AIND - Isolation Heuristic Analysis

Match	Opponent	AB_Improved Won | Lost	AB_Custom Won | Lost	AB_Custom 2 Won | Lost	AB_Custom 3 Won | Lost
1	Random	48 | 12	52 | 8	49 | 11	56 | 4
2	MM_Open	42 | 18	37 | 23	40 | 20	40 | 20
3	MM_Center	43 | 17	43 | 17	50 | 10	50 | 10
4	MM_Improved	33 | 27	36 | 24	41 | 19	40 | 20
5	AB_Open	36 | 24	30 | 30	31 | 29	31 | 29
6	AB_Center	30 | 30	32 | 28	29 | 31	33 | 27
7	AB_Improved	27 | 33	31 | 29	31 | 29	28 | 32
Win Rate		61.7%	62.1%	64.5%	66.2%

In this project I have developed 3 different heuristics. Each combination played 60 games to reduce the randomness of the result.

The first heuristic **AB_Custom** compares the sum of the possible moves of the player and the opponent in next two moves. Using the existing methods provided by the project, it is relatively easy to implement. However, because it calculates 2 moves ahead, the computation time is slightly longer.

$$sum\_player\_moves - sum\_opponent\_moves$$

The second heuristic **AB_Custom 2** follows the AIND course and makes the player follows the opponent but slightly less aggressive. Instead of subtracting two times of opponent moves, I used 1.75. It seems to provide slightly better result than 2. This heuristic is easy to implement and quick to compute.

$$len(my\_player\_move) - 1.75 * len(opponent\_move)$$

The third heuristic **AB_Custom 3** is the combination of the first and second heuristics. It is still relatively easy to implement. Due to its complexity, the computation cost is higher. This

may result timeout exception where the player could not complete the game while there are still possible moves.

$$len(player\_moves) * sum\_player\_moves - len(opponent\_moves) * sum\_opponent\_moves$$

Final Recommendation

I think AB_Custom 3 is the best heuristic out of all three.

1. It outperforms other heuristic with 66.2% winning rate.
2. The heuristic is still easy to implement base on the methods provided in this project.
3. The heuristic does use extra memory comparing to **AB_Custom 2.** Only considering 2 moves ahead doesn't sacrifice too much of memory in the current setting.
4. The computation cost is higher, but its benefits outweighs the cost. This is observed in the result. It is the heuristic provides the most steady results.