Analysis on Heuristic Evaluation Function

By Chen Wen Shuo

Overview

This analysis is fulfilled on different heuristics to decide which evaluation function is the best to use.

Heuristic Evaluation function #1 : Result & Analysis

```
************************* Evaluating: AB_Improved *************************

Playing Matches:

 ----------

Match 1: AB_Improved vs Random     Result: 19 to 1

Match 2: AB_Improved vs MM_Open     Result: 14 to 6

Match 3: AB_Improved vs MM_Center     Result: 17 to 3

Match 4: AB_Improved vs MM_Improved     Result: 17 to 3

Match 5: AB_Improved vs AB_Open    Result: 11 to 9

Match 6: AB_Improved vs AB_Center     Result: 11 to 9

Match 7: AB_Improved vs AB_Improved     Result: 12 to 8


Results:
----------
AB_Improved         72.1%


************************* Evaluating: Student *************************

Playing Matches:

----------

Match 1: Student vs Random     Result: 19 to 1

Match 2: Student vs MM_Open     Result: 16 to 4

Match 3: Student vs MM_Center     Result: 17 to 3
Match 4: Student vs MM_Improved     Result: 15 to 5

Match 5: Student vs AB_Open     Result: 9 to 11

Match 6: Student vs AB_Center     Result: 10 to 10

Match 7: Student vs AB_Improved     Result: 9 to 11


Results:
----------
Student        67.9%
```

Implementation of Heuristic Evaluation Function #1

```
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)))


    return float(own_moves - 2*opp_moves)
```

Analysis:

This heuristic counts the moves of the player and the moves of the opponent multiply by 2, which means moves of the opponent is more important.

Heuristic Evaluation function #2 : Result & Analysis

```
************************* Evaluating: AB_Improved *************************

Playing Matches:

 ----------

Match 1: AB_Improved vs Random     Result: 18 to 2

Match 2: AB_Improved vs MM_Open     Result: 17 to 3

Match 3: AB_Improved vs MM_Center     Result: 18 to 2

Match 4: AB_Improved vs MM_Improved     Result: 15 to 5

Match 5: AB_Improved vs AB_Open    Result: 10 to 10

Match 6: AB_Improved vs AB_Center     Result: 10 to 10

Match 7: AB_Improved vs AB_Improved     Result: 10 to 10


Results:
----------
AB_Improved         70.0%


************************* Evaluating: Student *************************

Playing Matches:

----------

Match 1: Student vs Random     Result: 17 to 3

Match 2: Student vs MM_Open     Result: 14 to 6

Match 3: Student vs MM_Center     Result: 19 to 1

Match 4: Student vs MM_Improved     Result: 15 to 5

Match 5: Student vs AB_Open    Result: 11 to 9

Match 6: Student vs AB_Center     Result: 11 to 9

Match 7: Student vs AB_Improved     Result: 12 to 8


Results:
----------
Student         70.7%
```

Implementation of Heuristic Evaluation Function #2

```
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)))
    approx_depth = 49 - len(game.get_blank_spaces())
    return float(own_moves - opp_moves) * (1 + approx_depth*0.01)
```

Analysis:

This heuristic counts the moves of the player, opponent’s moves, and the depth for the current value. If some moves which have same player’s moves and opponent’s moves, we’d rather select the higher depth because toward the end of the game, the difference between moves will have more influence on the endgame.

Heuristic Evaluation function #3 : Result & Analysis

```
************************* Evaluating: AB_Improved *************************

Playing Matches:

 ----------

Match 1: AB_Improved vs Random      Result: 19 to 1

Match 2: AB_Improved vs MM_Open      Result: 15 to 5

Match 3: AB_Improved vs MM_Center      Result: 17 to 3
Match 4: AB_Improved vs MM_Improved     Result: 14 to 6

Match 5: AB_Improved vs AB_Open     Result: 11 to 9

Match 6: AB_Improved vs AB_Center     Result: 13 to 7

Match 7: AB_Improved vs AB_Improved     Result: 11 to 9


Results:
----------
AB_Improved         71.4%


************************* Evaluating: Student *************************

Playing Matches:

----------

Match 1: Student vs Random     Result: 19 to 1

Match 2: Student vs MM_Open     Result: 18 to 2

Match 3: Student vs MM_Center     Result: 18 to 2

Match 4: Student vs MM_Improved     Result: 18 to 2

Match 5: Student vs AB_Open     Result: 10 to 10

Match 6: Student vs AB_Center     Result: 12 to 9

Match 7: Student vs AB_Improved     Result: 10 to 10


Results:
----------
Student         75.0%
```

Implementation of Heuristic Evaluation Function #3

```
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)))
approx_depth = 49 - len(game.get_blank_spaces())
return float(own_moves - opp_moves) * (1 + approx_depth*0.03)
```

Analysis:

This heuristic counts the moves of the player, opponent's moves, and the depth for the current value. It increases the parameter from 0.01 to 0.03, placing more importance on the depth.

Summary

	The chance of winning (AB_IMPROVED)	The chance of winning (STUDENT)
Heuristic #1	72.1%	67.9%
Heuristic #2	70.0%	70.7%
Heuristic #3	71.4%	75.0%

The best performance: Heuristic #3

The worst performance: Heuristic #1

The best performance has about 8 % of the chance of the winning over the worst performance.

Conclusion

We recommends using Heuristic Evaluation Function #3, because

1) it counts for opponent's move.
2) it counts for depth. Counting depth keeps the play competitive (ref.

Heuristic #3 Analysis)

3) it adjusts the parameter which gives better performance.