Heuristic Optimization Techniques
Exercise 3

Alexander Eisl (0250266), Peter Wiedermann (0025999)

November 21, 2015

1 General Variable Neighborhood search

In this programming exercise, we build on our local search from the previous exercise and implement a generalized variable neighborhood search heuristic.

1.1 Neighborhoods

We re-use the neighborhoods from assignment two and added a parameterized node move variant:

X-node-move Defined as all subsets where X vertices of the initial solution are moved to other positions.

- Size of neigborhood: $(n-1)^2 * X$
- Objective Function: Incremental, only crossings for moved edges are recalculated.

The neighborhoods from last assignement:

1-node flip This neighborhood is defined as all subsets where two vertices of the initial solution are flipped.

- Size of neigborhood: $n(n-1)/2$
- Objective Function: Incremental, crossings from flipped vertices are subtracted and recalculated.

1-edge move This neighborhood consists of all solutions where one edge is moved to a different page.

- Size of neigborhood: $(pages-1)edges$
- Objective Function: Incremental, only crossings for moved edges are recalculated.

1-node edge move This neighborhood consists of all solutions where the edges of one vertex on a specific page are moved to all different pages.

- Size of neigborhood: $(pages - 1)edges$ as worst case.[1]
- Objective Function: Incremental, only crossings for moved edges are recalculated.

1.2 Neigborhoodstructures in General VNS

As shown in the algorithm in section 1 we define two non-overlapping sets of neigborhoods:

VNS: used for the shaking, we decided to take the complex neighborhoods here, which dannot be completly searched.

1-node-move, 2-node-move, 3-node-move

VND: used for variable neighborhood descent in the inner loop:

1-edge-move, 1-node-edge-move

2 Results

We executed the code on a desktop computer with a Core i7 Quad-Core CPU with 2.67Ghz and 24 GB of main memory. Table 1 shows the number of crossings for the GVNS. We report average and standard deviations for 30 executions as well as the minim number of crossings. The run-time of our algorithm can be seen in 4

We have set a timout of 5 minutes for the calculations.

Order of the neighborhoods Tables 2 and 3 show the results when we change the order of our neighborhoods. In Table /reftab:resultsDeterminstic we report the results where we keep the order of the stochastic neighborhoods used for shaking the same and invert the order of the other neighborhoods. Table 3 shows the results where we flip the order of the neighborhoods used for shaking and keep the others the same. As expected, changing the order of our neighborhoods does influence the results of the GVNS.

[1]depending on number of edges on respective node pages

	var	avg	sd	min	min_run
automatic-1.txt	1.51	9.42	1.23	9	2
automatic-2.txt	35.90	41.68	5.99	36	0
automatic-3.txt	20.88	67.53	4.57	59	10
automatic-4.txt	234.26	97.05	15.31	84	0
automatic-5.txt	0.95	40.68	0.98	39	0
automatic-6.txt	0.00	6,153,059.00	0.00	6,153,059	0
automatic-7.txt	714,058.88	136,518.53	845.02	135,111	1
automatic-8.txt	581,888.04	541,495.47	762.82	539,317	11
automatic-9.txt	687,002.37	1,266,526.95	828.86	1,265,326	7
automatic-10.txt	24,946.19	55,289.26	157.94	55,035	16

Table 1: This table shows the results of our algorithm.

	var	avg	sd	min	min_run
automatic-1.txt	2.35	16.47	1.53	12	0
automatic-2.txt	19.15	37.89	4.38	36	0
automatic-3.txt	27.64	64.79	5.26	56	6
automatic-4.txt	116.13	137.37	10.78	108	6
automatic-5.txt	30.85	47.32	5.55	40	0
automatic-6.txt	954,027,528.83	6,011,850.26	30,887.34	5,962,740	16
automatic-7.txt	52,922.03	137,904.84	230.05	137,553	4
automatic-8.txt	5,544,075.63	504,782.95	2,354.59	501,545	12
automatic-9.txt	13,573,379.20	1,188,287.47	3,684.21	1,185,819	4
automatic-10.txt	28,478.53	55,428.00	168.76	55,178	18

Table 2: This table shows the results of our algorithm using when we flipped the order of the deterministic neighborhoods.

	var	avg	sd	min	min_run
automatic-1.txt	2.53	9.68	1.59	9	3
automatic-2.txt	41.53	39.21	6.44	25	12
automatic-3.txt	39.41	64.53	6.28	50	14
automatic-4.txt	13.62	127.53	3.69	112	7
automatic-5.txt	107.82	45.84	10.38	30	18
automatic-6.txt	0.00	6,153,059.00	0.00	6,153,059	0
automatic-7.txt	1,033,834.45	136,568.16	1,016.78	134,450	11
automatic-8.txt	0.00	541,969.00	0.00	541,969	0
automatic-9.txt	1,096,844.32	1,267,338.32	1,047.30	1,265,185	18
automatic-10.txt	76,894.93	55,855.74	277.30	55,425	5

Table 3: This table shows the results of our algorithm where we flipped the order of the stochastic neighborhoods used for shaking.

	var	avg	sd	min	min_run
automatic-1.txt	0.00	0.03	0.00	0	1
automatic-2.txt	0.00	0.02	0.01	0	10
automatic-3.txt	0.05	0.61	0.22	0	5
automatic-4.txt	0.00	0.08	0.01	0	11
automatic-5.txt	0.00	0.06	0.01	0	4
automatic-6.txt	0.26	309.22	0.51	309	13
automatic-7.txt	712.56	277.61	26.69	228	9
automatic-8.txt	0.01	302.82	0.09	303	16
automatic-9.txt	0.00	303.16	0.06	303	0
automatic-10.txt	0.00	302.35	0.04	302	7

Table 4: This table shows the runtime of our algorithm.

	var	avg	sd	min	min_run
automatic-1.txt	0.00	0.01	0.00	0	2
automatic-2.txt	0.00	0.04	0.01	0	16
automatic-3.txt	0.12	0.74	0.35	0	13
automatic-4.txt	0.00	0.06	0.02	0	7
automatic-5.txt	0.00	0.08	0.01	0	4
automatic-6.txt	0.90	309.33	0.95	309	16
automatic-7.txt	0.00	301.15	0.01	301	0
automatic-8.txt	0.00	302.81	0.02	303	6
automatic-9.txt	0.06	303.22	0.24	303	16
automatic-10.txt	0.00	302.34	0.02	302	18

Table 5: This table shows the runtime of our algorithm (det-order).

	var	avg	sd	min	min_run
automatic-1.txt	0.00	0.02	0.00	0	0
automatic-2.txt	0.00	0.02	0.01	0	13
automatic-3.txt	0.22	0.74	0.47	0	2
automatic-4.txt	0.00	0.05	0.02	0	8
automatic-5.txt	0.00	0.12	0.04	0	5
automatic-6.txt	0.01	309.07	0.11	309	12
automatic-7.txt	894.86	275.49	29.91	216	6
automatic-8.txt	0.00	302.81	0.03	303	0
automatic-9.txt	0.06	303.23	0.24	303	6
automatic-10.txt	0.00	302.34	0.01	302	6

Table 6: This table shows the runtime of our algorithm (stoch-order).