Analysis

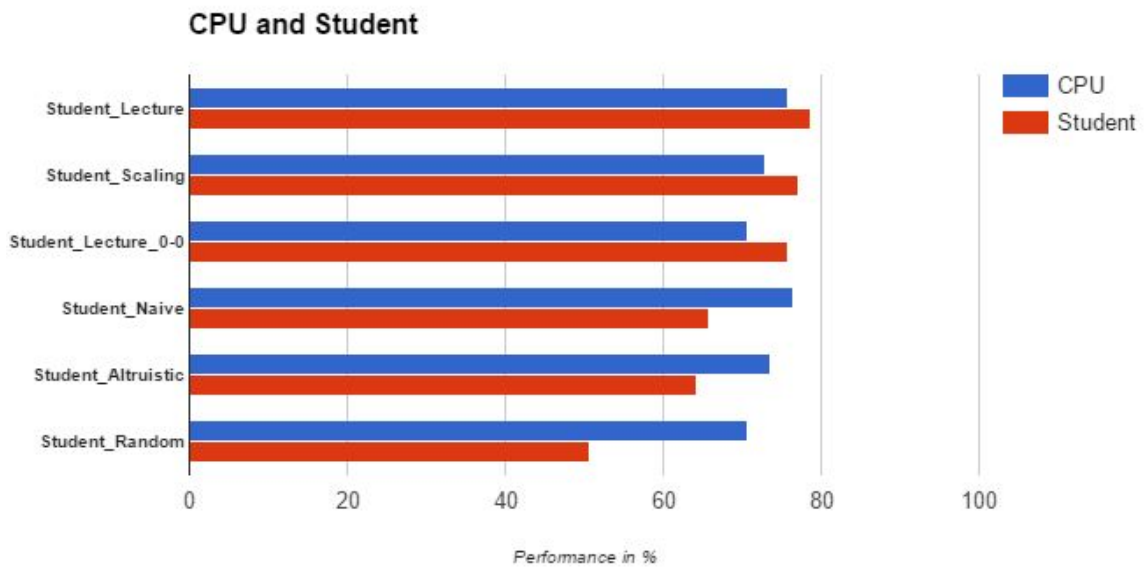
# Isolation

Andreas Mayer

3rd February, 2017

# Overview

This graphic shows the performance of the different scoring functions.




The following data was used to create the graphic.

| Function | CPU | Student |
|---|---|---|
| Student_Lecture | 75.71 | 78.57 |
| Student_Scaling | 72.86 | 77.14 |
| Student_Lecture_0-0 | 70.71 | 75.71 |
| Student_Naive | 76.43 | 65.71 |
| Student_Altruistic | 73.57 | 64.29 |
| Student_Random | 70.71 | 50.71 |

# Comparison of scoring functions

This chapter illustrates the different custom scoring functions that were evaluated and compared against each other and shows the tournament result aligned to the specific function.

I tried a hand full of scoring functions and documented my findings in this document.

## Random scoring function (center)

The player stars at the center when he has the first move.

The first test was done with a scoring function that returned random values for every node. As expected, the agent performed worse. In the lecture it was stated, that an optimal agent will always outperform an agent that uses no strategy or a non-optimal strategy.

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

Playing Matches:
----------
  Match 1: vs   Random      Result: 19 to 1
  Match 2: vs   MM_Null     Result: 15 to 5
  Match 3: vs   MM_Open     Result: 12 to 8
  Match 4: vs MM_Improved   Result: 10 to 10
  Match 5: vs   AB_Null     Result: 18 to 2
  Match 6: vs   AB_Open     Result: 15 to 5
  Match 7: vs AB_Improved   Result: 10 to 10

Results:
----------
        70.71%
```

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

Playing Matches:
----------
  Match 1: vs   Random      Result: 17 to 3
  Match 2: vs   MM_Null     Result: 13 to 7
  Match 3: vs   MM_Open     Result: 8 to 12
  Match 4: vs MM_Improved   Result: 8 to 12
  Match 5: vs   AB_Null     Result: 10 to 10
  Match 6: vs   AB_Open     Result: 6 to 14
  Match 7: vs AB_Improved   Result: 9 to 11

Results:
----------
Student             50.71%
```

## Simple scoring function (center)

The player stars at the center when he has the first move.

This improved scoring function simply returns the number of possible steps at the given node. As it does not put the opponent into consideration it is not surprising that the player lost against the computer agent. However, the score has increased related to the random scoring function, as moves are avoided that don't offer many next possible steps.

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

Playing Matches:
----------
  Match 1: vs   Random      Result: 20 to 0
  Match 2: vs   MM_Null     Result: 19 to 1
  Match 3: vs   MM_Open     Result: 13 to 7
  Match 4: vs MM_Improved   Result: 13 to 7
  Match 5: vs   AB_Null     Result: 15 to 5
  Match 6: vs   AB_Open     Result: 15 to 5
  Match 7: vs AB_Improved   Result: 12 to 8

Results:
----------
        76.43%
```

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

Playing Matches:
----------
  Match 1: vs   Random      Result: 18 to 2
  Match 2: vs   MM_Null     Result: 19 to 1
  Match 3: vs   MM_Open     Result: 13 to 7
  Match 4: vs MM_Improved   Result: 8 to 12
  Match 5: vs   AB_Null     Result: 12 to 8
  Match 6: vs   AB_Open     Result: 11 to 9
  Match 7: vs AB_Improved   Result: 11 to 9

Results:
----------
Student             65.71%
```

## Lecture scoring function (center)

The player stars at the center when he has the first move.

This test used the scoring function from the lecture, where situations that offer many moves to the opponent are penalized.

