Heuristic Opt. Techniques - Assignment 3 Report

Martin Blöschl and Cem Okulmus

1 Implementation

We decided to implement General Variable Neighborhood Search (GVNS). For shaking, we implemented a parametrized neighborhood N/M-swap, where n nodes are swapped with n other nodes and m edges are moved to a different page. Our set of neighborhood structures for shaking is then a set of N/M-swap neighborhoods with different parameters. The precise values used for n and m will be discussed in the evaluation part of this report. As an additional note, this neighbor is actually complete, in the sense that it would be theoretically possible to find every possible solution by iterative search.

For the VND, we use different combinations of One Edge Move, One Node Swap , Node Neighbor Swap (these were discussed in the last report) and another neighborhood called Edge Neighbor Move, introduced below.

1.1 Detailed description of additional neighborhoods

In addition to the neighborhoods we had so far, we also implemented two new neighborhoods.

1.1.1 Parametrized N/M-swap neighborhood

Until now, we only looked for solutions that changed a single pair of nodes in the spine order or moved a single edge to a new page. The new N/M-swap neighborhood takes two parameter from the range 1 to a certain maximal value. For edge moves, the maximal value is of course the number of all edges in the instance. Obviously, at most all of the edges can be changed at a single time. For node swaps, it is the number of nodes.

1.1.2 Edge Neighbor Move neighborhood

This is similar to “Node Neighbor Swap” from our previous report. It simply moves an edge to one of the the neighboring pages (so one higher index or one lower index modulo the number of pages). This neighborhood is used for doing GVNS in the larger instances, since it is significantly smaller than the One Edge Move neighbourhood (see last report for details).

1.2 Incremental evaluation

Like in the previous exercises, we used incremental evaluation when changing edges of a solution. In the first report we explained how the Active Edge Data structure works. Incremental evaluation is used in ”One Edge Move” and ”Edge Neighbor Move”. If the ordering of the nodes changes, the complete solution must be recalculated. Thus we cannot use incremental evaluation in the other neighborhood structures.

2 Evaluation

2.1 Order of the neighborhoods

To check if the order of the neighborhoods is important, we reversed the set of shaking neighborhoods, the set of VND neighborhoods and both of them at the same time.

All 4 combinations (original, shaking reversed, VND reversed, both reversed) where then executed 10 times for the Problem instance 5. The test was executed on the eowyn cluster. Unsurprisingly, the execution time of all 40 tests did not vary significantly (1,48 seconds mean, 0,30 st. deviation). The outcome of the crossing count is similar, it does not vary significantly between the 40 test runs (16,33 mean, 3,19 st. deviation). We thus conclude that the ordering of the neighborhoods does not matter.

2.1.1 Test setup

As with last time, we ran our tests on the eowyn cluster. Our experiments showed that the best results so far were accomplished by using for the VNS multiple N-M-swap neighborhoods, were we start with 1 and generate up to 20 variations with ever increasing parameters, thus "jumping" to ever farther points in the search space. The VND is done by using two smaller structures: One Edge Move and One Node Swap. We ran into a performance issue with the larger instances. Therefore, for the larger instances (6 to 10), we restricted the VNS to 10 variations of increasing N-M-swap neighborhoods, and the VND was simplified to Node Neighbor Swap and Edge Neighbor Move. This was done to stay under the time limit.

For the initial results, we found that while using completely random initial solutions showed the most relative improvement, the best results overall were achieved by starting from an already better solution from our randomized heuristics, which is explained in our first report.

To measure the possibly random results, we let each instance run 10 times and show the global best value that was found in any of the ten runs, the average and the sample standard deviation (computed as $s = \sqrt{\frac{1}{N-1} \sum_{1 \leq i \leq N} (x_i - \bar{x})}$ where $\bar{x}$ is the mean).

The results can be seen in figure 1.

3 Observations

The GVNS method allowed us to find very competitive (wrt. the solutions for assignment 2) results for instances 1 to 5. While there were some improvements on the larger instances, it would first require an even more performant method of evaluation to use GNVS with larger neighborhood structures (or simply run GNVS more often with more time).

Instance	Global best	Mean (10 runs)	std. dev.	runtime mean (sec.)	runtime std. dev.
automatic-1	9	9	0.0	0.43	0.02
automatic-2	1	5.5	3.56	1.11	0.13
automatic-3	45	49.6	2.63	2.66	1.44
automatic-4	3	9.9	6.10	1.71	0.63
automatic-5	7	18.7	6.48	1.56	0.22
automatic-6*	1834647	1939006.4	66431.69	132	3.88
automatic-7*	128777	130716.4	1138.15	20.71	3.70
automatic-8*	237711	247064	16468.9	100.89	5.84
automatic-9*	352161	359339	5061.8	89.47	2.70
automatic-10*	43807	46241	3477.7	128.45	4.13

Table 1: Test results for GVNS. Entries marked with Star (*) use different neighbourhoods as described in the report.