Heuristics análysis:

The evaluation functions choosen for are the following:

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

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

    own_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
    return float((own_moves-(opp_moves**2))/(len(game.get_blank_spaces())))
```

For this first evaluation function, I give more importance to the opponent moves resting the square of them to the own_moves. Then this result is divided by the number of blank spaces in the board for “normalizing” the result as the game evolves.

```
def custom_score_2(game, player):
   if game.is_loser(player):
        return float("-inf")

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

    own_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
    return float((own_moves-opp_moves)/(len(game.get_blank_spaces())))
```

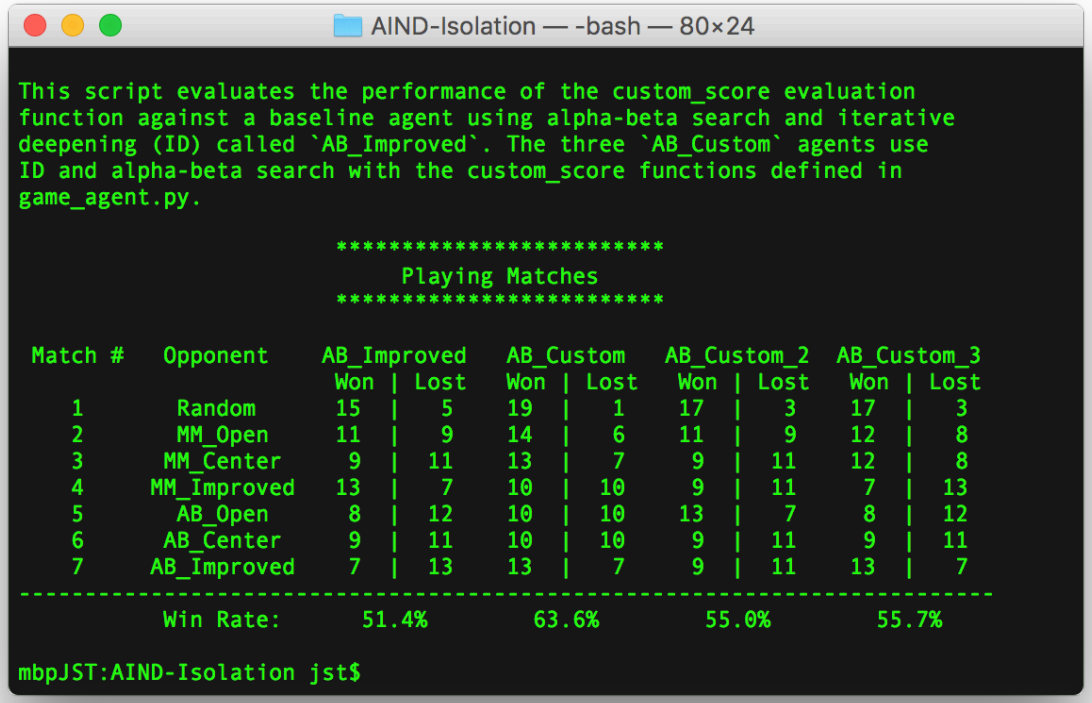
For this evaluation function, the result is the same as the “Improved” evaluation function, but dividing it by the number of blank spaces in the board, resulting in a more normalized score as the game evolves.

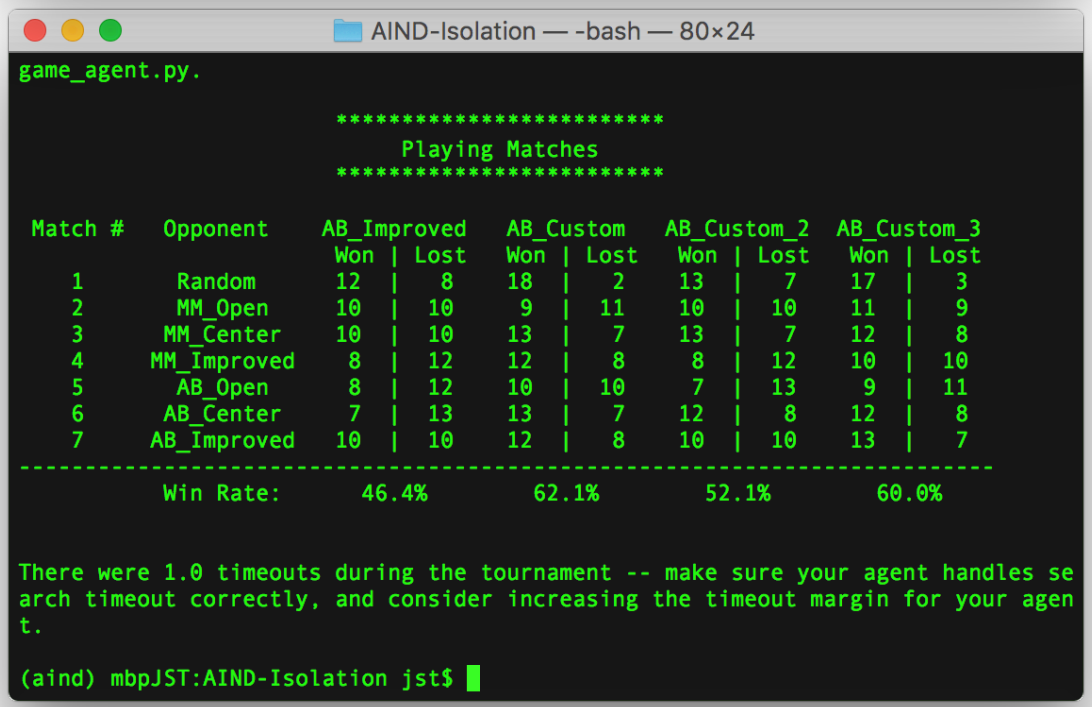
```
def custom_score_3(game, player):
  if game.is_loser(player):
        return float("-inf")

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

    own_moves = len(game.get_legal_moves(player))
    opp_moves = len(game.get_legal_moves(game.get_opponent(player)))
    if own_moves>opp_moves:
        return float(math.sqrt(own_moves**2 - opp_moves**2)/(len(game.get_blank_spaces())))
    else:
        return float(math.sqrt(opp_moves**2 - own_moves**2)/(len(game.get_blank_spaces())))*(-1.0)
```

For this last evaluation function, we calculate the square distance between own moves and opponent moves, and the divide it by the number of remaining blank spaces for normalizing.

Performance analysis:

Here we can see two different executions of the tournament.py, after modify it to play 20 games against each agent to get a higher number of matches to analyze:





In both captures, we can see that all the implemented evaluation functions beat the AB_Improved evaluation function, being the best result from latest capture with a difference in the win rate of 15.7 points.

The **recommendation** is to use the custom_score evaluation function, as it performs with higher win rate than AB_Improved, has the highest evaluation score compared to the other implemented evaluation functions, and beats the AB_Improved evaluation function in a rate of 6 over the 7 matches with different agents, being able to beat all of them if we consider the two presented tournament result captures.