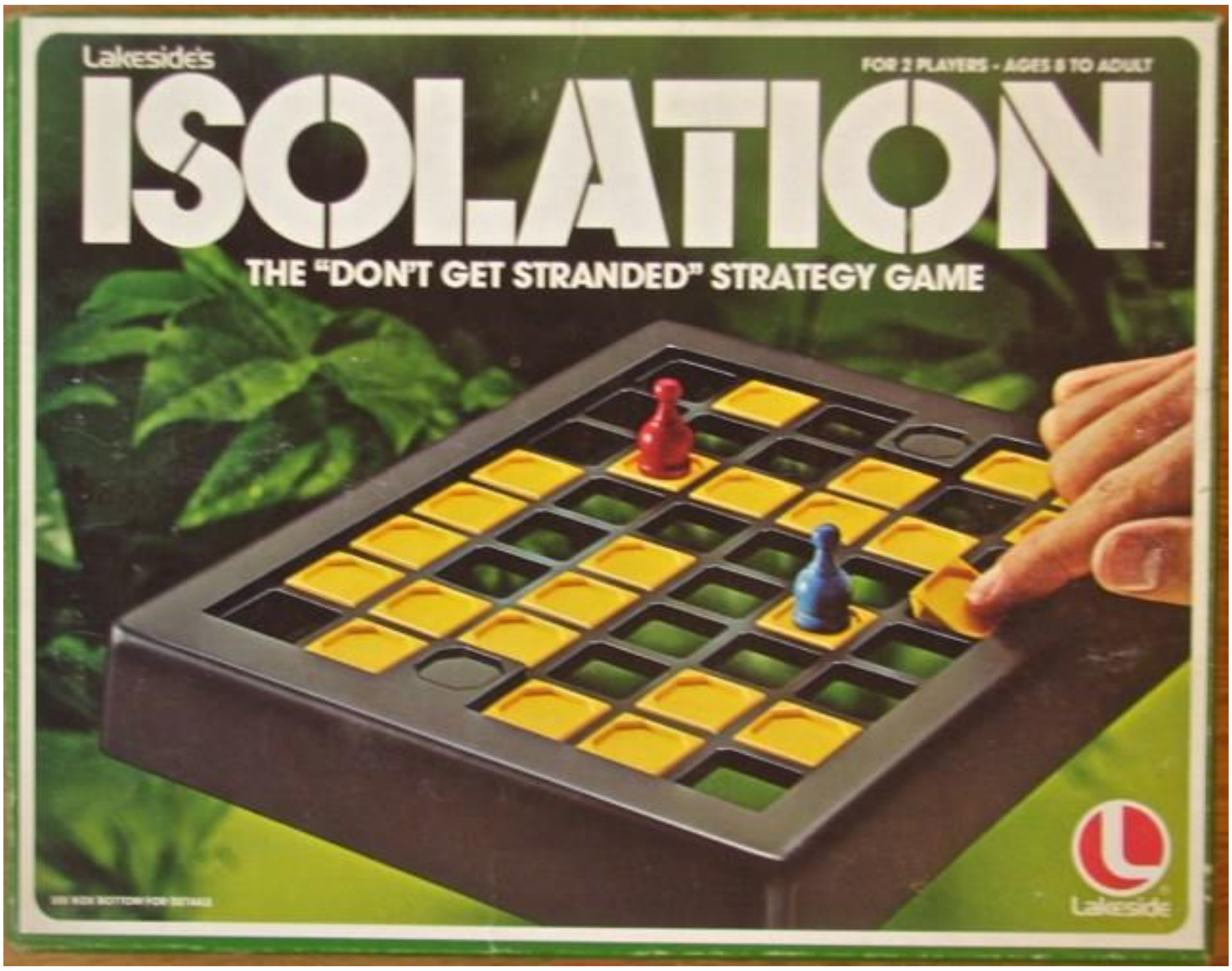
Heuristic Analysis Report for Adversarial Game Playing Agent for Isolation

By Tejasvi Nuthalapati
As part of **Artificial Intelligence Nano Degree (AIND)** at **Udacity**


