# Isolation Game Heuristic Analysis

By Yutao Liu

For the project, I experimented different heuristic functions. Compared with limit time and calculation resources, the simpler function allows the search to proceed deeper in the game tree, at first I have tried to implement Open Score and Center Score, but they always perform not very well. I think they only contain very few information about the game. So I tried some complex function to get better result.

Here is a summary of these complex results:

I’ve compared 4 different heuristic functions, and the first two functions are modified from the improved\_score() and center\_score.

The last two functions merged the first two functions in different ways.

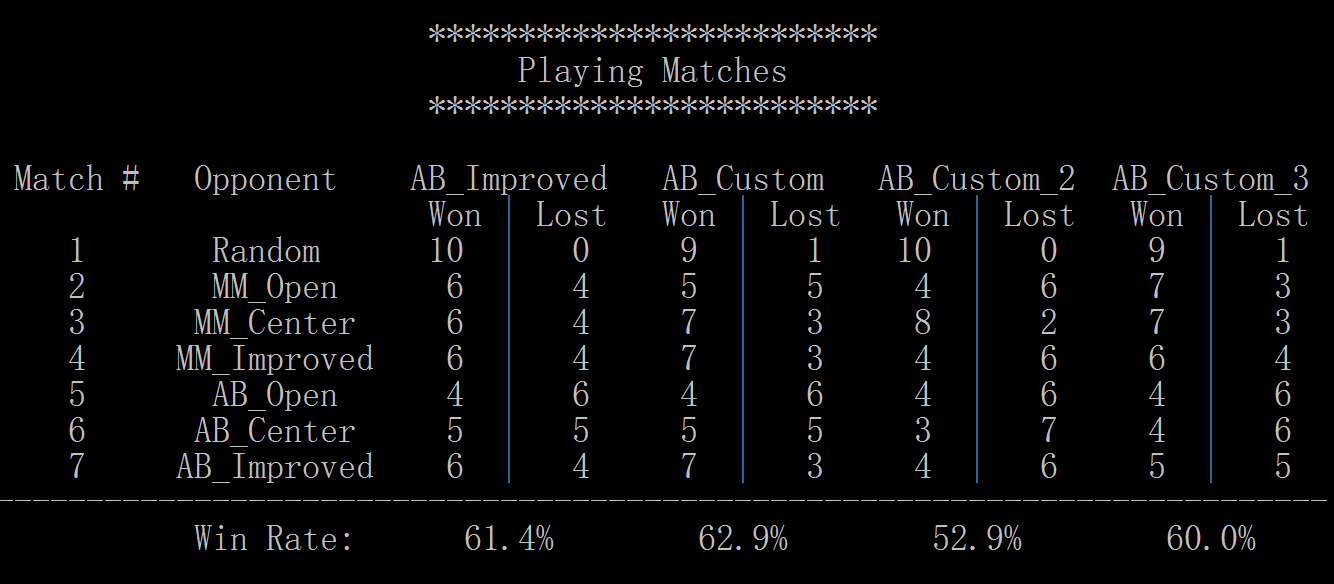
|  |  |  |
| --- | --- | --- |
| Customer Player | Opponent Player’s Win Rate | Average player’s Win Rate |
| Heuristic 1 | 61.4% | 58.6% |
| Heuristic 2 | 60.0% | 60.96% |
| Heuristic 3 | 61.4% | 62.3% |
| Heuristic 4 | 55.7% | 60.46% |

**Heuristic 1:**

With this heuristic, I got inspiration from improving\_score (). Where should the customer\_player do when the two players have the common move. I think if the player and the opponent player have the common move, its better to choose the common move and make the opponent player have less move to go.

