

Heuristics análisis:

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:

```
AIND-Isolation — -bash — 80×24

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     15 |  5      19 |  1      17 |  3      17 |  3
2         MM_Open    11 |  9      14 |  6      11 |  9      12 |  8
3         MM_Center   9 | 11      13 |  7       9 | 11      12 |  8
4         MM_Improved 13 |  7      10 | 10       9 | 11       7 | 13
5          AB_Open    8 | 12      10 | 10      13 |  7       8 | 12
6          AB_Center   9 | 11      10 | 10       9 | 11       9 | 11
7         AB_Improved 7 | 13      13 |  7       9 | 11      13 |  7
-----
Win Rate:   51.4%      63.6%      55.0%      55.7%

mbpJST:AIND-Isolation jst$
```

```
AIND-Isolation — -bash — 80×24

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     12 |  8      18 |  2      13 |  7      17 |  3
2         MM_Open    10 | 10       9 | 11      10 | 10      11 |  9
3         MM_Center   10 | 10      13 |  7      13 |  7      12 |  8
4         MM_Improved 8 | 12      12 |  8       8 | 12      10 | 10
5          AB_Open    8 | 12      10 | 10       7 | 13       9 | 11
6          AB_Center   7 | 13      13 |  7      12 |  8      12 |  8
7         AB_Improved 10 | 10      12 |  8      10 | 10      13 |  7
-----
Win Rate:   46.4%      62.1%      52.1%      60.0%

There were 1.0 timeouts during the tournament -- make sure your agent handles se
arch timeout correctly, and consider increasing the timeout margin for your agen
t.

(aind) mbpJST:AIND-Isolation jst$
```

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.