Nikolas Katigbak
March AIND — Term 1
Project 1 — Advanced Game Playing Agent
Heuristic Analysis

Heuristic 1

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

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

    oppMoves = set(game.get_legal_moves(game.get_opponent(player)))
    myMoves = set(game.get_legal_moves(player))

    myTurn = game.active_player == player
    return len(myMoves) / (1 + len(oppMoves.intersection(myMoves)))
```

This metric tries to capture the count of our player's moves discounted by the count of common moves between our player and the opponent. The motivation behind this metric is that we would like to score higher for outcomes wherein we not only have more legal moves but those legal moves are ours only such that we are limiting our exposure to attack from the opponent.

Heuristic 2

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

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

    blanks = game.get_blank_spaces()
    myMoves = set(game.get_legal_moves(player))

    outs = 0
    for blank in  blanks:
        if blank[0] <= game.width/3 or blank[0] >= 2/3*game.width:
            outs+=1
        elif blank[1] <= game.height/3 or blank[1] >= 2/3*game.height:
            outs+=1
    return -outs + len(myMoves) / len(game.get_blank_spaces())
```

This metric measure how many of the blank spaces are in the perimeter of the board; “perimeter access”. Through my own game-playing experience, having more blank spaces in the perimeter of the board meant more of the moves had been played in the center of the board and this meant escaping to the perimeter was dangerous. This metric still also captures a ratio of legal moves to blank spaces which expresses our player’s access to the board while it also subtracts perimeter access from the score.

Heuristic 3

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

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

    totalMoves = len(game.get_blank_spaces())
    oppMoves = len(game.get_legal_moves(game.get_opponent(player)))
    myMoves = len(game.get_legal_moves(player))

    meLoc = game.get_player_location(game.active_player)
    oppLoc = game.get_player_location(game.get_opponent(game.active_player))
    usDist = abs(meLoc[0] - oppLoc[0]) + abs(meLoc[1] - oppLoc[1])

    return float(myMoves - oppMoves) / float(totalMoves)
```

This metric attempts to capture board access enhanced by our player’s lack of board access. playerMove/blankSpaces is a metric I use to measure how much access player has to the board. If this ratio is close to 1 then that means player has full access while the closer this ratio is to 0 means our player has lesser access perhaps because the opponent is blocking us.

Below Are Three Tournament Runs

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	9	1	10	0	10	0	10	0
2	MM_Open	8	2	7	3	8	2	10	0
3	MM_Center	10	0	9	1	9	1	10	0
4	MM_Improved	7	3	9	1	8	2	8	2
5	AB_Open	6	4	6	4	4	6	5	5
6	AB_Center	7	3	5	5	6	4	6	4
7	AB_Improved	3	7	5	5	3	7	5	5
	Win Rate:	71.4%		72.9%		68.6%		77.1%	

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	10	0	10	0	10	0	10	0
2	MM_Open	8	2	8	2	10	0	8	2
3	MM_Center	10	0	10	0	8	2	10	0
4	MM_Improved	8	2	8	2	7	3	6	4
5	AB_Open	6	4	5	5	5	5	7	3
6	AB_Center	6	4	6	4	5	5	4	6
7	AB_Improved	5	5	3	7	4	6	8	2
	Win Rate:	75.7%		71.4%		70.0%		75.7%	

Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	10	0	10	0	10	0	10	0
2	MM_Open	9	1	9	1	10	0	7	3
3	MM_Center	9	1	10	0	10	0	9	1
4	MM_Improved	8	2	8	2	7	3	8	2
5	AB_Open	4	6	5	5	6	4	6	4
6	AB_Center	6	4	5	5	4	6	8	2
7	AB_Improved	5	5	5	5	5	5	6	4
	Win Rate:	72.9%		74.3%		74.3%		77.1%	

Results Analysis

Heuristic 3 performed the most consistently and strongly throughout the tournament runs. I believe this is because it simultaneously scores for board configurations that not only give our player access but gives our opponent less access.

However, I also noted that Heuristic 3's performance is/was still very close to that of AB_Improved despite many different adjustments made to Heuristic 3. This is most probably due to the fact that Heuristic 3 is an enhancement to AB_Improved.

Recommendation

Based on the tournament results I recommend that we use heuristic 3 for tournament playing for the following reasons:

- Heuristic 3 consistently performed the best among all other heuristics developed

- Heuristic 3 is not costly. It is constant time relative to game_state and node count since the game already keeps the state of the game in memory and therefore is not researched at every call to the Heuristic function.
- Heuristic 3 is also very straightforward to implement and reason about. It does not implement, on its own, any complicated/advanced or costly calculations.
- Heuristic 3 uses floats and gives a higher degree of precision and comparability than ints