European Social Fund Prague & EU: We invests in your future.

Euclidean Travelling Salesman Problem

2-opt heuristic

Libor Bukata (bukatlib@fel.cvut.cz)

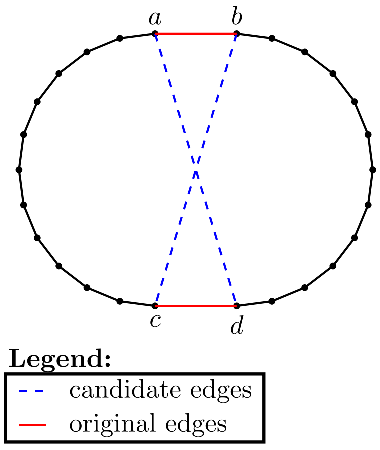
April 4, 2013

1 2-opt heuristic

The 2-opt heuristic which is often used in *Euclidean Travelling Salesman Problem* is based on local search technique. The heuristic starts with an initial tour (a closed path generated randomly or by some heuristic) and tries to improve it using local modifications until the local minimum is found. It can be proved that the final tour will never be worse than $4\sqrt{n}$ times the optimum where n is the number of locations [1]. Fortunately, in practise the obtained tour is usually much better than the worst case scenario. The pseudo-code of the 2-opt heuristic is in Algorithm 1 and the step of the heuristic (lines 3-5 in Algorithm 1) is depicted in Figure 1.

Algorithm 1 The 2-opt heuristic [1].

```
Require: $K_n$ – A complete graph, i.e. an instance.
Require: $T$ – It is an initial tour.
Ensure: The 2-opt optimal tour.
1: Let $\mathcal{S}$ is a set of 2-element subsets of $E(T)$.
2: % For example $\mathcal{S} = \{\{e_1, e_2\}, \{e_1, e_3\}, \{e_2, e_3\}\}$ is generated from $E(T) = \{e_1, e_2, e_3\}$.
3: for $\forall s \in \mathcal{S}$ and $\forall$ tours $T'$ with $E(T') \supseteq E(T) \setminus s$ do
4:     if $cost(E(T')) < cost(E(T))$ then
5:         $T := T'$
6:         Go to line 1.
7:     end if
8: end for
9: return $T$
```



A step of the 2-opt heuristic

```
Require: $T$ – A Hamiltonian Circuit.
Require: $W$ – The matrix of weights $w_{i,j}$.

if $w_{a,d} + w_{b,c} < w_{a,b} + w_{c,d}$ then
    $T = T \setminus \{(a,b),(c,d)\} \cup \{(a,d),(b,c)\}$
else
    No improvement...
end if
```

Figure 1: Illustration of the 2-opt heuristic.

2 Required matrices

Input locations are stored as matrix $L \in \mathbb{R}^{2\times n}$ such that each matrix's column corresponds with coordinate $(x_i, y_i)^T$ of location i.

$$L = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ y_1 & y_2 & \cdots & y_n \end{pmatrix} \tag{1}$$

From the set of locations *complete graph* K_n is created. This graph is described using the matrix of weights W where each weight $w_{i,j}$ corresponds to the distance between locations i and j. It should be obvious that the matrix is symmetric since it is reasonable to assume that $w_{i,j} = w_{j,i}$.

$$W = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,n} \end{pmatrix} \tag{2}$$

A tour is stored in matrix T where at each column a weighted edge e_j is stored. Weight w_j of edge e_j is the distance between locations stored at the $p_{1,j}$-th and $p_{2,j}$-th columns of matrix L.

$$T = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ w_1 & w_2 & \cdots & w_n \end{pmatrix} \tag{3}$$

After that total tour distance is calculated by the following way.

$$cost(T) = \sum_{\forall e_j \in T} w_j \tag{4}$$

3 A seminar assignment

Apply 2-opt heuristic to improve the initial tour in Figure 2 and calculate the final route distance. The problem is stated in the equations (5), (6) and (7).

