Heuristic Analysis Report

In my report I would like to compare three evaluation functions:

1. Simple evaluation function where evaluation calculated like player_available_moves / all_blank_spaces.

```
def custom_score(game, player):

    player_available_moves = len(game.get_legal_moves(player))

    if game.is_loser(player):
        return 0.


    if game.is_winner(player):
        return 1.

    all_blank_spaces = len(game.get_blank_spaces())


    return player_available_moves / all_blank_spaces
```

2. Second evaluation function calculated like a sum of row position and column position.

```
def custom_score(game, player):

    if game.is_loser(player):
        return float("-inf")

    if game.is_winner(player):
        return float("inf")


    sum = 0

    for legal_move in game.get_legal_moves(player):
        x, y = legal_move

        sum += x + y

    return sum
```

3. In third function we using two different evaluation functions and changing from one function to another one in the middle of the game.

```
def custom_score(game, player):
    player_available_moves = len(game.get_legal_moves(player))

    opponent_available_moves =
len(game.get_legal_moves(game.get_opponent(player)))

    all_blank_spaces = len(game.get_blank_spaces())

    if game.is_loser(player):
        return float("-inf")
```

```
    if game.is_winner(player):
        return float("inf")

    if all_blank_spaces > all_blank_spaces/2:

        sum = 0

        for legal_move in game.get_legal_moves(player):
            x, y = legal_move

            sum += (x + y) / game.height

        return sum

    else:
        return player_available_moves - opponent_available_moves
```

Results of execution tournament.py for every evaluation function:

1. First evaluation function: ID_Improved 74.29% vs Student 88.57%.

```
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: 20 to 0
  Match 4: ID_Improved vs MM_Improved   Result: 20 to 0
  Match 5: ID_Improved vs   AB_Null     Result: 11 to 9
  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         74.29%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 19 to 1
  Match 2:   Student   vs   MM_Null     Result: 20 to 0
  Match 3:   Student   vs   MM_Open     Result: 20 to 0
  Match 4:   Student   vs MM_Improved   Result: 20 to 0
  Match 5:   Student   vs   AB_Null     Result: 15 to 5
  Match 6:   Student   vs   AB_Open     Result: 15 to 5
  Match 7:   Student   vs AB_Improved   Result: 15 to 5


Results:
----------
Student             88.57%
(aind) ➜  AIND-Isolation git:(master) ✗
```

2. Second evaluation function: ID_Improved 76.43% va Student 82.86%.

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

Playing Matches:
---------
 Match 1: ID_Improved vs   Random      Result: 16 to 4
 Match 2: ID_Improved vs   MM_Null     Result: 18 to 2
 Match 3: ID_Improved vs   MM_Open     Result: 20 to 0
 Match 4: ID_Improved vs MM_Improved   Result: 20 to 0
 Match 5: ID_Improved vs   AB_Null     Result: 13 to 7
 Match 6: ID_Improved vs   AB_Open     Result: 11 to 9
 Match 7: ID_Improved vs AB_Improved   Result: 9 to 11


Results:
---------
ID_Improved         76.43%

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

Playing Matches:
---------
 Match 1:   Student   vs   Random      Result: 18 to 2
 Match 2:   Student   vs   MM_Null     Result: 20 to 0
 Match 3:   Student   vs   MM_Open     Result: 20 to 0
 Match 4:   Student   vs MM_Improved   Result: 20 to 0
 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: 12 to 8


Results:
---------
Student             82.86%
```

3. Third evaluation function: ID_Improved 75% vs Student 81.43%.

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

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 15 to 5
  Match 2: ID_Improved vs   MM_Null     Result: 18 to 2
  Match 3: ID_Improved vs   MM_Open     Result: 20 to 0
  Match 4: ID_Improved vs MM_Improved   Result: 20 to 0
  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: 9 to 11


Results:
----------
ID_Improved         75.00%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 17 to 3
  Match 2:   Student   vs   MM_Null     Result: 19 to 1
  Match 3:   Student   vs   MM_Open     Result: 20 to 0
  Match 4:   Student   vs MM_Improved   Result: 20 to 0
  Match 5:   Student   vs   AB_Null     Result: 14 to 6
  Match 6:   Student   vs   AB_Open     Result: 11 to 9
  Match 7:   Student   vs AB_Improved   Result: 13 to 7


Results:
----------
Student             81.43%
```

Based on execution all evaluation functions showed constant win over ID_Improved evaluation function. But winner rate is different for every evaluation function. Let's consolidate results in single table:

	Function #1	**Function #2**	**Function #3**
Student vs Random	19 : 1	18 : 2	17 : 3
Student vs MM_Null	20 : 0	20 : 0	19 : 1
Student vs MM_Open	20 : 0	20 : 0	20 : 0
Student vs MM_Improved	20 : 0	20 : 0	20 : 0
Student vs AB_Null	15 : 5	14 : 6	14 : 6
Student vs AB_Open	15 : 5	12 : 8	11 : 9
Student vs AB_Improved	15 : 5	12 : 8	13 : 7
Win rate over ID_Improved	14.33	6.43	6.43

Based on these results we can see what all evaluation functions easily win in case of minimax algorithm, a little bit more difficult to win client random moves, but still easy. Things getting harder with alpha beta pruning, especially with improved evaluation function.

Custom functions #2 and #3 are equal in the table. But in most cases function #3 gives worse rate. Logic behind functions #2 is simple, instead of random selection of the move, we trying to fill up all available spaces in particular order. So in function #2 we starting fill up the board from right bottom corner and going up to the upper left corner. In the third function, we start the game with same approach, but after the half of the game, trying to select moves which give more available moves for us, but less for opponent. Based on results both evaluation functions work pretty well, but required a lot of computation, since we need to loop thought all legal moves to evaluate every game state. With these evaluation functions can be performance issues if size of the board will be increased.

Obviously, the absolutely leader function #1. Logic behind same like in *Improved* evaluation function. We trying select moves where more moves available for us, and less for opponent with small improvement, initial moves less important, so initial steps will have less score in comparison with moves in the end of the game. In addition, function #1 very simple, and will perform the same with any Board size. In addition, function #1 provides more unique scores for moves, since we using number of available spaces on the board.