UDACITY

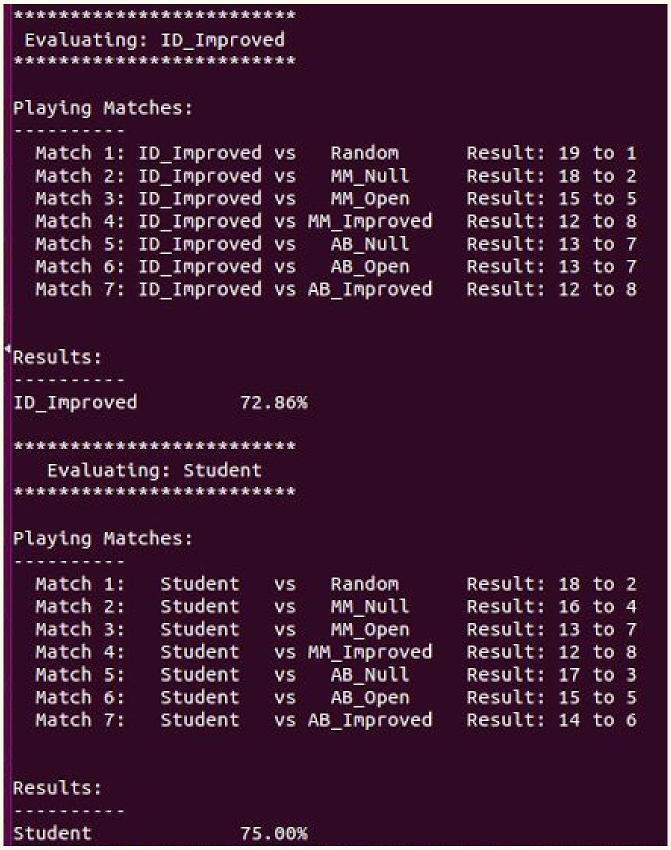
Heuristic Analysis

Gerardo de la Rosa

Heuristic Analysis

Heuristic 1

This heuristic gets the player moves and the opponent moves, which are respectively the amount of possible valid movements from the position of the computer and opponent at any state. On this evaluation function we chose 3 times the opponent moves in order to get an aggressive game. We got not the best result; but it improves the player movements.



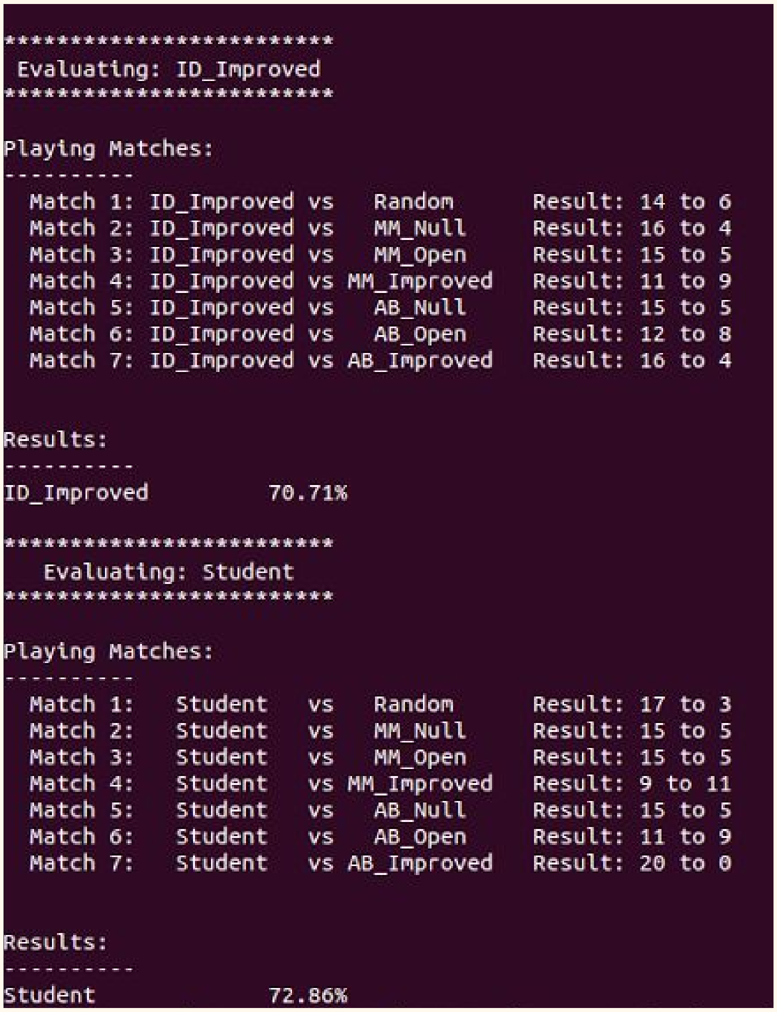
```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 19 to 1
  Match 2: ID_Improved vs   MM_Null     Result: 18 to 2
  Match 3: ID_Improved vs   MM_Open     Result: 15 to 5
  Match 4: ID_Improved vs MM_Improved   Result: 12 to 8
  Match 5: ID_Improved vs   AB_Null     Result: 13 to 7
  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         72.86%

*************************
   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: 13 to 7
  Match 4:   Student   vs MM_Improved   Result: 12 to 8
  Match 5:   Student   vs   AB_Null     Result: 17 to 3
  Match 6:   Student   vs   AB_Open     Result: 15 to 5
  Match 7:   Student   vs AB_Improved   Result: 14 to 6


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

Figure 1. Heuristic 1