Lakeside's
ISOLATION
FOR 2 PLAYERS - AGES 8 TO ADULT
THE "DON'T GET STRANDED" STRATEGY GAME
Lakeside

INDEX OF CONTENTS

INTENTIONALLY LEFT BLANK

1. INTRODUCTION

Isolation is a deterministic, two player game where players alternatively take turns to move their pieces from one cell to the other, on occupying a cell, that cell is deactivated/blocked for further moves. As the game proceeds the last player who has no more moves loses the game. In this project the moves of the pieces are ‘L’ shaped like the Knights in chess. The agent can move to any open cell on the board in the form of 2-row + 1-column or 2-column + 1-row with restrictions at the edges without wrap-around but they can jump over blocked or occupied cell. In this game there is a Time Limit on each turn to search a best move and play that player’s turn. On time limit expiry that player loses the match letting the opponent win.

2. EVALUATION

In the code base of AIND-Isolation, the files

- **game_agent.py** - Has the Heuristics, Minimax, Alpha-Beta algorithms and Iterative Deepening Search
- **MinimaxPlayer.**minimax() - implements the full recursive search procedure described in lecture (AIMA Minimax Decision).
- **AlphaBetaPlayer**.alphabeta() - method to implement the full recursive search procedure described in lecture (ref. AIMA Alpha-Beta Search)
- **AlphaBetaPlayer.get_move()** to implement Iterative Deepening. (ref. AIMA Iterative Deepening Search)
- custom_score() , custom_score_2(), custom_score_3() – **Student** heuristics
- **tournament.py -** script is used to evaluate the effectiveness of my (named "**Student**" in the tournament) custom heuristics. The script measures relative performance of my agent in a round-robin tournament against several other pre-defined agents. The **Student** agent uses time-limited Iterative Deepening along with my custom heuristics.

***END GOAL: Develop a heuristic that makes ‘Student’ outperform ID_Improved.**

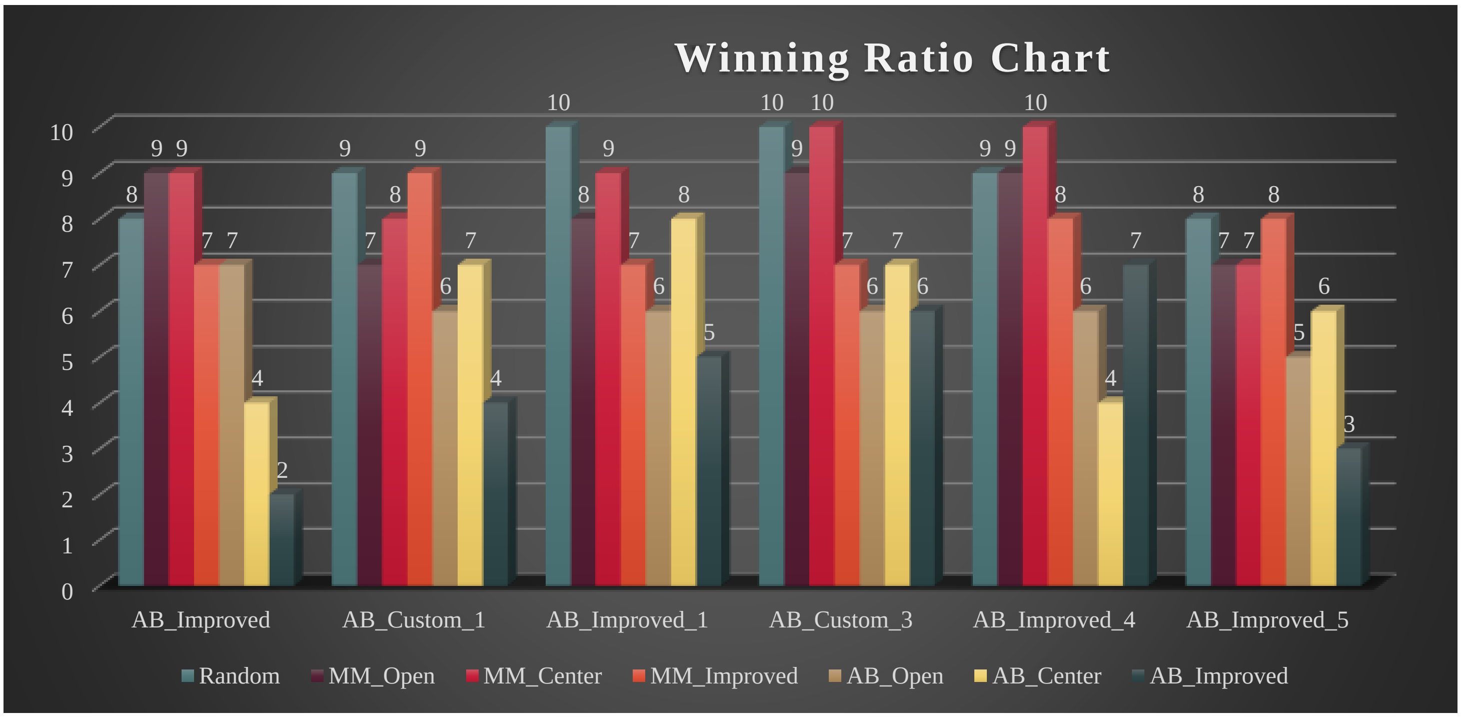
2.1 Opponent Heuristics in this tournament are:

SNO	Heuristic	Description
1	**Random**	An agent that randomly chooses a move each turn
2	**MM_Open**	MinimaxPlayer agent using the open_move_score heuristic with search depth 3
3	**MM_Center:**	MinimaxPlayer agent using the center_score heuristic with search depth 3
4	**MM_Improved**	MinimaxPlayer agent using the improved_score heuristic with search depth 3
5	**AB_Open**	AlphaBetaPlayer using iterative deepening alpha-beta search and the open_move_score heuristic
6	**AB_Center**	AlphaBetaPlayer using iterative deepening alpha-beta search and the center_score heuristic
7	**AB_Improved**	AlphaBetaPlayer using iterative deepening alpha-beta search and the improved_score heuristic

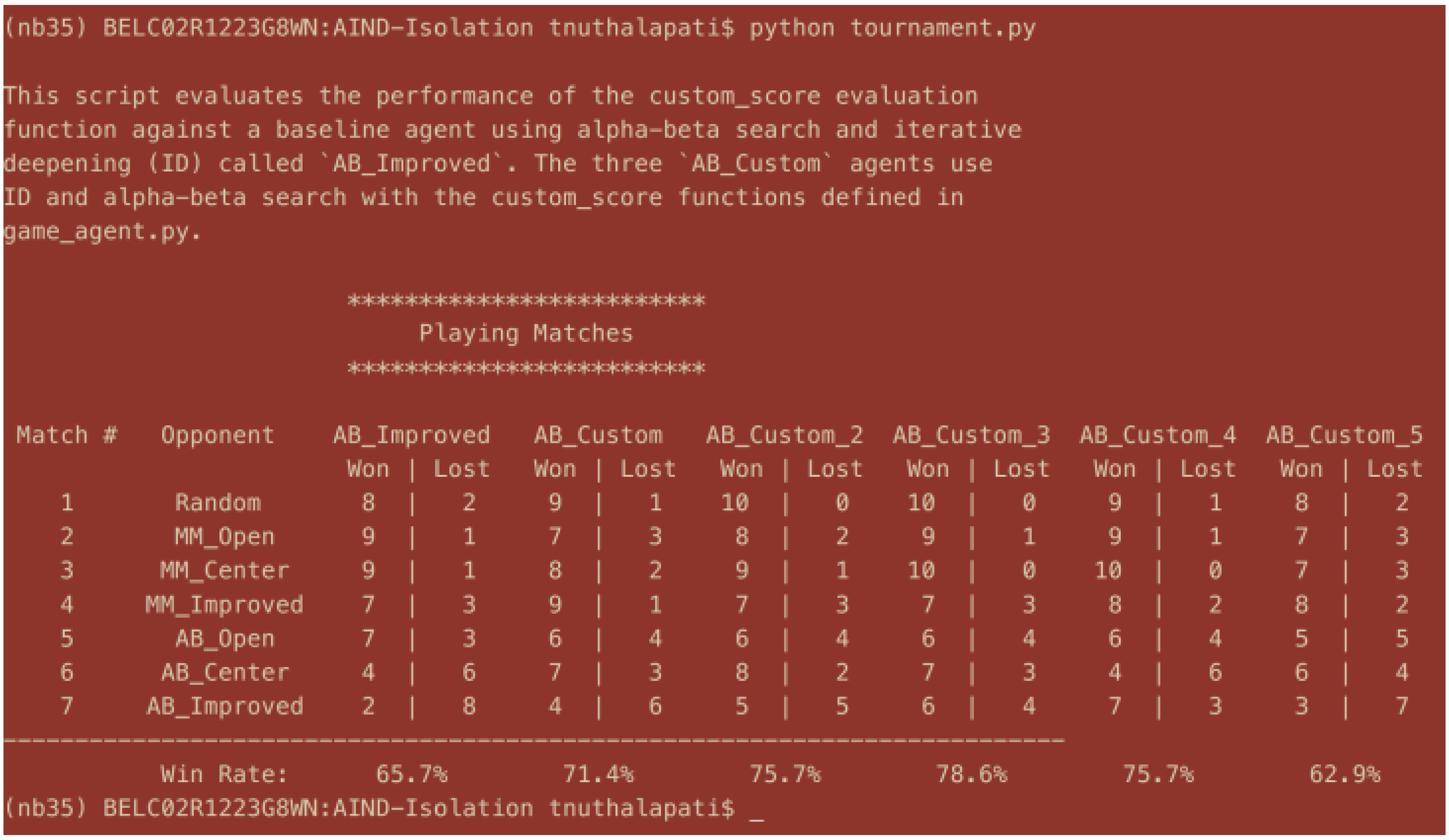
2.2 Student Heuristics in this tournament are:

SNO	Heuristic	Description
1	**Heuristic_1**	Increases the Weighted Chance Heuristic of ‘Student’ agent = my_moves^2 – 1.5*opp_moves^2
2	**Heuristic_2**	This Heuristic maximizes the winning chance of the ‘Student’ agent = my_moves/opp_moves
3	**Heuristic_3**	This Heuristic reduces the loosing chance of the ‘Student’ agent = - opp_moves/my_moves
4	**Heuristic_4**	This Heuristic maximizes the winning chance of the ‘Student’ agent = my_moves* my_moves /opp_moves
5	**Heuristic_5**	This Heuristic maximizes the winning chance of the ‘Student’ agent = 3*my_moves* my_moves /opp_moves

3. ANALYSIS



3.1 This graph represents the Tournament analysis with winning ratios for 10 games average against each of the Agents



```
(nb35) BELC02R1223G8WN:AIND-Isolation tnuthalapati$ python tournament.py

This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                        *************************
                             Playing Matches
                        *************************

 Match #   Opponent    AB_Improved   AB_Custom    AB_Custom_2  AB_Custom_3  AB_Custom_4  AB_Custom_5
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       8  |  2      9  |  1     10  |  0     10  |  0      9  |  1      8  |  2
    2       MM_Open      9  |  1      7  |  3      8  |  2      9  |  1      9  |  1      7  |  3
    3      MM_Center     9  |  1      8  |  2      9  |  1     10  |  0     10  |  0      7  |  3
    4     MM_Improved    7  |  3      9  |  1      7  |  3      7  |  3      8  |  2      8  |  2
    5       AB_Open      7  |  3      6  |  4      6  |  4      6  |  4      6  |  4      5  |  5
    6      AB_Center     4  |  6      7  |  3      8  |  2      7  |  3      4  |  6      6  |  4
    7     AB_Improved    2  |  8      4  |  6      5  |  5      6  |  4      7  |  3      3  |  7
--------------------------------------------------------------------------
          Win Rate:      65.7%        71.4%        75.7%        78.6%        75.7%        62.9%
(nb35) BELC02R1223G8WN:AIND-Isolation tnuthalapati$ _
```

