Heuristic analysis

TREBUCHET CLEMENT

AIND-Isolation

Introduction

In this project we have to implement an adversarial search agent for playing to the isolation game.
In this version of the isolation game each agent is restricted to L-shaped moves on a square grid of 7 x 7. A time limit is fixed for each player in order to search the best possible move to apply to the next play. The player who has reached the time limit loss the match by forfeits.

Heuristic implementation

Heuristic 1 : custom_score

increase #own_moves
The custom_score function try to maximize the own moves.

```
  sp1 = len(game.get_legal_moves(player))
  sp2 = len(game.get_legal_moves(game.get_opponent(player)))
 m × sp1 − sp2, with m ∈ ℝ
```

Heuristic 2: custom_score_2

reduce #opponent_moves:
The cutom_score_2 function try to reduce the opponent move

```
  sp2 = len(game.get_legal_moves(game.get_opponent(player)))
   − sp2
```

Heuristic 3: custom_score_3

decrease #opponent_moves
The custom_score_3 function try to decrease the opponent moves.

```
  sp1 = len(game.get_legal_moves())
  sp2 = len(game.get_opponent(player)))
```

sp1 $- m \times$ sp2, with $m \in \mathbb{R}$

```
```

Evaluation of the heuristic implementation

In order to test whether the implementation of the heuristic functions is effective, the tournament module is used.
The functions are tested in competition with the heuristic function improved_score, which is performed by an agent using the alpha-beta search and the iterative deepening search algorithm.

Opponents are materialized by three agents to evaluate different heuristics.
The first agent "Random" does not implement an heuristic function.
The second 'MinMax' agent implements the MinMax search algorithm.
The third agent 'AlphaBeta' implements the alphabeta pruning search algorithm.
Three heuristics are evaluated by MinMax and AlphaBeta, improved, central movement and open movement.

The result of tournament execution is shown in the table below.

In order to choose the coefficient m, I proceed to three tests with the values:

Test 1 m = 1,5
Test 2 m = 2
Test 3 m = 3

As indicated in the project the most efficient heuristic function has been implemented in "AB_custom" which surpasses the result of "AB_improved" with any coefficient of m.

Coefficient m equal to 1.5

```
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      20  |   0    16  |   4    20  |   0    19  |   1
    2       MM_Open     15  |   5    15  |   5    15  |   5    12  |   8
    3      MM_Center    15  |   5    20  |   0    18  |   2    16  |   4
    4     MM_Improved   15  |   5    15  |   5    12  |   8    13  |   7
    5       AB_Open     12  |   8    11  |   9     7  |  13    10  |  10
    6      AB_Center    11  |   9    14  |   6    13  |   7     9  |  11
    7     AB_Improved    8  |  12    11  |   9    11  |   9     9  |  11
--------------------------------------------------------------------------
           Win Rate:      68.6%        72.9%        68.6%        62.9%

Process finished with exit code 0
```

Coefficient m equal to 2

```
                        *************************
                             Playing Matches
                        *************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      16  |   4    19  |   1    17  |   3    18  |   2
    2       MM_Open     19  |   1    15  |   5    15  |   5    16  |   4
    3      MM_Center    18  |   2    19  |   1    15  |   5    16  |   4
    4     MM_Improved   15  |   5    14  |   6    13  |   7    15  |   5
    5       AB_Open     10  |  10    12  |   8    12  |   8    12  |   8
    6      AB_Center    14  |   6    14  |   6    14  |   6    10  |  10
    7     AB_Improved   10  |  10    10  |  10    11  |   9    11  |   9
--------------------------------------------------------------------------
           Win Rate:      72.9%        73.6%        69.3%        70.0%

There were 4.0 timeouts during the tournament -- make sure your agent handles s
```

Coefficient m equal to 3

```
                    *************************
                         Playing Matches
                    *************************

 Match #   Opponent    AB_Improved   AB_Custom    AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      19  |   1    19  |   1    18  |   2    19  |   1
    2       MM_Open     12  |   8    13  |   7    12  |   8    14  |   6
    3      MM_Center    17  |   3    17  |   3    18  |   2    19  |   1
    4     MM_Improved   15  |   5    15  |   5    14  |   6    14  |   6
    5       AB_Open      9  |  11    10  |  10     8  |  12    11  |   9
    6      AB_Center    11  |   9    13  |   7     9  |  11     9  |  11
    7     AB_Improved   13  |   7    15  |   5    12  |   8    11  |   9
--------------------------------------------------------------------------
           Win Rate:      68.6%        72.9%        65.0%        69.3%

Process finished with exit code 0
```

Performance of agents using the implemented evaluation functions

Based on the best of the three tests (done with 1.5 as the coefficient).

Agent	Win Rate
AB_Improved	68.6 %
AB_Custom with m = 1.5	**72.9 %**
AB_Custom_2	68.6 %
AB_Custom_3 with m = 1.5	62.9 %

Recommendation

I would recommend to use the ab_custom with 1.5 as coefficient, this function try to maximize the player moves.

1. AB_custom has increased its success rate by 4.3% compared to the results of the AB_Improved heuristics function.
2. The ratio between efficiency and execution time is good.
3. It can be implemented simply.
4. It confirmed that maximizing the players moves give more chances of winning.