```
my_moves = len(game.get_legal_moves(player))
opponent_moves = len(game.get_legal_moves(game.get_opponent(player)))
filled = (game.width * game.height) - len(game.get_blank_spaces())
return float((my_moves - 3 * opponent_moves) * filled)
```

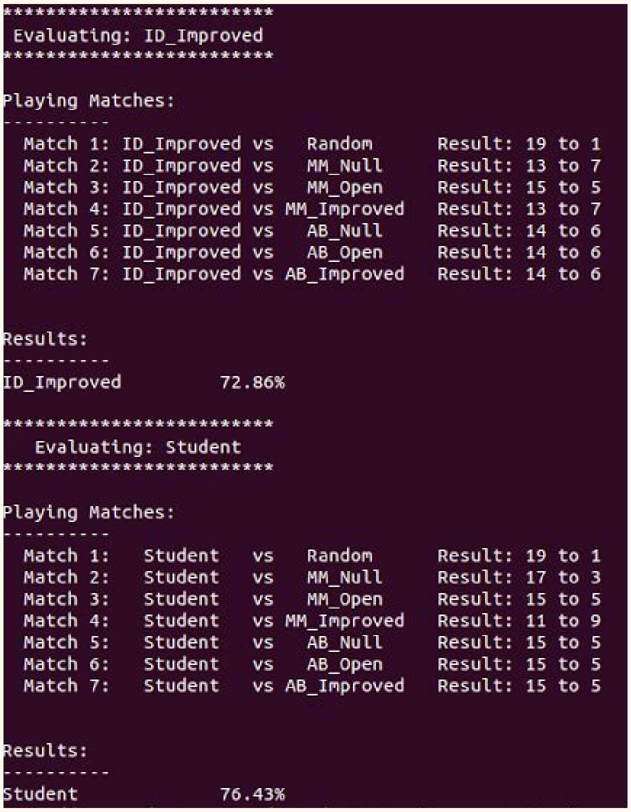
Heuristic 2

This heuristic function gets advantage from the opponent trying to choose moves that are near to the center of the board. I use the Manhattan distance to get which player has the highest score take it from all his legal movements that aren't touching the walls, that is because if the player stay on the center it will have more open moves and also will avoid to be cornered by the opponent. On this evaluation function I played with the multiplication factor on the opponent movements which got me a better result just until 2 multiplication factor; then the results got worse.



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

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 14 to 6
  Match 2: ID_Improved vs   MM_Null     Result: 16 to 4
  Match 3: ID_Improved vs   MM_Open     Result: 15 to 5
  Match 4: ID_Improved vs MM_Improved   Result: 11 to 9
  Match 5: ID_Improved vs   AB_Null     Result: 15 to 5
  Match 6: ID_Improved vs   AB_Open     Result: 12 to 8
  Match 7: ID_Improved vs AB_Improved   Result: 16 to 4


Results:
----------
ID_Improved         70.71%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 17 to 3
  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: 9 to 11
  Match 5:   Student   vs   AB_Null     Result: 15 to 5
  Match 6:   Student   vs   AB_Open     Result: 11 to 9
  Match 7:   Student   vs AB_Improved   Result: 20 to 0


Results:
----------
Student             72.86%
```