3.2 Console output of the Tournament Analysis

4. CONCLUSION

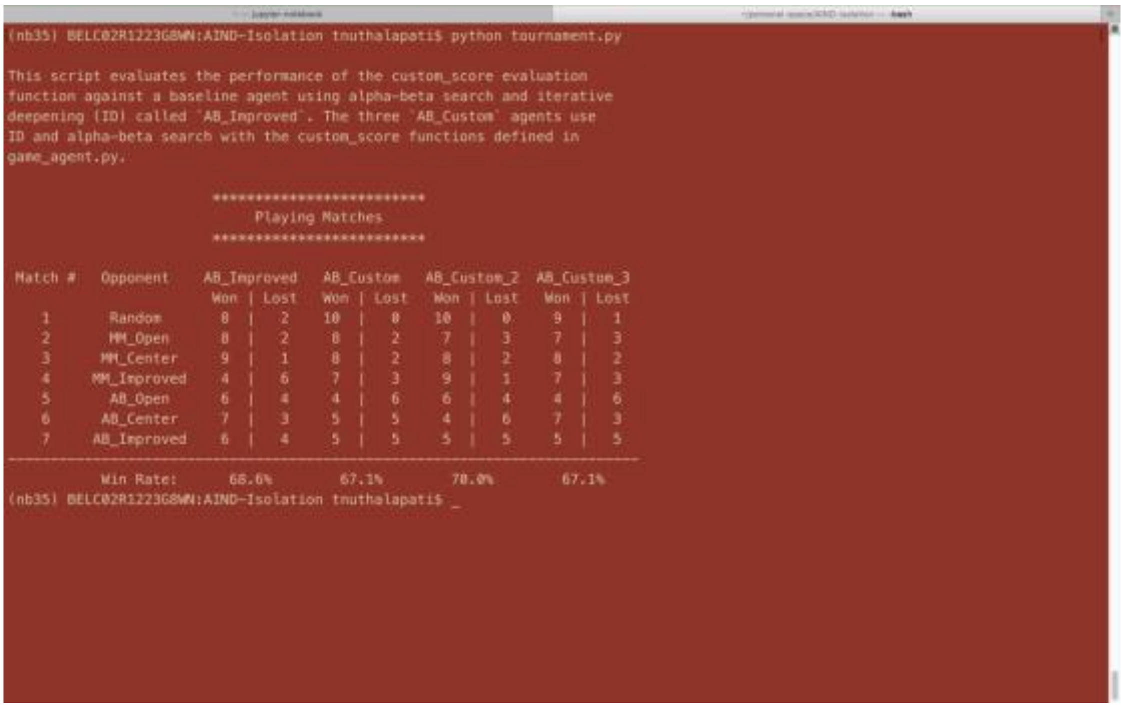
Most of the heuristics performed significantly better than the **AB_Improved** heuristic, where the **winning heuristic ratio AB_custom_3** outperforms of them all with a winning 78.6% of the matches.
I recommend using AB_custom_3 heuristics as:

- It outperforms others with a 78.6% winning chance
- It’s quick and easy to implement
- The heuristic always focuses on reducing the loosing chance which in-turn helps to increase the winning chance
- This is a simple and fast heuristic which proceeds deeper in the game tree
- No strings attached, it means the heuristic doesn’t make any assumptions about the previous/next state but just based on the current state of the player

This result is dependent upon the Hardware performance but my assumption after running comparatively on a Windows/Mac PC the results slightly differ but never significantly. By this analysis and project implementation I can conclude that using my Student AB_Custom_3 heuristic there are 78.6% of chances of winning a tournament.

5. APPENDIX:

Other Analysis reports performed before concluding the above results.