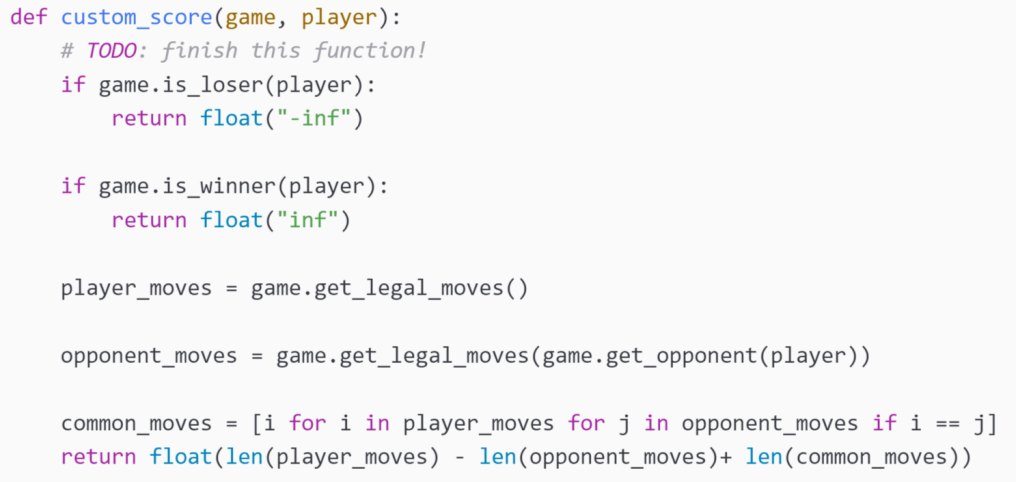
So I use the length of the player’s moves minus the length of opponent’s moves and then add the length of common moves.

**Results:**



**Implementation:**

Here is the implementation of the heuristic function:



**Analysis:**

Not a great heuristic function and not performs well every time. I think its not a good choice to occupy the opponent’s possible move at first but maybe its useful in the last few moves.

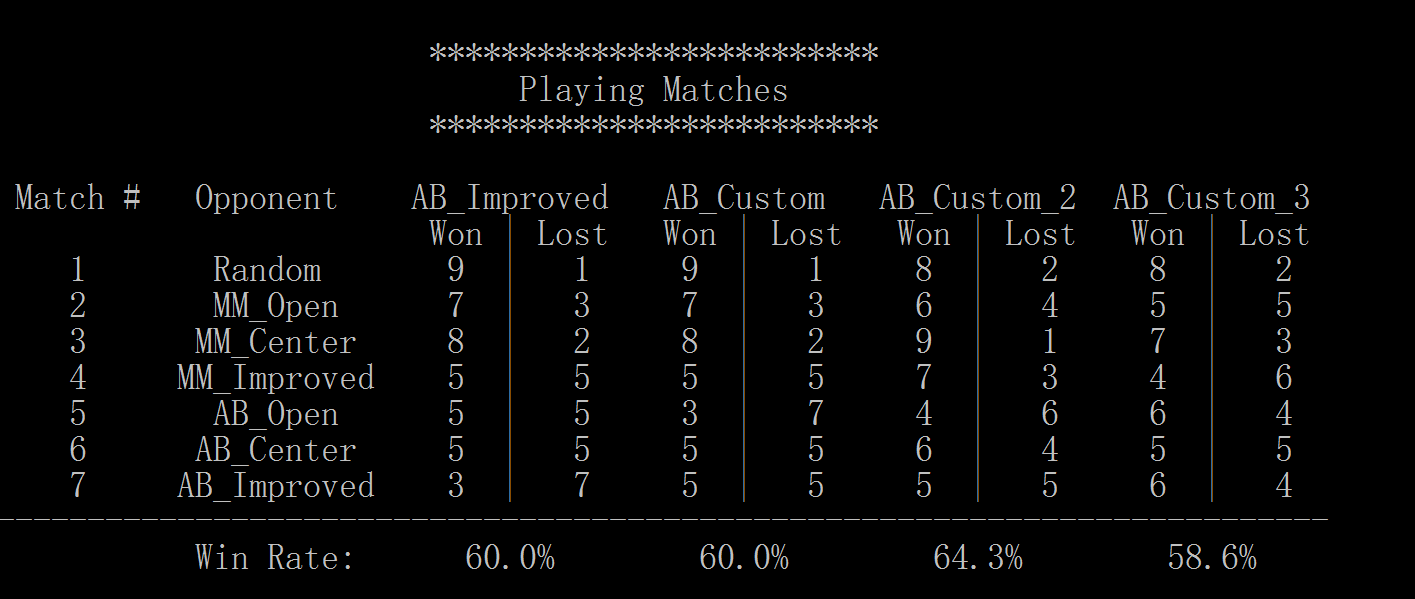
* Compared with the following heuristics, its very easy to compute and search deeper.
* According to the result, it wins once but lose twice, and average win rate is less than the AB\_improved heuristic.

