Analysis on Heuristic Evaluation Function

By jae Min Baek

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: ID_Improved
*************************

Playing Matches:
----------
 Match 1: ID_Improved vs   Random     Result: 17 to 3
 Match 2: ID_Improved vs   MM_Null    Result: 15 to 5
 Match 3: ID_Improved vs   MM_Open    Result: 12 to 8
 Match 4: ID_Improved vs MM_Improved  Result: 10 to 10
 Match 5: ID_Improved vs   AB_Null    Result: 15 to 5
 Match 6: ID_Improved vs   AB_Open    Result: 14 to 6
 Match 7: ID_Improved vs AB_Improved  Result: 11 to 9


Results:
----------
ID_Improved        67.14%

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

Playing Matches:
----------
 Match 1:   Student   vs   Random     Result: 13 to 7
 Match 2:   Student   vs   MM_Null    Result: 17 to 3
 Match 3:   Student   vs   MM_Open    Result: 12 to 8
 Match 4:   Student   vs MM_Improved  Result: 11 to 9
 Match 5:   Student   vs   AB_Null    Result: 14 to 6
 Match 6:   Student   vs   AB_Open    Result: 12 to 8
 Match 7:   Student   vs AB_Improved  Result: 9 to 11


Results:
----------
Student            62.86%
```

Implementation of Heuristic Evaluation Function #1

```
def evaluation_function1(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))
        return float(own_moves)
```

Analysis:

This heuristic is to select the move which has the most moves available for the player. It doesn't count the opponent's moves, therefore it's unfavorable to defeat the opponent.

Heuristic Evaluation function #2 : Result & Analysis

```
*************************
Evaluating: ID_Improved
*************************

Playing Matches:
----------
 Match 1: ID_Improved vs   Random     Result: 16 to 4
 Match 2: ID_Improved vs   MM_Null    Result: 15 to 5
 Match 3: ID_Improved vs   MM_Open    Result: 12 to 8
 Match 4: ID_Improved vs MM_Improved  Result: 12 to 8
 Match 5: ID_Improved vs   AB_Null    Result: 14 to 6
 Match 6: ID_Improved vs   AB_Open    Result: 13 to 7
 Match 7: ID_Improved vs AB_Improved  Result: 12 to 8


Results:
----------
ID_Improved        67.14%

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

Playing Matches:
----------
 Match 1:   Student   vs   Random     Result: 16 to 4
 Match 2:   Student   vs   MM_Null    Result: 18 to 2
 Match 3:   Student   vs   MM_Open    Result: 11 to 9
 Match 4:   Student   vs MM_Improved  Result: 11 to 9
 Match 5:   Student   vs   AB_Null    Result: 15 to 5
 Match 6:   Student   vs   AB_Open    Result: 12 to 8
 Match 7:   Student   vs AB_Improved  Result: 11 to 9


Results:
----------
Student            67.14%
```

Implementation of Heuristic Evaluation Function #2

```
def evaluation_function2(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 - opp_moves)
```

Analysis:

This heuristic counts the moves of the player and the opponent, but it doesn't know about the game's position and depth.

Heuristic Evaluation function #3 : Result & Analysis

```
*************************
Evaluating: ID_Improved
*************************

Playing Matches:
----------
 Match 1: ID_Improved vs   Random     Result: 19 to 1
 Match 2: ID_Improved vs   MM_Null    Result: 12 to 8
 Match 3: ID_Improved vs   MM_Open    Result: 14 to 6
 Match 4: ID_Improved vs MM_Improved  Result: 13 to 7
 Match 5: ID_Improved vs   AB_Null    Result: 12 to 8
 Match 6: ID_Improved vs   AB_Open    Result: 11 to 9
 Match 7: ID_Improved vs AB_Improved  Result: 13 to 7


Results:
----------
ID_Improved        67.14%

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

Playing Matches:
----------
 Match 1:   Student   vs   Random     Result: 18 to 2
 Match 2:   Student   vs   MM_Null    Result: 16 to 4
 Match 3:   Student   vs   MM_Open    Result: 15 to 5
 Match 4:   Student   vs MM_Improved  Result: 11 to 9
 Match 5:   Student   vs   AB_Null    Result: 16 to 4
 Match 6:   Student   vs   AB_Open    Result: 12 to 8
 Match 7:   Student   vs AB_Improved  Result: 12 to 8


Results:
----------
Student            71.43%
```

Implementation of Heuristic Evaluation Function #3

```
def evaluation_function3(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)))
        # if the depth between moves are different and the scores is same, we'd rather choose the move which has not deeper,
        because it can finish the game earlier.

        # we can get the depth from the remaining spaces.
        approx_depth = 50 - len(game.get_blank_spaces())
        # i.c) 49 remaining, depth is 1 => 0.01
        #      46 remaining, depth is 4 => 0.04
        #      we'd rather take depth of 1.