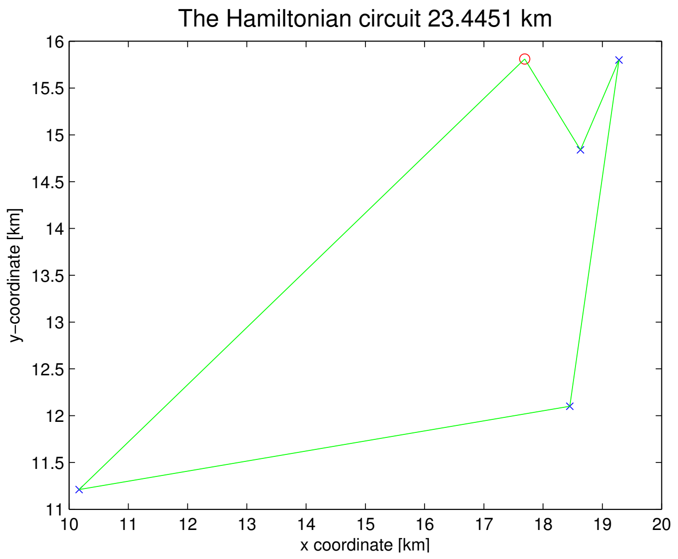


Figure 2: An initial tour found by the Double-Tree algorithm.

$$L = \begin{pmatrix} 17.69 & 19.28 & 10.17 & 18.63 & 18.45 \\ 15.81 & 15.8 & 11.21 & 14.84 & 12.1 \end{pmatrix} \tag{5}$$

$$W = \begin{pmatrix} 0 & 1.59 & 8.82 & 1.35 & 3.787 \\ 1.59 & 0 & 10.2 & 1.16 & 3.79 \\ 8.82 & 10.2 & 0 & 9.21 & 8.33 \\ 1.35 & 1.16 & 9.21 & 0 & 2.75 \\ 3.787 & 3.79 & 8.33 & 2.75 & 0 \end{pmatrix} \tag{6}$$

$$T = \begin{pmatrix} 1 & 4 & 2 & 5 & 3 \\ 4 & 2 & 5 & 3 & 1 \\ 1.35 & 1.16 & 3.79 & 8.33 & 8.82 \end{pmatrix} \tag{7}$$

4 A homework assignment

To get more details about the *Travelling Salesman Problem* and the Double-Tree algorithm, which is used to generate an initial tour, take a look at Korte at al. [1]. The main task is to use available skeleton script, which can be downloaded from the subject's homepage, to implement 2-opt heuristic. To be able to do so a few functions have to be described.

1. `function tsp_skeleton(varargin)`[1]
   - It is entry point to the Travelling Salesman Problem.
   - If no optional parameter L (1) is given locations are generated randomly.
2. `function apply2optHeuristicsDCV(circuit, locations)`
   - At this function you should implement 2-opt heuristic.
   - The matrix of weights W (2) is precomputed at the beginning.
   - *circuit* $\sim T$ (3); *locations* $\sim L$ (1)
3. `function drawKoptImprovement(circuit, locations, oldEdges, newEdges)`
   - It displays newly added and removed edges to the Hamiltonian circuit.
   - *circuit* $\sim T$ (3); *locations* $\sim L$ (1)
   - Parameters *oldEdges* and newEdges have the same format as T (3) with the difference that the number of columns is smaller.
   - The distance of the tour is printed in the title of the figure.

A homework assignment: Using the skeleton script implement the 2-opt heuristic. Intermediate results should be visualised by the *drawKoptImprovement* function.

Hints: Every time you modify the Hamiltonian circuit you should check if the newly created tour is still a circuit (i.e. one graph component). Each pair of candidate edges should have four distinct nodes, therefore two edges interconnected with a node are improper.

References

[1] B. H. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*. Springer, third ed., 2006.

[2] J. Demel, *Grafy a jejich aplikace*. Academia, second ed., 2002.

[3] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall; United States Ed edition, 1993.

[1] You can rename this function and m-file if you want.