Technische Universität Berlin
Institut für Mathematik

Wintersemester 11/12

Computational Integer Programming

PD Dr. Ralf Borndörfer
Dr. Thorsten Koch

Exercise sheet 12

Deadline: Thu, 26 Jan. 2012, by email to `reuther@zib.de`

Exercise 1. **(Tutorial session)**

Improve the tour in Fig. 1 using the Lin-Kernighan heuristic.

Exercise 2. **(Tutorial session)**

Consider the sequential k-opt move of the Lin-Kernighan heuristic. Prove:

a) Every improving 2-opt move is sequential.
b) Every improving 3-opt move is sequential.
c) The double-bridge move depicted in Fig. 2 is a non-sequential 4-opt move.

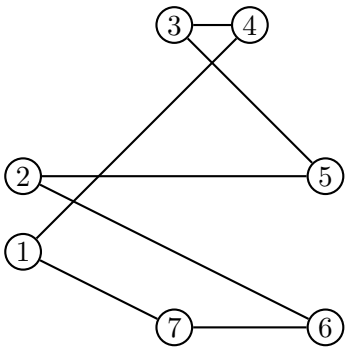

Figure 1: A tour in the Euclidean plane.

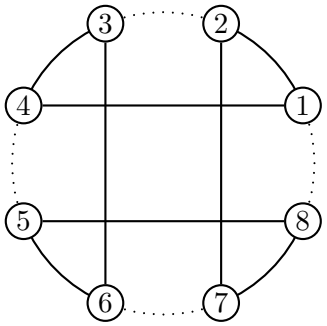

Figure 2: A double-bridge move.

Exercise 3. **(Tutorial session)**

Consider a graph $G = (V, E)$ with $|V| = n$ nodes, edge weights c_e, and a special node $1 \in V$. Let T be a cost minimal 1-tree (i.e., $T \setminus \delta(1)$ is a spanning tree in $G \setminus \{1\}$ and $T \cap \delta(1)$ consists of the two cheapest edges in $\delta(1)$, breaking ties arbitrarily). Let $T(e)$ be a minimum cost 1-tree containing edge $e \in E$ and define the α-tolerance of each edge as

$$\alpha(e) := c\big(T(e)\big) - c(T).$$

Prove:

a) $\alpha(e) = 0$ for every $e \in T$.
b) $\alpha(e) \geq 0$ for every $e \in E$.
c) $\alpha : E \to \mathbb{R}$ can be computed in $O(n^3)$ time.

Exercise 4. **10 points**

Implement the Lin-Kernighan heuristic for the euclidean TSP by using a template Java program provided in the tarball `jLK.tgz` on the lecture's web page. Download this archive, copy it into your virtual machine and unpack it with the command `tar xzf jLK.tgz`. A detailed description of the contents of `jLK` is provided in the file `README`. A quick start reads as follows:

a) Build the program by typing `ant` in the `jLK/` directory. The jar archive jLK.jar should be created.
b) Run the program by typing `java -jar jLK.jar [problem-file]`.
c) You can use every file from the `instances/` directory as problem-file.

The template program will do the following steps:

a) Parse the TSP instance.
b) Compute the delaunay neighborhood.
c) Compute a initial tour by a very simple construction heuristic.
d) Run the Lin-Kernighan template implementation *without doing any improvement*.
e) Visualize every step of computation.

Your task is to complete the implementation of the member function `step()` to make the algorithm working. This function is located in the file `lk/LinKernighan.java` approx. line 167. It is the heart of the algorithm and finds improving k-OPT moves by a recursive depth first search. The whole implementation of the program is derived form the paper [1]. It provides a very clear, generic, and compact description of an implementation of the Lin-Kernighan heuristic. So, you do not have to complete the function body by yourself. Moreover you do not have to implement any data structure and you will find some hints at the affected code locations (search for `write code here`).

Your "only" have to understand what should be going on in this function and to translate the pseudo code on page 7 of [1] into Java code. A possible solution of this exercise has approx. 70 lines of code.

References

[1] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. Finding tours in the TSP, 1999. `www.tsp.gatech.edu/methods/papers/lk_report.ps`.