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

Playing Matches:
----------
  Match 1: vs   Random      Result: 20 to 0
  Match 2: vs   MM_Null     Result: 18 to 2
  Match 3: vs   MM_Open     Result: 17 to 3
  Match 4: vs MM_Improved   Result: 11 to 9
  Match 5: vs   AB_Null     Result: 17 to 3
  Match 6: vs   AB_Open     Result: 11 to 9
  Match 7: vs AB_Improved   Result: 12 to 8

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

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

Playing Matches:
----------
  Match 1: vs   Random      Result: 19 to 1
  Match 2: vs   MM_Null     Result: 18 to 2
  Match 3: vs   MM_Open     Result: 16 to 4
  Match 4: vs MM_Improved   Result: 12 to 8
  Match 5: vs   AB_Null     Result: 19 to 1
  Match 6: vs   AB_Open     Result: 14 to 6
  Match 7: vs AB_Improved   Result: 12 to 8

Results:
----------
Student             78.57%
```

## Altruistic scoring function (center)

The player stars at the center when he has the first move.

This scoring function assumes that situations where the enemy has more moves could lead to a winning situations. The idea was to prefer moves still provide some space for the opponent. The evaluation shows that this scoring function performs comparable to the naive scoring function.

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

Playing Matches:
----------
  Match 1: vs    Random      Result: 17 to 3
  Match 2: vs    MM_Null     Result: 20 to 0
  Match 3: vs    MM_Open     Result: 15 to 5
  Match 4: vs MM_Improved    Result: 13 to 7
  Match 5: vs    AB_Null     Result: 16 to 4
  Match 6: vs    AB_Open     Result: 12 to 8
  Match 7: vs AB_Improved    Result: 10 to 10

Results:
----------
        73.57%
```

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

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

Results:
----------
Student             64.29%
```

## Lecture scoring function (0,0)

The player stars at the coordinate (0,0) when he has the first move.

This test used the scoring function from the lecture, where situations that offer many moves to the opponent are penalized.

The change of the starting position should evaluate if the starting position has a major effect or not, as the set of moves in this variant of isolation is strongly constrained compared to original isolation.

The performance was comparable to the scoring function from the lecture, but performed a little bit worse. This could however be an effect of randomness.

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

Playing Matches:
----------
  Match 1: vs   Random      Result: 19 to 1
  Match 2: vs   MM_Null     Result: 19 to 1
  Match 3: vs   MM_Open     Result: 11 to 9
  Match 4: vs MM_Improved   Result: 14 to 6
  Match 5: vs   AB_Null     Result: 16 to 4
  Match 6: vs   AB_Open     Result: 11 to 9
  Match 7: vs AB_Improved   Result: 9 to 11

Results:
----------
        70.71%
```

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

Playing Matches:
----------
  Match 1: vs   Random      Result: 19 to 1
  Match 2: vs   MM_Null     Result: 18 to 2
  Match 3: vs   MM_Open     Result: 11 to 9
  Match 4: vs MM_Improved   Result: 14 to 6
  Match 5: vs   AB_Null     Result: 15 to 5
  Match 6: vs   AB_Open     Result: 14 to 6
  Match 7: vs AB_Improved   Result: 15 to 5

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

## Adapting scoring function (center)

The player stars at the center when he has the first move.

The idea of this scoring function is to give the opponent more space at the beginning of the match. Moves that offer more space to the opponent are penalized more and more while the game progresses. So when we get close to the end game, the enemy should not get possibilities to evolve his game, for free.

The performance of this scoring function was comparable to the scoring function used in the lecture.

$$score(move) = ownMoves - \frac{0.25 * numberOfMaxMoves}{numberOfMovesPerformed} * opponentMoves$$

As the performance is very close, I would avoid this function due to the highly increased complexity compared to the lecture scoring function.

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

Playing Matches:
----------
  Match 1: vs   Random      Result: 20 to 0
  Match 2: vs   MM_Null     Result: 18 to 2
  Match 3: vs   MM_Open     Result: 14 to 6
  Match 4: vs MM_Improved   Result: 15 to 5
  Match 5: vs   AB_Null     Result: 16 to 4
  Match 6: vs   AB_Open     Result: 11 to 9
  Match 7: vs AB_Improved   Result: 8 to 12


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

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

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


Results:
----------
Student             77.14%
```

# Conclusion

It turns out that besides the correct implementation of the algorithms, that enable the player to evaluate the search space in a directed manner, the scoring function (heuristic) is very important and will have a significant impact on the outcome of a game.

Scoring functions that penalize situations that could be advantageous for the opponent, perform better than scoring functions that completely ignore the opponent.

The complexity of the scoring function and its performance do not correlate with each other. A good scoring function could be super simple and effective at the same time whereas a complex, multi-factor function could perform much worse if it takes the wrong factors into consideration.

I would select the **heuristic from the lecture** because of the following reasons:

## Performance

In my tests it performed better than all the other heuristics (78.57%)

## Simplicity

```
def scoring_function_lecture(game, player):

  ownMoves = game.get_legal_moves(player).__len__()
  opponentMoves = game.get_legal_moves(game.get_opponent(player)).__len__()

  return ownMoves - 2. * opponentMoves
```

## Efficiency

The function can be executed very efficiently and will therefore not thwart the search too much.

| Operation | Complexity |
|---|---|
| `len()` | $C_{len} \times O(1)$ |
| `variable assignment` | $C_{assignment} \times O(1)$ |
| `calculation` | $C_{calculation} \times O(1)$ |
| `getter-calls` | $C_{getters} \times O(1)$ |