  10     9  |   9
    4     MM_Improved    6  |  12     3  |  15     4  |  14     4  |  14
    5       AB_Open      6  |  12    13  |   5     8  |  10     9  |   9
    6      AB_Center     6  |  12     9  |   9    11  |   7    10  |   8
    7     AB_Improved    7  |  11    10  |   8    11  |   7     8  |  10
--------------------------------------------------------------------------
           Win Rate:      34.9%         42.1%        45.2%        41.3%
```

Game Series 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       7  |  11    10  |   8    11  |   7    12  |   6
    2       MM_Open      4  |  14     7  |  11     4  |  14     4  |  14
    3      MM_Center    12  |   6     5  |  13     9  |   9     4  |  14
    4     MM_Improved    1  |  17     5  |  13     1  |  17     5  |  13
    5       AB_Open      8  |  10     8  |  10     8  |  10     6  |  12
    6      AB_Center     6  |  12     9  |   9    11  |   7     9  |   9
    7     AB_Improved    8  |  10     9  |   9     7  |  11     7  |  11
--------------------------------------------------------------------------
           Win Rate:      36.5%         42.1%        40.5%        37.3%
```

Game Series 3



```
                       *************************
                            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  |   8    13  |   5     8  |  10     6  |  12
    2       MM_Open      4  |  14     4  |  14     3  |  15     2  |  16
    3      MM_Center     6  |  12     9  |   9     3  |  15     5  |  13
    4     MM_Improved    3  |  15     4  |  14     4  |  14     3  |  15
    5       AB_Open      8  |  10     8  |  10    14  |   4     6  |  12
    6      AB_Center    11  |   7     5  |  13     8  |  10     8  |  10
    7     AB_Improved   11  |   7     6  |  12     8  |  10    13  |   5
--------------------------------------------------------------------------
           Win Rate:      42.1%        38.9%        38.1%        34.1%
```

Over the 3 game series, AB_Improved had a win rate of 37.84%, while this custom heuristic (Custom_Score_3) had an average win rate of 37.57%.

Summary

Overall, the first heuristic (combination of open moves and advantage score) and the second heuristic (combination of open moves and aggressive advantage score) had higher win rates than AB_Improved. However, between the first and second heuristic, the aggressive advantage score heuristic had a higher average win rate.

Before running my analysis to compare all 3 heuristics, I assumed the first heuristic, the less aggressive advantage score, would have the best performance since through manually playing games I found having a less aggressive approach to be more successful. However, in hindsight, my results from manual game playing are bound to be biased by both the skill level of my opponent and myself.

As a result, my final game playing agent will be using the second heuristic (combination of open moves and aggressive advantage score) as its best custom evaluation function.