**Heuristics Analysis (AIND Isolation)**

**Heuristic functions implemented**

1. custom\_score – A basic variation of improved score. It calculates the difference between the number of own player moves and opponent player moves. It wights the number of opponent player moves with the factor 2.

own\_moves = len(game.get\_legal\_moves(player))

opp\_moves = len(game.get\_legal\_moves(game.get\_opponent(player)))

return float(own\_moves - 2\*opp\_moves)

1. custom\_score\_2 - This function is the improved center\_score in sample\_players. It calculates the difference between own player distance and opponent distance from center.

w, h = game.width / 2., game.height / 2.

y, x = game.get\_player\_location(player)

own\_moves\_from\_center = float((h - y)\*\*2 + (w - x)\*\*2)

a, b = game.get\_player\_location(game.get\_opponent(player))

opp\_moves\_from\_center = float((h - a)\*\*2 + (w - b)\*\*2)

return float(opp\_moves\_from\_center - own\_moves\_from\_center)

1. custom\_score\_3 - This function calculate the distance between own player and opponent player.

own\_moves = game.get\_legal\_moves(player)

opp\_position = game.get\_player\_location(game.get\_opponent(player))

mid\_width = float(game.width / 2)

mid\_height = float(game.height / 2)

score = 0

for move in own\_moves :

score = score + (move[0] - opp\_position[0])\*\*2 + (move[1] - opp\_position[1])\*\*2

score = 1 / max(score,100)

return score

**Agent Information**

1. Random: Agent that randomly choose a move each turn
2. MM\_Null: Agent using fixed-depth mini-max search and the null\_score function
3. MM\_Open: Agent using fixed-depth mini-max search and the open\_move\_score function
4. MM\_Improved: Agent using fixed-depth mini-max search and the improved\_score function
5. AB\_Null: Agent using fixed-depth alpha-beta search and the null\_score function
6. AB\_Open: Agent using fixed-depth alpha-beta search and the open\_move\_score function
7. AB\_Improved: Agent using fixed-depth alpha-beta search and the improved\_score function

**Results**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Match** | **Opponent** | **AB\_Improved** | | **AB\_Custom** | | **AB\_Custom2** | | **AB\_Custom3** | |
|  |  | **Won** | **Lost** | **Won** | **Lost** | **Won** | **Lost** | **Won** | **Lost** |
| 1 | Random | 10 | 0 | 6 | 4 | 9 | 1 | 10 | 0 |
| 2 | MM\_Open | 8 | 2 | 9 | 1 | 7 | 3 | 7 | 3 |
| 3 | MM\_Null | 10 | 0 | 10 | 0 | 10 | 0 | 9 | 1 |
| 4 | MM\_Improved | 8 | 2 | 9 | 1 | 9 | 1 | 8 | 2 |
| 5 | AB\_Null | 5 | 5 | 5 | 5 | 4 | 6 | 4 | 6 |
| 6 | AB\_Open | 5 | 5 | 7 | 3 | 6 | 4 | 5 | 5 |
| 7 | AB\_Improved | 5 | 5 | 8 | 2 | 4 | 6 | 5 | 5 |
|  | **Win Rate :** | **72.9%** | | **77.1%** | | **70.0%** | | **68.6%** | |

**Recommendations**

custom\_score function is recommended based on playing results because -

1. It is relatively simple to implement.
2. If we would like to improve this function, we could tune using grid search for this function:

A\*own\_moves - B\*opp\_moves

1. This function has the highest Win Rate.