Technische Universität Berlin
Institut für Mathematik

Dr. Rüdiger Stephan

Mathematical Tools for Engineering and Management
Winter Term 2011/2012

December 07, 2011

Exercise sheet 7

The due date of the graded homework (Exercise 2) is January 4, 2012. Please send your AIMMS project file by email to `stephan@math.tu-berlin.de`.

In this session we implement some starting heuristics for the symmetric TSP. Let $K_n = (V, E)$ be a complete graph on $n \geq 3$ nodes with edge length $c_{ij} \geq 0$ for all $ij \in E$.

(a) **Nearest Neighbour (NN)**
1. Choose any node $i \in V$, set $W := V \setminus \{i\}$, $T := \emptyset$, and $p := i$.
2. If $W = \emptyset$, set $T := T \cup \{ip\}$ and STOP (T is a tour).
3. Determine a node $j \in W$ such that

$$c_{pj} = \min\{c_{pk} | k \in W\}.$$

4. Set $T := T \cup \{pj\}$, $W := W \setminus \{j\}$, $p := j$, and go back to 2.

(b) **Nearest Insert (NI)**
1. Choose any cycle on three nodes.
2. If $V \setminus V(C) = \emptyset$, STOP, ($C$ is a tour).
3. Determine a node $p \in V \setminus V(C)$ such that there exists a node $q \in V(C)$ with

$$c_{pq} = \min\{\min\{c_{ij} | j \in V(C)\} | i \in V \setminus V(C)\}.$$

4. Determine an edge $uv \in C$ with

$$c_{up} + c_{pv} - c_{uv} = \min\{c_{ip} + c_{pj} - c_{ij} | ij \in C\}.$$

5. Set $C := (C \setminus \{uv\}) \cup \{up, pv\}$ and go back to 2.

(c) **Farthest Insert (FI)**
Like NI, just replace the third step by:
3. Determine a node $p \in V \setminus V(C)$ such that there exists a node $q \in V(C)$ with

$$c_{pq} = \max\{\min\{c_{ij} | j \in V(C)\} | i \in V \setminus V(C)\}.$$

(d) **Cheapest Insert (CI)**
Like NI, just replace the third and the fourth step by:
3. Determine a node $p \in V \setminus V(C)$ and an edge $uv \in C$ with

$$c_{up} + c_{pv} - c_{uv} = \min\{c_{ik} + c_{kj} - c_{ij} | ij \in C, k \in V \setminus C\}.$$

(e) **Doubly Minimum Spanning Tree (DMST)**
1. Determine a minimum spaning tree $G = (V, F)$ of K_n.
2. Obtain a directed graph $D = (V, A)$ from G by replacing each edge $ij \in F$ by the arcs (i, j) and (j, i), and determine an Eulerian Tour $T = (v_1, v_2, \ldots, v_{2n-1})$ in D, that is, a closed directed walk through all nodes ($v_1 = v_{2n-1}$, $(v_i, v_{i+1}) \in A$ for $i = 1, 2, \ldots, 2n-2$).
3. Determine a (traveling salesman) tour from T by short cutting, that is, delete a node v_j ($1 < j < 2n-1$) from T whenever $v_j = v_i$ for some $1 \leq i < j$.

Exercise 1

In this exercise we adapt the example application "Traveling Salesman" from the AIMMS webpage. This application provides start and improvement heuristics for the Euklidean TSP. In the Euklidean TSP each node i comes with a location (x_i, y_i) in the Euclidean plane. The edge length c_{ij} for two nodes $i, j \in V$ is defined to be the Euclidean distance between i and j, that is,

$$c_{ij} := \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

The NN heuristic is already impemented. So in this exercise we focus on DMST.

(i) **Preparation**
Download from the AIMMS webpage

`http://www.aimms.com/downloads/application-examples/traveling-salesman`

the `.aimmspack`-file, open the AIMMs project file `traveling.prj`, and make a copy of the page `City Tour` and call the copy `Starting Heuristics`.

Rename the following procedures:

`NewTour` ↬ `NewGraph`, `InitialTour` ↬ `NearestNeighbour`

Go to the body of `NewGraph` and set the value of both `MaxNeighbors` and `IterationLimit` to `MaxCities-1`.

(ii) **Implementation of Kruskal's Algorithm**
Declare a procedure called `Kruskal` and create a declaration node as subnode of the procedure in the model tree.

First, we want to sort the edges of the graph such that they are in nondecreasing order. For this, define the following set as subnode of `Create New and Initial Tour` → `Declaration`:

```
Routes      := {(i,j)|i<j}
Subset of   (Cities,Cities)
Index       e
Tags        (tail, head)
Order by    Distance(e)
```

Next, we need an element parameter `Kruskal` → `Declaration` → `ReferenceCity` with index domain `c` and range `Cities`, and we need a binary parameter `Create New and Initial Tour` → `Declaration` → `TreeRoute` with index domain `(i,j)` in order to indicate if `(i,j)` is in the tree to be created. To visualize your following implementation of Kruskal's algorithm go to the page `Starting Heuristics`, switch to the edit mode, and redefine one of the buttons used for starting one of the heuristics and the associated network object. (The action to be performed by pressing the button should be `Kruskal`, and the arcs to be visualized from the network should be `TreeRoute`. In order to display the city numbers, click on `Text` in the properties window of the network object and select "Element name". If you want to avoid deformation of X- and Y-space, click on `Network` and set the check mark for equal X and Y scale.)

Finally, go back to the model tree and implement Kruskal's algorithm in the body of `Kruskal`. Here, first make sure that `TreeRoute` and `ReferenceCity` are empty.

(iii) **Finding an Eulerian Tour in the MST**
This is already implemented. Add the procedure `DMST` to the model tree, go to the homepage of this course, open the file `dmst.txt`, and paste its content to the procedure.

(iv) **Creating a tour from the Eulerian tour**
Add the short cutting procedure to the end of the body of `DMST`, and visualize the tour.

Graded Homework

Exercise 2

Implement NI, FI, and CI in AIMMS, incorporate your heuristics into the Traveling salesman application, and display the generated tours and their length on the user page. If you do not have enough space, remove the network objects for Kruskal and DMST or create a new user page within the Demo page.