Heuristic Opt. Techniques - Assignment 1 Report

Martin Blöschl and Cem Okulmus

Summary

Our group consisting of Blöschl Martin and Okulmus Cem was given the task of finding a heuristic for the K Page Book Embedding Problem. The problem description will not be repeated for sake of brevity. We chose a simple two-step algorithm that constructs a greedy heuristic with no further local search. As for the randomization, we found ways to extends these steps to allow for a parameterized uncertainty factor (similar to the α factor in GRASP). Our results showed that randomization improved our results for almost all our test instances.

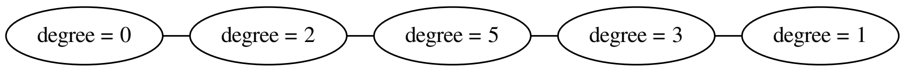
Implementation

Solution representation

Internally, we represent a solution as the order of its vertices, an integer array, and the lists of edges assigned to pages. We decided that our construction heuristic should first fix the vertex order and afterwards determine a good page partition for this ordering. This allowed us to find clear rules for the construction at each step, as we will explain in the following sections. A downside of this method is that it does not allow to avoid certain crossings encountered during the heuristic by changing the vertex order. This could be alleviated by adding a local search procedure, ideally searching for both changes in vertex ordering and page partition.

Deterministic Construction

Our deterministic construction looks at the degree of each vertex to determine its order. The idea here is to have vertices with higher degree nearer the middle of the "spine", as shown in the graph below.



The idea here is that since most edges will be incident to the degrees in the middle, there will be less crossings.

For the edge partitioning, we implemented a first fit algorithm that chooses the first page with which the new edge introduces no new crossings. If no such page exists, the one with the least introduced crossings is chosen.

Randomized construction

For the randomization, we tried to find meaningful ways to introduced a controlled randomization factor into each step, that does not automatically regress to a random search. For this purpose, we used a parameter α. $\alpha = 0.0$ means the algorithm proceeds as the deterministic case, with $\alpha = 1.0$ essentially reducing each to step to be almost completely random. Every value in between gradually changes the behavior.

For the vertex ordering, a random offset for the degree count is added. The vertices are then sorted by degree and offset together.

As for the edges, we first collect all edges to be added and then shuffle the order in which the first fit procedure from above is considered. We experimented with also adding edges randomly (depending on α), but the results proved not satisfiable.

Evaluation

Our tests were run mostly on our private machines. For Cem Okulmus this is a laptop with a Intel i7-4500U @ 1.8 GHz and 8 GB of DDR3 @ 1333 Mhz.

Our best results can be seen on the tool provided to submit solutions. We found that the best results were almost always achieved by the randomized variant (except for instance 10, where our best result stems from the deterministic version).

We assume the most obvious way to improve our results would be the implementation of a local search. Since we implemented an incremental way to detect crossings (the most computationally expensive part of our heuristic), we could quickly introduce changes to the page partition (though not the vertex order) and check for improvements.