Heuristic analysis

For this project, I try to create three different strategies to evaluate the better steps for computer players, and then mixed those strategies to find an average result.

Heuristic 1

The concept of first strategy is to calculate the difference between my total moves and opponent's total moves. However, we can set an additional condition for this strategy. If the game still has available moves more than 1/3 of total moves, the decision of better moves will be my moves twice greater than opponent's moves, which is stricter than the available moves less than 1/3. If the difference between my moves and opponent's move is higher than another, we can know that the next move is able to have higher chance to win.

In order to get more reasonable result, the difference of moves should include their position information. When the difference of moves is divided by relative position of mine and opponent, the better moves can be decided by the next move is close to each other.

```
    moves_own = len(game.get_legal_moves(player))
    moves_opp = len(game.get_legal_moves(game.get_opponent(player)))
    board = game.height * game.width
    moves_board = game.move_count / board
    if moves_board > 0.33:
        move_diff = (moves_own - moves_opp*2)
    else:
        move_diff = (moves_own - moves_opp)

    pos_own = game.get_player_location(player)
    pos_opp = game.get_player_location(game.get_opponent(player))

    m_distance = abs(pos_own[0] - pos_opp[0]) + abs(pos_own[1] - pos_opp[1])
```

Heuristic 2

I add a new condition called center difference in the move comparison to help to analyze the better move. This condition is to calculate the distance between center and the move we concerned. If the my move is as the same as opponent's moves, this condition can be used for a reference to decide which step is better. We can guess if the move is closer to center it should be considered a better move.

```
    pos_center = (int(game.height / 2), int(game.width / 2))
    player_distance = abs(pos_own[0] - pos_center[0]) + abs(pos_own[1] -
pos_center[1])
```

```
    opponent_distance = abs(pos_opp[0] - pos_center[0]) + abs(pos_opp[1] -
pos_center[1])
    center_diff = opponent_distance - player_distance

    if moves_board > 0.33:
        if moves_own == moves_opp:
            move_diff = moves_own - moves_opp*2 + center_diff
        else:
            move_diff = moves_own - moves_opp*2
    else:
        if moves_own == moves_opp:
            move_diff = moves_own - moves_opp + center_diff
        else:
            move_diff = moves_own - moves_opp

    return float(move_diff / m_distance)
```

Heuristic 3

The third heuristic includes another condition called common-set which is able to calculate the intersection between my moves and opponent's moves. I assume that the more common move between my and opponent's is a better because I have more opportunities to interfere opponent.

```
    common_set = len(set(pos_own).intersection(set(pos_opp)))

    if moves_board > 0.33:
        if moves_own == moves_opp:
            move_diff = moves_own - moves_opp*2 + center_diff + common_set
        else:
            move_diff = moves_own - moves_opp*2
    else:
        if moves_own == moves_opp:
            move_diff = moves_own - moves_opp + center_diff + common_set
        else:
            move_diff = moves_own - moves_opp

    return float(move_diff / m_distance)
```

Tournament Results

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

 Match #    Opponent    AB_Improved    AB_Custom    AB_Custom_2  AB_Custom_3
                         Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random        5  |   5     8  |   2     6  |   4     9  |   1
```

```
    2       MM_Open      7  |   3     7  |   3     7  |   3     9  |   1
    3      MM_Center     6  |   4     6  |   4     8  |   2     8  |   2
    4     MM_Improved    7  |   3     4  |   6     4  |   6     6  |   4
    5       AB_Open      5  |   5     5  |   5     3  |   7     7  |   3
    6      AB_Center     4  |   6     6  |   4     5  |   5     6  |   4
    7     AB_Improved    7  |   3     4  |   6     5  |   5     3  |   7
----------------------------------------------------------------------------
          Win Rate:      58.6%        57.1%        54.3%        68.6%
```

```
                    *************************
                         Playing Matches
                    *************************
┌
 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       9  |   1     7  |   3     9  |   1     9  |   1
    2       MM_Open      6  |   4     9  |   1     7  |   3     7  |   3
    3      MM_Center     9  |   1     8  |   2     8  |   2     7  |   3
    4     MM_Improved    5  |   5     8  |   2     6  |   4     7  |   3
    5       AB_Open      5  |   5     5  |   5     5  |   5     6  |   4
    6      AB_Center     5  |   5     4  |   6     5  |   5     6  |   4
    7     AB_Improved    5  |   5     6  |   4     6  |   4     5  |   5
----------------------------------------------------------------------------
          Win Rate:      62.9%        67.1%        65.7%        67.1%
```

For the match 3 and 6, the AB_Custom_2 sometime performs better result than AB_Improved and AB_Custom because it's score is influenced by the center heuristic function. It means that when the board game is close to the center, the AB_Custom_2 will obtain greater chance to win.

In addition, although I mixed heuristic 1 & 2 to the AB_Custom_3 it does not always play best result in every match. It is because the heuristic center diff and common set do not always influence efficiently to the win or lose, but sometimes they can be very effective to help us to choose the best step. Thus, when using heuristic 3 we can always retrieve average result but not best result.

Recommendations

Based on the results in the tournament of all the three heuristics, I would be inclined to recommend heuristic 3 to be used because of the following reasons:

- Heuristic 3 can provide a reasonable result when the opponent's total moves is the same as my total moves, which means it can perform better result in this undetermined situation of game broad.
- Heuristic 3 includes positional advantage to evaluate the better step in game broad by distance measurement, which makes the result is more reasonable. It is because we don't want the next move is too far away from opponents.
- Heuristic 3 is able to block available move to the opponent, and always selecting board states where these is a possibility for the me to block the opponent. Also, it does not increase complexity of programing and then we can get better result.