Heuristics Introduction

I proposed 3 heuristics modified from improved score, with heuristic 1 and 2 weighting opponent player and player, resulting heuristic 1 relatively more offensive, heuristic 2 relatively defensive, and heuristic 3 with addition of domain knowledge measuring corners on the board.

Heuristic 1:

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

    if game.is_winner(player):
        return float("inf")

    own_moves_len = len(game.get_legal_moves(player))
    opp_moves_len = len(game.get_legal_moves(game.get_opponent(player)))

    return float(own_moves_len - opp_moves_len*2)
```

This heuristic evaluates the difference between players available move and twice of opponent's available move. This heuristic translates into a more offensive behavior consist in trying to obstruct the opponent's game by occupying what would appear to be the best cell for the opponent.

Heuristic 2:

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

    if game.is_winner(player):
        return float("inf")

    own_moves_len = len(game.get_legal_moves(player))
    opp_moves_len = len(game.get_legal_moves(game.get_opponent(player)))

    return float(own_moves_len*2 - opp_moves_len)
```

Heuristic 2 evaluates the difference between twice of players available move and opponent's available move. This heuristic translates into a more defensive behavior consist in trying to keep as many options as possible open, which results in a general gravitation of the moves towards the center of the board, which is where one has more moves available as oppose to the sides of the board.

Heuristic 3:

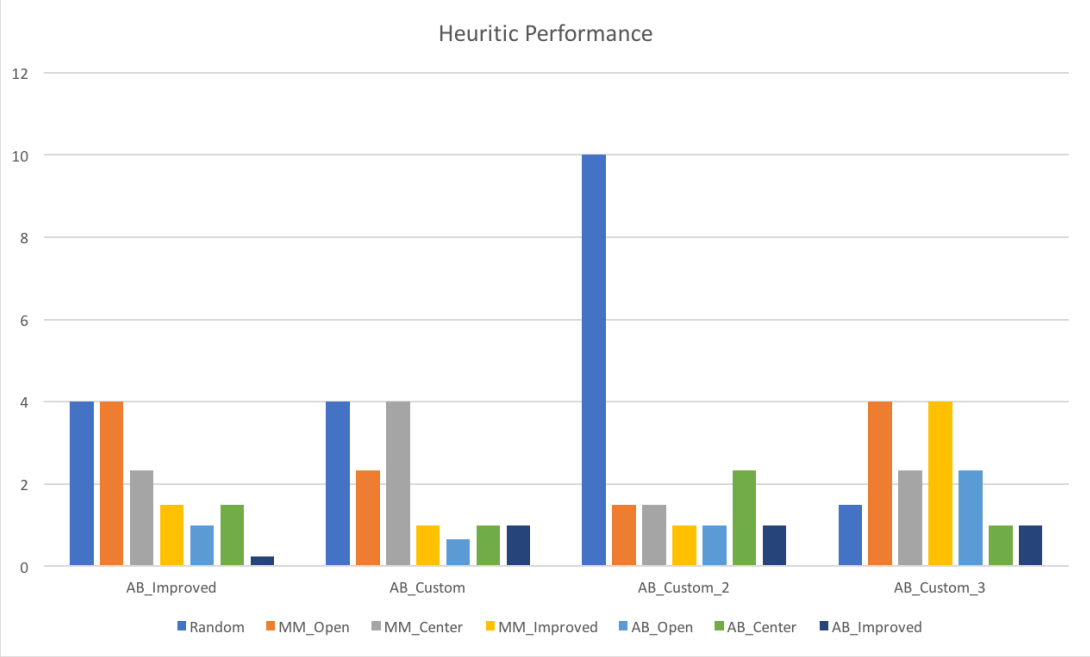
```
    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")

    game_state_factor = 1

    if len(game.get_blank_spaces()) < game.width * game.height / 4.:
        game_state_factor = 4

    corners = [(0, 0),
               (0, (game.width - 1)),
               ((game.height - 1), 0),
               ((game.height - 1), (game.width - 1))]

    own_moves = game.get_legal_moves(player)
    own_in_corner = [move for move in own_moves if move in corners]
    opp_moves = game.get_legal_moves(game.get_opponent(player))
    opp_in_corner = [move for move in opp_moves if move in corners]

    return float(len(own_moves) - (game_state_factor * len(own_in_corner)) - len(opp_moves)
+ (game_state_factor * len(opp_in_corner)))
```

Heuristic 3 is a combination of available moves with domain knowledge. It evaluates the difference between players available moves deducts moves to corner, and opponent's available moves plus moves to corner. The goal of the heuristic is to evaluate available moves while penalizing player's move towards corner and rewarding opponent's moves towards corner.

Performance

```
**************************
   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	8	2	10	0	6	4
2	MM_Open	8	2	7	3	6	4	8	2
3	MM_Center	7	3	8	2	6	4	7	3
4	MM_Improved	6	4	5	5	5	5	8	2
5	AB_Open	5	5	4	6	5	5	7	3
6	AB_Center	6	4	5	5	7	3	5	5
7	AB_Improved	2	8	5	5	5	5	5	5
	Win Rate:	60.0%		60.0%		62.9%		65.7%	



To evaluate the performance of the heuristics, a tournament was utilized with a setup consist of a Random AI agent that moves randomly. Minimax (MM) with a search depth of 3, and Minimax with alpha-beta pruning (AB). There are three heuristics provided for tests, open score, which evaluates the available move of the player on the board, improved score, which evaluates the difference between the available moves of player and opponent, center score, which evaluates the distance between the player and the center of the board.

The tournament simulated 5 matches against each opponents with a time limit of 150 milliseconds, which resulted the table above, showing heuristic 3 has a better overall win rate. A visualization of won/lost also demonstrates that heuristic 3 also a more consistent

performance against different opponents. Base on the result, I would recommend heuristic 3 for following reasons:

1. Best overall win rate.
2. Most consistent performance over different opponents.
3. Heuristic 3 has similar time complexity as score improved, heuristic 1 and 2, it is still using available move as the main measurement. This allows heuristic 3 to reach similar maximum depth in the search tree only with better performance because of the simple addition of the corner measurement.
4. Heuristic 3 is relatively more complex than heuristic 1 and 2. However, it is still very easy to implement with the only addition is the predetermined corner domain knowledge.