# Heuristic Analysis

In order to win the game isolation, a player has to maximize his number of moves with every move, while at the same time minimizing the number of his opponent’s moves. The calculation of a weighted balance between these two goals is therefore a good approach to scoring individual boards in the game tree.

All of the three evaluated heuristics used in this project place an infinite penalty on losing a game and an infinite bonus on winning.

The first heuristic simply subtracts the number of opponents moves from the number of the active player's moves:

def custom\_score(game, player):

if game.is\_loser(player):

return float("-inf")

if game.is\_winner(player):

return float("inf")

number\_of\_active\_players\_moves = len(game.get\_legal\_moves(player))

number\_of\_oponentes\_players\_moves = len(game.get\_legal\_moves(game.get\_opponent(player)))

return float(number\_of\_active\_players\_moves - number\_of\_oponentes\_players\_moves)

The second heuristic tries to reduce the number of opponent's moves more aggressively by multiplying it with a weight of 1.5:

def custom\_score\_2(game, player):

if game.is\_loser(player):

return float("-inf")

if game.is\_winner(player):

return float("inf")

number\_of\_active\_players\_moves = len(game.get\_legal\_moves(player))

number\_of\_opponent\_players\_moves = len(game.get\_legal\_moves(game.get\_opponent(player)))

return number\_of\_active\_players\_moves - 1.5 \* number\_of\_opponent\_players\_moves

The third heuristic takes an even more aggresive approach by squaring the number of opponent's moves:

def custom\_score\_3(game, player):

if game.is\_loser(player):

return float("-inf")

if game.is\_winner(player):

return float("inf")

number\_of\_active\_players\_moves = len(game.get\_legal\_moves(player))

number\_of\_opponents\_players\_moves = len(game.get\_legal\_moves(game.get\_opponent(player)))

return number\_of\_active\_players\_moves - number\_of\_opponents\_players\_moves^2

To evaluate the scoring functions, 5 tournaments (T1-T2) were simulated with the following results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | AB\_Improved | AB\_Custom | AB\_Custom\_2 | AB\_Custom\_3 |
| Win rates T1 | 51.4% | 54.3% | 67.1% | 58.6% |
| Win rates T2 | 65.7% | 64.3% | 65.7% | 60.0% |
| Win rates T3 | 64.3% | 58.6% | 75.7% | 62.9% |
| Win rates T4 | 61.4% | 67.1% | 67.1% | 55.7% |
| Win rates T5 | 60.0% | 60.0% | 71.4% | 57.1% |
| Mean | 60.56% | 60.86% | 69.4% | 58.86% |

Table 1 Tournament results

Table 1 shows that the second heuristic not only achieves a significantly higher winning rate than the other two custom heuristics, but also generates good results over all 5 setups, without any negative outlier. Since it also clearly beats the AB\_Improved benchmark, it is recommended to use the second heuristic.