# Heuristic functions analisys

Stanislav Levental

February 20, 2017

**Abstract**

This document includes comparison of different heuristics I've tried and reasoning for choosing one used in submission.

## 1 Heuristics

### 1.1 Baseline (improved score)

First and very intuitive approach was presented during lectures: using difference between number of player's moves and opponent's moves, this will allow search agent to discover moves towards maximizing this difference - which should lead to better choices. And this strategy worked very well for me. Results in *tournament.py* was about **80%**.

### 1.2 Weighted number of moves

Also I was tried suggested weighted amount of moves, giving more weight to opponent's moves will make agent to behave more aggressive, trying to minimize number of moves opponent can do. I've used weight equal to **1.5** it had better results according to my experiments (about **89%**).

### 1.3 Connected components

Also I've tried to add number of connected blank positions to the score, this intuitively will make agent to move to bigger islands of connected positions instead of just pursuing larger amount of moves. Union-Find algorithms was used to maintain size of connected parts of the board and maintain information about their sizes. Using this technique didn't make overall performance better, actually it became worse: **75%**. This was caused by changed rules of the game, where jumping over closed position was not allowed and

move pattern was changed from queen to knight. After some improvements: changing components generation function and let it use L-shape moves for joining blank component performance became much better - **85%**

### 1.4 Connected components combined with moves count

In some cases previous heuristic wasn't working, I'd determined that cases were related to state of the game where the whole board were connected and number of available joined components in both cases (player 1 & 2) were the same, this gave zero score and as a result - wrong initial moves were made. After adding condition which would back-off to previous heuristic when current one became zero, I've got even better results: **95%**. Also, I've tried to add weights to number of connected components (blank positions) to figure out if aggressive strategy will help improving overall performance, but any weights besides ones were working worse (less than **90%**).