Heuristic analysis

All my Heuristic based on the own\_moves and opp\_moves, here is how to compute: 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)))

As we know, the point of isolation is to illuminate the opponent's moves. A good evaluation function could be: my moves - opponents moves, it help us seek the most options while trying to get in the way of the opponent's moves. Moreover, increase the weight of my moves would help us seek most options regardless the opponent's behavior, increase the weight of opponent's move would lead agent chase after the opponent's behavior. Suppose the weight of my moves is Wm, weight of opponent's move is Wo, we are now considering the tradeoff between Wm and Wo.

In this casem I tried the following heuristic: custom\_score：

return float(own\_moves - opp\_moves)

custom\_score\_2：

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

custom\_score\_3：

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

custom\_score\_4：

return float(4\*own\_moves - opp\_moves)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Playing Matches

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Match # Opponent AB\_Improved AB\_Custom AB\_Custom\_2 AB\_Custom\_3 AB\_Custom\_4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Lost |  | Won | | Lost | Won | | | Lost | Won | | | Lost | Won | | | Lost | Won | | |
| 1 | Random | 7 | | 3 | 8 | | | 2 | 6 | | | 4 | 8 | | | 2 | 10 | | |
| 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | MM\_Open | 5 | | 5 | 8 | | | 2 | 8 | | | 2 | 7 | | | 3 | 6 | | |
| 4 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 | MM\_Center | 7 | | 3 | 8 | | | 2 | 9 | | | 1 | 6 | | | 4 | 5 | | |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 4 | MM\_Improved | 5 | | 5 | 7 | | | 3 | 9 | | | 1 | 4 | | | 6 | 8 | | |
| 2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 5 | AB\_Open | 7 | | | 3 | 5 | | | 5 | 4 | | | 6 | 4 | | | 6 | 5 | | |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 6 | AB\_Center | 7 | | | 3 | 5 | | | 5 | 6 | | | 4 | 8 | | | 2 | 5 | | |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 7 | AB\_Improved | 6 | | | 4 | 6 | | | 4 | 5 | | | 5 | 7 | | | 3 | 5 | | |

5

---------------------------------------------------------------------------------

----

Win Rate: 62.9% 67.1% 67.1% 62.9%

62.9%

We can easily find either too large or too small Wm/Wo would decrease the win rate.

Consider Custom 1-4, we can easily find it is almost symmetry. It would be better use Wm/Wo equal to sqrt(2).

To make calculation simple, I would use the following heuritic to test:

custom\_score：

return float(3\*own\_moves - 2\*opp\_moves)

custom\_score\_2：

return float(3\*own\_moves - 2\*opp\_moves + 1)

custom\_score\_3：

return float(3\*own\_moves - 2\*opp\_moves - 1)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Playing Matches

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Match # Opponent AB\_Improved AB\_Custom AB\_Custom\_2 AB\_Custom\_3

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Won | | | Lost | Won | | | Lost | Won | | | Lost | Won | | | Lost |
| 1 | Random 8 | | | 2 | 9 | | | 1 | 8 | | | 2 | 9 | | | 1 |
| 2 | MM\_Open 7 | | | 3 | 8 | | | 2 | 8 | | | 2 | 7 | | | 3 |
| 3 | MM\_Center 9 | | | 1 | 9 | | | 1 | 7 | | | 3 | 8 | | | 2 |
| 4 | MM\_Improved 5 | | | 5 | 5 | | | 5 | 8 | | | 2 | 7 | | | 3 |
| 5 | AB\_Open 6 | | | 4 | 6 | | | 4 | 6 | | | 4 | 6 | | | 4 |
| 6 | AB\_Center 3 | | | 7 | 7 | | | 3 | 5 | | | 5 | 6 | | | 4 |
| 7 | AB\_Improved 5 | | | 5 | 5 | | | 5 | 4 | | | 6 | 4 | | | 6 |

-------------------------------------------------------------------------- Win Rate: 61.4% 70.0% 65.7% 67.1%

We can easily find Wm/Wo = 1.5 get the best result, winning rate 70%