**Heuristic2:**

With this heuristic, I got inspiration from center\_score (). I think maybe its better to occupy the center location first then the player may have more possible move to go.

So I use the square of the distance between center and the opponent’s location minus the square of the distance between center and the player’s location.

**Results:**



**Implementation:**

Here is the implementation of the heuristic function:



**Analysis:**

The heuristic preforms a bit better than AB\_improved. It seems like that occupy the center location first is a good strategy.

* Its a little more complex than the first heuristic, because it has some multiplication and square algorithms.
* But now it wins AB\_improved once and lose once, it’s average win rate is a little higher than AB\_improved.

**Heuristic3:**

With this heuristic, I just want to merge the heuristic1 and heuristic2. Maybe this time could perform better.

So I use the first heuristic score plus the second heuristic score.

**Results:**



**Implementation:**

Here is the implementation of the heuristic function:



**Analysis:**

This is a good strategy.But it doesn’t work very well all the time, because it is complex.

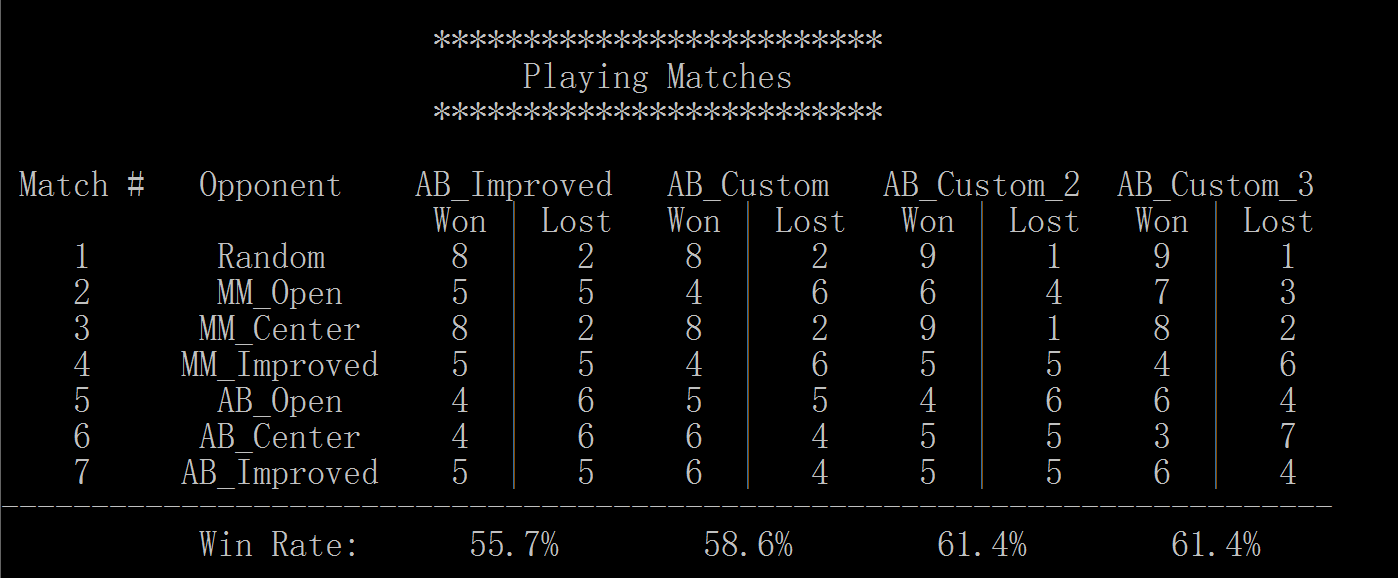
* It is a better strategy but cost too much calculate resource.
* In this heuristic function, we win the AB\_improved player once and achieve 70% win rate.

**Heuristic4:**

With this heuristic, I think the first heuristic function can perform well in the last few moves, and the second heuristic can perform better in the beginning of the game.

So the the heuristic function’s idea is when there is more than 3 moves to occupy, we choose the heuristic2 function, if there is less than 3 moves to occupy, we choose the heuristic1 function.

**Results:**



**Implementation:**

Here is the implementation of the heuristic function:



**Analysis:**

With different strategy in different state, performs better than the other heuristic function.

* A efficient strategy, which can go deeper than the third heuristic function.
* In this heuristic function, we already won the AB\_improved player 3 times.