Premise

I have experimented with multiple combinations of heuristics individually and then clubbing them in score values. Heuristics explored are based on following premises:

1. Centrality of the move - Moves leading to center should be better (less probable to get blocked)
2. Location of the player with respect to the center
3. Number of common moves available: number of moves which both player have in common (degree of freedom for blocking)
4. Number of games moves which have passed - to change strategy as the games goes older

It was observed that centrality of the resulting moves plays a very important role. Further experiments like different strategies in beginning and end also were interesting. In the final run, custom_score_5 based on centrality of location performed better. In intermediate results however total centrality scores were better.

custom_score

This function augments the improved_score score by combining it with centrality of all possible moves. Achieved 58.6% in tournament as opposed to 61.4% of improved_score.

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

custom_score_2

This function unit tests the options besides common move as a parameter.
Achieved 60.0% in tournament as opposed to 61.4% of improved_score. It was unexpected correlation for me.

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

if game.is_winner(player):
```

```
        return float("inf")

    return float(freedom_from_common_moves(game, player))
```

custom_score_3

This function uses improved_score function if the player is in middle of the board or else freedom from common moves (8- common moves).
Achieved 60.0% in tournament as opposed to 61.4% of improved_score.

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

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

    ploc_x, ploc_y = game.get_player_location(player)
    oloc_x, oloc_y = game.get_player_location(game.get_opponent(player))
    pmoves = game.get_legal_moves(player)
    omoves = game.get_legal_moves(game.get_opponent(player))

    if math.fabs(ploc_x - oloc_x) >= game.width / 2 and math.fabs(ploc_y -
    oloc_y) >= game.height / 2:
        return float(len(pmoves) - len(omoves))
    return float(freedom_from_common_moves(game, player))
```

custom_score_4

Checks the score for distance between players.
Achieved 58.6% in tournament as opposed to 61.4% of improved_score.

```
    ploc_x, ploc_y = game.get_player_location(player)
    oloc_x, oloc_y = game.get_player_location(game.get_opponent(player))

    return float((ploc_x-oloc_x)**2 + (ploc_y-oloc_y)**2)
```

custom_score_5

It augments the improved_score by adding centrality of the player location.
Achieved 65.7% in tournament as opposed to 61.4% of improved_score. It is the best performing score.

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

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

    pmoves = len(game.get_legal_moves())
    omoves = len(game.get_legal_moves(game.get_opponent(player)))

    return float(pmoves - omoves + centrality(game,
    game.get_player_location(player)))
```

custom_score_6

Experiments with a combination of freedom of common moves and centrality of such moves.
Achieved 60.0% in tournament as opposed to 61.4% of improved_score.

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

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

    return float(freedom_from_common_moves(game, player) +
    best_common_move_to_center(game, player))
```

custom_score_7

Use freedom from common moves as a parameter in beginning and then use distance between players.
Achieved 64.3% in tournament as opposed to 61.4% of improved_score.

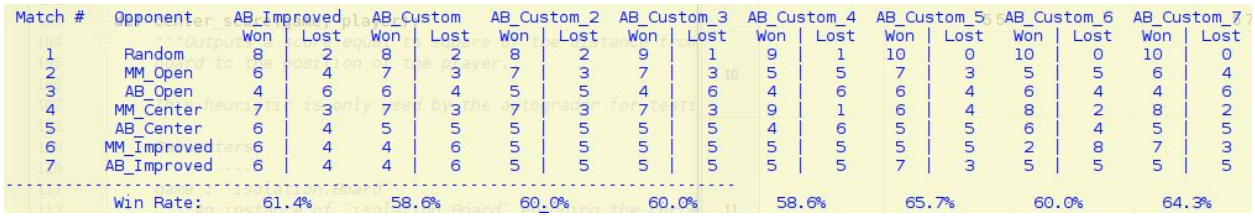
```
    if game.is_loser(player):
        return float("-inf")

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

    if game.move_count <= 5:
        return custom_score_2(game, player)
    return custom_score_4(game, player)
```

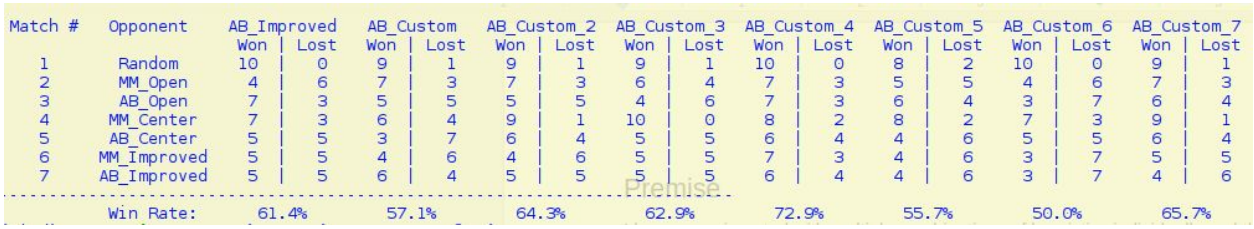
Performance

Performance figures of final run



Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3		AB_Custom_4		AB_Custom_5		AB_Custom_6		AB_Custom_7	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost	Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	8	2	8	2	8	2	9	1	9	1	10	0	10	0	10	0
2	MM_Open	6	4	7	3	7	3	7	3	5	5	7	3	5	5	6	4
3	AB_Open	4	6	6	4	5	5	4	6	4	6	6	4	6	4	4	6
4	MM_Center	7	3	7	3	7	3	7	3	9	1	6	4	8	2	8	2
5	AB_Center	6	4	5	5	5	5	5	5	4	6	5	5	6	4	5	5
6	MM_Improved	6	4	4	6	5	5	5	5	5	5	5	5	2	8	7	3
7	AB_Improved	6	4	4	6	5	5	5	5	5	5	7	3	5	5	5	5
	Win Rate:	61.4%		58.6%		60.0%		60.0%		58.6%		65.7%		60.0%		64.3%	

Performance figures of intermediate run where a result of 72.9% was achieved with respect to 61.4% of improved_score.



Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3		AB_Custom_4		AB_Custom_5		AB_Custom_6		AB_Custom_7	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost	Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	10	0	9	1	9	1	9	1	10	0	8	2	10	0	9	1
2	MM_Open	4	6	7	3	7	3	6	4	7	3	5	5	4	6	7	3
3	AB_Open	7	3	5	5	5	5	4	6	7	3	6	4	3	7	6	4
4	MM_Center	7	3	6	4	9	1	10	0	8	2	8	2	7	3	9	1
5	AB_Center	5	5	3	7	6	4	5	5	6	4	4	6	5	5	6	4
6	MM_Improved	5	5	4	6	4	6	5	5	7	3	4	6	3	7	5	5
7	AB_Improved	5	5	6	4	5	5	5	5	6	4	4	6	3	7	4	6
	Win Rate:	61.4%		57.1%		64.3%		62.9%		72.9%		55.7%		50.0%		65.7%	