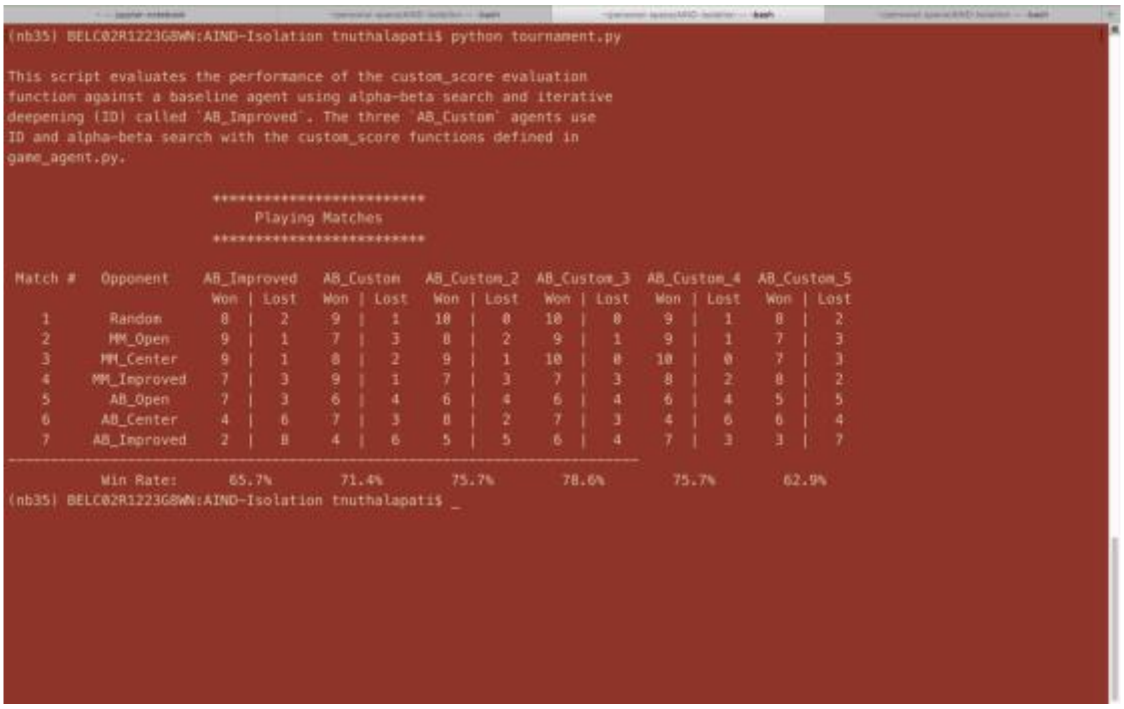
```
(nb35) BELC02R1223G8WN:AIND-Isolation tnuthalapati$ python tournament.py

This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                        *************************
                             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    10  |   0    10  |   0     9  |   1
    2       MM_Open      8  |   2     8  |   2     7  |   3     7  |   3
    3      MM_Center     9  |   1     8  |   2     8  |   2     8  |   2
    4     MM_Improved    4  |   6     7  |   3     9  |   1     7  |   3
    5       AB_Open      6  |   4     4  |   6     6  |   4     4  |   6
    6      AB_Center     7  |   3     5  |   5     4  |   6     7  |   3
    7     AB_Improved    6  |   4     5  |   5     5  |   5     5  |   5
--------------------------------------------------------------------------
           Win Rate:      68.6%        67.1%        70.0%        67.1%
(nb35) BELC02R1223G8WN:AIND-Isolation tnuthalapati$ _
```

5.1 First report generated after successful execution of the code



```
(nb35) BELC02R1223G8WN:AIND-Isolation tnuthalapati$ python tournament.py

This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                        *************************
                             Playing Matches
                        *************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3  AB_Custom_4  AB_Custom_5
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       8  |   2     9  |   1    10  |   0    10  |   0     9  |   1     8  |   2
    2       MM_Open      9  |   1     7  |   3     8  |   2     9  |   1     9  |   1     7  |   3
    3      MM_Center     9  |   1     8  |   2     9  |   1    10  |   0    10  |   0     7  |   3
    4     MM_Improved    7  |   3     9  |   1     7  |   3     7  |   3     8  |   2     8  |   2
    5       AB_Open      7  |   3     6  |   4     6  |   4     6  |   4     6  |   4     5  |   5
    6      AB_Center     4  |   6     7  |   3     8  |   2     7  |   3     4  |   6     6  |   4
    7     AB_Improved    2  |   8     4  |   6     5  |   5     6  |   4     7  |   3     3  |   7
--------------------------------------------------------------------------
           Win Rate:      65.7%        71.4%        75.7%        78.6%        75.7%        62.9%
(nb35) BELC02R1223G8WN:AIND-Isolation tnuthalapati$ _
```

5.2 Second report generated after minor tweeks to the old heuristics and introducing the new ones