Heuristic Optimization Techniques
Exercise 2

Alexander Eisl (0250266), Peter Wiedermann (0025999)

November 8, 2015

1 Local Search

In this programming exercise, we improve the results of the construction heuristic developed in the previous exercise by using a local search. In the following subsection, we will first explain the neighborhoods that we define and then outline the step functions used.

Algorithm 1.1: Local search

```
1   input: Graph
2   output: Improved solution
3   begin
4       sol ← calculate initial solution
5       repeat:
6               choose sol' ∈ N(x)
7               if f(sol') ≤ f(sol) then
8                       sol ← sol'
9       until stopping criteria satisfied
10  end
```

1.1 Neighborhoods

We use the following neighborhoods:

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 Stepfunctions

We implemented the following thre stepfunctions:

first This step-function selects the first better solution found in the neighborhood.

best This step-function iterates over all elements of the neighborhood and chooses the best one.

random This step-function selects a random element of the neighborhood.

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 results of our local search.

We consider a local optimum if we cannot find a better solution after 10 iterations. We have set a timout of 5 minutes for the calculations.

Best neighborhood step function strategy We compare the results for three different neighboorhoods and all the implemented step-functions.

Depending on the size of the instance, either `1-node-flip` or `1-node-edge-move` yield the best results. These neighborhoods outperfom `1-edge-move` , in most cases. For smaller instances (1-5), the `1-node-flip` in combination with the `best` step-function delivers the best results. For bigger instances (6-10), the `1-node-edge-move` finds better solutions in combination with the `best` step-function. Note that for large instances, `1-node-flip` and `1-node-edge move` are stopped by our run-time restrictions before they can reach a local optimum.

A very interesting result is the performance of the `best` step-function for both the `1-node-flip` and the `1-node-edge move`. For the given instances, it seems to be important to look for the best improvement, `first` and `random` tend to get stuck in worse local optima.

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

Initial Solution The choice of the initial solution is crucial to achieve results in reasonable calculation time. Using random initialization might degenerate into random search or end in a bad local optimum with less improvement.

When we compare the results to a random initial solution, we could measure an average increase of 55% crossings.[2]

Neighborhoods Incremental objective functions implemented as described in Section 1.1 are performing equally well for all three neighborhoods.

Subsequent (possibly random) moves in our neighborhoods can only reach valid solutions in the search space because every solution in our neighborhoods is valid.

Reaching local optimum If a local optimum is reached it is found in less than 10 iterations in most cases. As can be seen in Table 1, there are only a few exceptions. The results with iterations$=-1$ did not reach a local optimum and ran into our time-out condition. For one instance, we do not find an improvement. So with respect to our neighborhood, we seem to be in a local optimum.

[2]measured with 1-node flip and `best` stepfunction

	1-Node (first)	1-Node (best)	1-Node (random)	1-Edge (first)	1-Edge (best)	1-Edge (random)	1-N-E (first)	1-N-E (best)	1-N-E (random)
automatic-1.txt	***21*** 0 / 0.00	***21*** 0 / 0.01	***21*** 0 / 0.00	***21*** 0 / 0.01	***21*** 0 / 0.01	***21*** 0 / 0.00	***21*** 0 / 0.00	***21*** 0 / 0.01	***21*** 0 / 0.00
automatic-2.txt	48 0 / 0.00	***30*** 4 / 0.05	48 0 / 0.00	48 0 / 0.00	48 0 / 0.01	48 0 / 0.00	48 0 / 0.00	48 0 / 0.01	48 0 / 0.00
automatic-3.txt	104 0 / 0.02	94 5 / 0.22	104 0 / 0.00	104 0 / 0.00	***79*** 8 / 0.10	104 0 / 0.02	104 0 / 0.00	97 2 / 0.06	104 0 / 0.01
automatic-4.txt	69 36 / 0.16	***12*** 17 / 0.45	192 0 / 0.00	163 6 / 0.00	155 9 / 0.02	192 0 / 0.00	164 9 / 0.02	169 4 / 0.02	188 0 / 0.01
automatic-5.txt	75 0 / 0.00	***32*** 10 / 0.32	76 0 / 0.00	77 0 / 0.00	41 15 / 0.04	77 0 / 0.00	77 0 / 0.00	69 1 / 0.02	77 0 / 0.00
automatic-6.txt	6,153,059 0 / 224.99	6,151,784 -1 / 308.61	6,153,059 0 / 8.61	6,033,148 -1 / 308.59	6,151,720 -1 / 308.78	6,151,901 -1 / 343.80	6,153,059 0 / 7.94	***5,965,084*** -1 / 308.62	6,153,059 0 / 10.86
automatic-7.txt	142,777 32 / 3.27	140,291 -1 / 301.14	143,570 0 / 0.15	140,027 60 / 11.99	***137,629*** 315 / 204.66	143,793 11 / 2.91	143,905 0 / 0.18	142,786 19 / 18.80	143,905 0 / 1.63
automatic-8.txt	541,935 3 / 39.74	537,728 -1 / 302.71	541,969 0 / 1.81	521,950 -1 / 302.71	539,307 -1 / 303.26	541,969 0 / 54.97	541,969 0 / 1.83	***498,763*** -1 / 302.76	541,969 0 / 7.15
automatic-9.txt	1,268,407 4 / 2.85	1,232,469 -1 / 303.03	1,268,456 0 / 2.18	1,231,026 -1 / 303.04	1,265,579 -1 / 303.03	1,265,936 50 / 286.19	1,267,963 0 / 2.19	***1,186,295*** -1 / 303.04	1,268,005 16 / 10.64
automatic-10.txt	56,021 70 / 44.04	56,115 -1 / 302.29	56,260 0 / 1.47	***53,952*** -1 / 302.27	54,848 -1 / 302.27	56,260 0 / 7.72	56,260 0 / 1.34	55,235 -1 / 302.33	56,260 0 / 5.21

Table 1: Local search results. For each instance, we show the number of crossings (first row), the iteration needed to reach the local optimum (second row, first value) and the run-time in seconds of our algorithm (second row, second value). If no local optimum was found the number of iterations is -1. The best solution is highlighted in bold.