Heuristic Opt. Techniques - Assignment 3 Report

Martin Blöschl and Cem Okulmus

1 Implementation

We decided to implement General Variable Neighbourhood Search (GVNS). For shaking, we implemented a parameterized neighbourhood N/M-swap, where n nodes are swapped with n other nodes and m edges are moved to a different page. Our set of neighbourhood structures for shaking is then a set of N/M-swap neighbourhoods 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 neighbour 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 Neighbour Swap (these were discussed in the last report) and another neighbourhood called Edge Neighbour Move, introduced below.

1.1 Additional Neighbourhoods

In addition to the ones we had so far, we also implemented new neighbourhoods. Mostly these are just simple adaptions to make full use of GVNS.

1.1.1 Parametrized Neighbourhoods

Until now, we only looked for solutions that "swapped" or "moved" a single pair of nodes in the spine order or a single edge to a new page. These new structures take a parameter that ranges from 1 to a certain maximal value. For edge moves, the maximal value is of course the number of edges in the instance. For node swaps, it is half the number of nodes (rounded upwards to the nearest integer). The restriction is to make sure that no duplicate neighbours are produced.

1.1.2 Edge Neighbour Move Neighbourhood

This is similar to "Node Neighbour Swap" from our previous report. It simply moves an edge to both the neighbouring pages (so one higher and one lower wrt. modulo the number of pages). This neighbourhood is used for doing GVNS in the larger instances, since it is significantly smaller.

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 Neighbour Move". If the ordering of the nodes changes, the complete solution must be recalculated. Thus we cannot use incremental evaluation in the other neighbourhood structures.

2 Evaluation

2.1 Order of the neighbourhoods

To check if the order of the neighbourhoods is important, we reversed the set of shaking neighbourhoods, the set of VND neighbourhoods 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 neighbourhoods 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 neighbourhoods, 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 neighbourhoods, and the VND was simplified to Node Neighbour Swap and Edge Neighbour 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 solution 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 neighbourhood structures (or simply wait for far longer than 15 minutes per instance per solution).

Instance	Global best	Average (from 10)	standard Deviation
automatic-1	9	9	0.0
automatic-2	1	5.5	3.56
automatic-3	45	49.6	2.63
automatic-4	3	9.9	6.10
automatic-5	7	18.7	6.48
automatic-6	1834647	1939006.4	66431.69
automatic-7	128777	130716.4	1138.15
automatic-8	237711	247064	16468.9
automatic-9	352161	359339	5061.8
automatic-10	43807	46241	3477.7

Figure 1: Test results for GVNS