Figure 2. Heuristic 2 with not multiplication factor on opponent moves



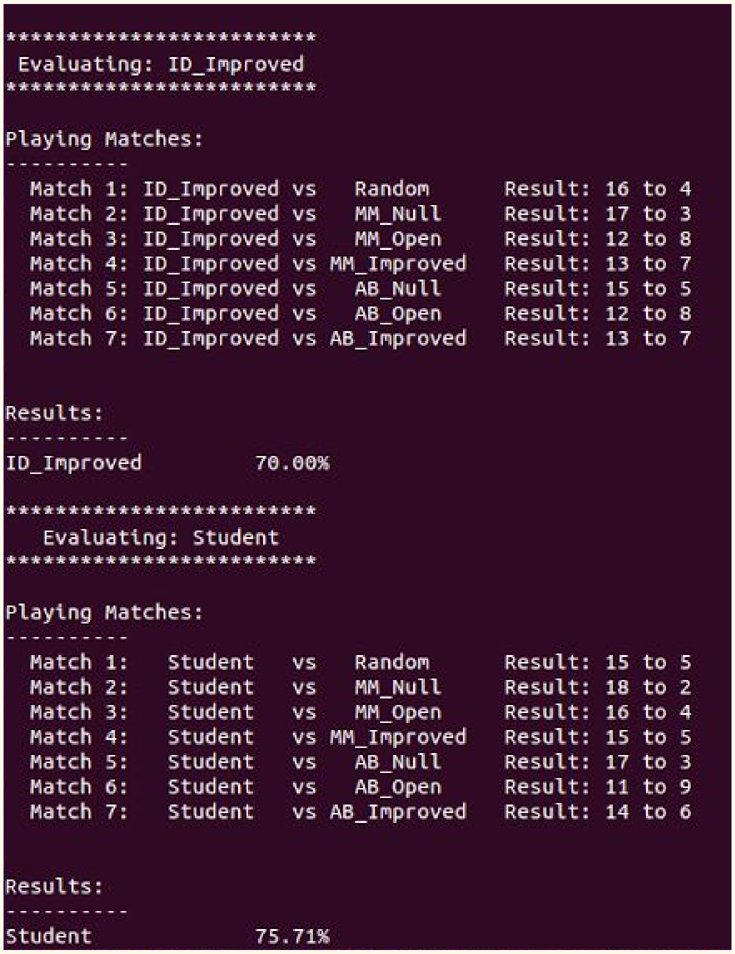
```
*************************
 Evaluating: ID_Improved
*************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 19 to 1
  Match 2: ID_Improved vs   MM_Null     Result: 13 to 7
  Match 3: ID_Improved vs   MM_Open     Result: 15 to 5
  Match 4: ID_Improved vs MM_Improved   Result: 13 to 7
  Match 5: ID_Improved vs   AB_Null     Result: 14 to 6
  Match 6: ID_Improved vs   AB_Open     Result: 14 to 6
  Match 7: ID_Improved vs AB_Improved   Result: 14 to 6


Results:
----------
ID_Improved         72.86%

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

Playing Matches:
----------
  Match 1:   Student   vs   Random      Result: 19 to 1
  Match 2:   Student   vs   MM_Null     Result: 17 to 3
  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: 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             76.43%
```

Figure 2. Heuristic 2 with not multiplication factor on opponent moves

```python
    walls = [
        [(0, i) for i in range(game.width)],
        [(i, 0) for i in range(game.height)],
        [(game.width - 1, i) for i in range(game.width)],
        [(i, game.height - 1) for i in range(game.height)]]

    # Remove moves that are touching the walls
    my_moves = [m for m in game.get_legal_moves(player) if m not in walls]
    opponent_moves = [m for m in game.get_legal_moves(game.get_opponent(player)) if m not in walls]
    my_moves_score = 0
    opponent_moves_score = 0
    # middle
    cx, cy = int(game.width / 2), int(game.height / 2)
    # Get Manhatan distance on each point that is not touching the walls to the center point
    for m in my_moves:
        p_x, p_y = m
        my_moves_score += abs(p_x - cx) + abs(p_y - cy)
    for m in opponent_moves:
        opp_x, opp_y = m
        opponent_moves_score += abs(opp_x - cx) + abs(opp_y - cy)
    return float(my_moves_score - 2 * opponent_moves_score)
```

Heuristic 3

This heuristic had better results; as long as I tried different ways to improve the evaluation function, I released that some techniques works better during different stages on the play time life. On this evaluation I decided to try three different functions in order to get the best using different techniques on 3 game stages.



