Heuristic Analysis

1. Description of heuristics

I've analyzed the following heuristics:

1.0. Basic heuristic.
Heuristic from the lecture: number of own legal moves minus number of opponents legal moves. It is used as a baseline and as a part of more complex heuristics.
Function in code: *basic_score*

1.1. Distance from center
Assumption: in cases, where basic heuristic is equal, it is better to be closer to the center of the field.
Function in code: *strategy_center*

1.2. Distance from opponent
Assumption: in cases, where you have more legal moves, it is better to be closer to your opponent. Else, you'd better be far from him
Function in code: *strategy_close_or_far*

1.3. Be closer to a blank spaces
Assumption: you should choose a place with a minimum sum of distances to blank spaces
Function in code: *strategy_free_field*

1.4. Be closer to a blank spaces
A modification of 1.1.
Assumption: in cases, where basic heuristic is equal, it is better to be closer to the center of the field than your opponent
Function in code: *strategy_closer_to_center*

2. Individual heuristic tests

In the individual test custom_score simply equal heuristic. Tournament.py is modified to play more games (NUM_MATCHES = 20)
The results of the individual tests are shown in the table.

Heuristic	ID_Impoved	Student	delta	Comment
basic_score	62,50%	62,70%	0,20%	Expected to be nearly equal
strategy_center	61,61%	66,43%	4,82%	
strategy_close_or_far	64,64%	67,86%	3,22%	
strategy_free_field	62,86%	60,89%	-1,97%	
strategy_closer_to_center	64,46%	68,39%	3,93%	

The results show that most of the heuristics are a bit better than basic score. Anyhow there is no "star" heuristic. Further improvements are needed.

3. Stage-based tests

It is possible, that some heuristics are better at the beginning of the game, and some at the end. I've implemented complex heuristic, which uses different heuristic functions at the different stages of the game:

```
def strategy_complex(game, player):

# Begginning of the game
    if len(game.get_blank_spaces()) > game.width*game.height*0.7:
            score = <heuristic1>

# Game middle
    elif len(game.get_blank_spaces()) > game.width*game.height*0.4:
        score = <heuristic2>

# Game ending
    else:
        score = <heuristic3>

    return score
```

Table shows performance of different heuristics at different stages

Heuristics	Stage	ID_Impoved	Student	delta
strategy_center	Beginning[1]	61,61%	62,50%	0,89%
	Middle[2]	60,9%	64,56%	3,66%
	Ending	63,04%	64,29%	1,25%
strategy_close_or_far	Beginning	63,21%	63,93%	0,72%
	Middle	64,82%	63,04%	-1,78%
	Ending	61,96%	63,39%	1,43%
strategy_free_field	Beginning	63,04%	65,54%	2,50%
	Middle	64,46%	62,68%	-1,78%
	Ending	61,07%	61,96%	0,89%
strategy_closer_to_center	Beginning	61,96%	59,82%	-2,14%
	Middle	63,39%	66,61%	3,22%
	Ending	62,14%	59,82%	-2,32%

[1] it means that I use strategy_center at the beginning, and strategy_basic at the middle and end.
[2] It means that I use strategy_center in the middle, and strategy_basic at the beginning and end. Etc.

Strategy_center looks good at the middle of the game and possibly **strategy_free_field** looks good at the beginning.

Lets try to combine it to complex heuristic:

At the beginning - **strategy_free_field**, at the middle - **strategy_center** and at the end – **basic strategy**.

The results are: 63,75% for ID_Improved, 64,64% for Student. Not very impressive.

4. Re-check of "strategy_center" heuristics

Strategy_center heuristics for now looks preferable, but does it consequently outperforms ID_Improved? To be confident I repeat a measurement with it again. The results are shown in the table:

Heuristic	ID_Impoved	Student	delta	Comment
strategy_center	61,61%	66,43%	4,82%	Previous measurement
strategy_center	61,07%	63,93%	2,86%	New measurement

5. Summary

1. It seems that an error in measurements is rather big. To reduce errors it is possible to increase NUM_MATCHES even more. For now, it is impossible due to the project deadline time limit.

2. For now, I choose **strategy_center** as a preferable heuristic because**:**

– it consequently outperforms ID_Improved (in fact it basics on the same heuristic, but with an improvement);

– it is easy to implement and compute, so it doesn't reduce the depth of the search too much;

– it can be reasonably used for other variations of Isolation game (not only for Knight movements).