        # we don't want to affect the own_moves and opp_moves decision so depth is less than an one.
        return float(own_moves - opp_moves - approx_depth*0.01)
```

Analysis:

This heuristic counts the player’s moves, opponent’s moves, and the depth for the current value. If we have some moves which have same player’s moves and opponent’s moves, we’d rather select the lower depth because it increases the chance of the winning by finishing the game early.

Heuristic Evaluation function #4 : Result & Analysis

```
*************************
Evaluating: ID_Improved
*************************

Playing Matches:
----------
 Match 1: ID_Improved vs   Random      Result: 16 to 4
 Match 2: ID_Improved vs   MM_Null     Result: 14 to 6
 Match 3: ID_Improved vs   MM_Open     Result: 11 to 9
 Match 4: ID_Improved vs MM_Improved  Result: 12 to 8
 Match 5: ID_Improved vs   AB_Null     Result: 12 to 8
 Match 6: ID_Improved vs   AB_Open     Result: 9 to 11
 Match 7: ID_Improved vs AB_Improved  Result: 12 to 8


Results:
----------
ID_Improved         61.43%

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

```
Playing Matches:
----------
 Match 1:   Student   vs   Random     Result: 19 to 1
 Match 2:   Student   vs   MM_Null    Result: 15 to 5
 Match 3:   Student   vs   MM_Open     Result: 15 to 5
 Match 4:   Student   vs MM_Improved  Result: 14 to 6
 Match 5:   Student   vs   AB_Null    Result: 14 to 6
 Match 6:   Student   vs   AB_Open     Result: 16 to 4
 Match 7:   Student   vs AB_Improved  Result: 12 to 8


Results:
----------
Student           75.00%
```

Implementation of Heuristic Evaluation Function #4

```
def evaluation_function4(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())

        # get the position array
        center_spaces = [(3, 3)]

        #if its depth is 3, it's ALWAYS better to move to the center position ( my assumption )
        center_value = 0
        #the game is set to pick random positions for players
        if approx_depth == 3:
          if game.get_player_location(player) in center_spaces:
             center_value = 99999
        return float(center_value + own_moves - opp_moves - approx_depth*0.01)
```

Analysis:

This heuristic counts the player’s moves, opponent’s moves, depth for the current value. If the player is available to select the very center position, it forces the player to take it. The center position makes the player have the inside track.

Heuristic Evaluation function #5 : Result & Analysis

****

```
Evaluating: ID_Improved
*************************

Playing Matches:
----------
 Match 1: ID_Improved vs   Random     Result: 15 to 5
 Match 2: ID_Improved vs   MM_Null    Result: 15 to 5
 Match 3: ID_Improved vs   MM_Open    Result: 12 to 8
 Match 4: ID_Improved vs MM_Improved  Result: 10 to 10
 Match 5: ID_Improved vs   AB_Null    Result: 14 to 6
 Match 6: ID_Improved vs   AB_Open    Result: 10 to 10
 Match 7: ID_Improved vs AB_Improved  Result: 11 to 9


Results:
----------
ID_Improved         62.14%

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

Playing Matches:
----------
 Match 1:   Student   vs   Random     Result: 14 to 6
 Match 2:   Student   vs   MM_Null    Result: 16 to 4
 Match 3:   Student   vs   MM_Open    Result: 14 to 6
 Match 4:   Student   vs MM_Improved  Result: 14 to 6
 Match 5:   Student   vs   AB_Null    Result: 16 to 4
 Match 6:   Student   vs   AB_Open    Result: 13 to 7
 Match 7:   Student   vs AB_Improved  Result: 13 to 7


Results:
----------
Student             71.43%
```

Implementation of Heuristic Evaluation Function #5

```
def evaluation_function5(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())

        center_spaces = [(3, 3)]

        center_value = 0
```

```
if approx_depth <= 5:
  if game.get_player_location(player) in center_spaces:
    center_value = 99999
return float(center_value + own_moves - opp_moves - approx_depth*0.01)
```

Analysis:

This heuristic counts the player’s moves, opponent’s moves, depth for the current value. It forced to take the center space until the third time of moves, however, it decreased the chance of the winning.

Summary

	The chance of the winning *(ID_IMPROVED)*	The chance of the winning *(STUDENT)*
Heuristic #1	67.14 %	62.86 %
Heuristic #2	67.14 %	67.14 %
Heuristic #3	67.14 %	71.43 %
Heuristic #4	61.43 %	75.00 %
Heuristic #5	62.14 %	71.43 %

The best performance: Heuristic #4
The worst performance: Heuristic #1

The best performance has about 13 % of the chance of the winning over the worst performance.

Conclusion

We recommends using Heuristic Evaluation Function #4, 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 gives positional advantage