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

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 16 to 4
  Match 2: ID_Improved vs   MM_Null     Result: 17 to 3
  Match 3: ID_Improved vs   MM_Open     Result: 12 to 8
  Match 4: ID_Improved vs MM_Improved   Result: 13 to 7
  Match 5: ID_Improved vs   AB_Null     Result: 15 to 5
  Match 6: ID_Improved vs   AB_Open     Result: 12 to 8
  Match 7: ID_Improved vs AB_Improved   Result: 13 to 7


Results:
----------
ID_Improved         70.00%

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

Playing Matches:
----------
  Match 1:   Student    vs   Random      Result: 15 to 5
  Match 2:   Student    vs   MM_Null     Result: 18 to 2
  Match 3:   Student    vs   MM_Open     Result: 16 to 4
  Match 4:   Student    vs MM_Improved   Result: 15 to 5
  Match 5:   Student    vs   AB_Null     Result: 17 to 3
  Match 6:   Student    vs   AB_Open     Result: 11 to 9
  Match 7:   Student    vs AB_Improved   Result: 14 to 6


Results:
----------
Student             75.71%
```

This first stage is when the filled cells percentage on the game is less than 30%; so at that moment the player will play an aggressive game.

```
if board <= 30:
    my_moves = len(game.get_legal_moves(player))
    opponent_moves = len(game.get_legal_moves(game.get_opponent(player)))
    return float((my_moves - 2 * opponent_moves))
```

The second stage is when the filled cells percentage on the game is less than 50 and greater than 30, on this stage the player will avoid the movements that touch the walls, that is because the player can be cornered by the opponent.

```
elif board > 30  and board <= 50:
    # Avoid walls
    # Remove moves that are on the walls
    my_moves = [m for m in game.get_legal_moves(player) if m not in walls]
    opponent_moves = [m for m in game.get_legal_moves(game.get_opponent(player)) if m not in walls]
    return float(len(my_moves) - len(opponent_moves))
```

The final stage involves to generate a dynamic partition and maximize the longest possible movements. So on this stage I get the players position; then I calculate the midpoint from both points on the board; once I have the midpoint I decide where does the partition can be created; the best option is between the two points; at the end I decide if the partition will be horizontal or vertical; this decision depends on the player's position.

```
  else:
    # Getting players position to create a partition
    p_x, p_y = game.get_player_location(player)
    opp_x, opp_y = game.get_player_location(game.get_opponent(player))
    blank_spaces = game.get_blank_spaces()
    # Check partition area
    # Get the midpoint to determinate the best area to generate the partition
    midpoint_x, midpoint_y = int((p_x + opp_x) / 2), int((p_y + opp_y) / 2)
    # Get players moves
    my_moves = game.get_legal_moves(player)
    opponent_moves = game.get_legal_moves(game.get_opponent(player))

    # Check if horizontal or vertical partition can be applied
    if p_x != opp_x:  # vertical
      # Remove moves that are out of the partition
      if midpoint_x < p_x:  # Right
        my_moves = [m for m in my_moves if m[0] > midpoint_x]
        opponent_moves = [m for m in opponent_moves if m[0] < midpoint_x]
      else:  # left
        my_moves = [m for m in my_moves if m[0] < midpoint_x]
        opponent_moves = [m for m in opponent_moves if m[0] > midpoint_x]
    else:  # horizontal
      # Remove moves that are out of the partition
      if midpoint_y < p_y:  # Up
```

```
            my_moves = [m for m in my_moves if m[1] > midpoint_y]
            opponent_moves = [m for m in opponent_moves if m[1] < midpoint_y]
        else:  # Down
            my_moves = [m for m in my_moves if m[1] < midpoint_y]
            opponent_moves = [m for m in opponent_moves if m[1] > midpoint_y]

    return float(len(my_moves) + len(opponent_moves))
```

Conclusion

To get an optimal evaluation function it is necessary a really good understanding about the game were heuristic will be applied and also to try with many possible combinations; open moves are quite good it help us to take better movements and it removes possible ones that are not the best option.