Trains

Denis Denysyev

June 2022

1 Task

The schedule of train crossings between several stations is given in the format "train number; departure station; arrival station; cost; departure time; arrival time" (it is guaranteed that there are no crossings for more than a day) . It is necessary to get the "best" options (several, if possible) of travel between all stations so that you visit each station 1 time. Requests for the best options:

- Best by price
- Best by time

As we can see we have to different tasks, so we going to solve them separately.

2 Best by price path

We consider that the schedule repeats after some time. Then we will not use any information about the time in this task.

2.1 Formal task

We have directed graph. Elements of vertices set V are all stations and elements of edges set E are *the cheapest voyage* between two stations where weight of the edge is *price* of voyage. Our task is to find Hamiltonian path which has the smallest sum of edges weights.

Hamiltonian path is a path in an undirected or directed graph that visits each vertex exactly once.

2.2 Greedy algorithm

For our specific data set this algorithm gives the cheapest path with complexity $O(n)$. Unfortunately it will not work on all graphs and will give only approximate results. Lets take a look at steps of this algorithm:

1. Create graph adjacency matrix with minimal weights of edges between each two vertices: $\begin{bmatrix} w_{11} & w_{12} & w_{13} & \dots & w_{1n} \\ w_{21} & w_{22} & w_{23} & \dots & w_{2n} \\ \dots & \dots & \dots & \dots & \dots \\ w_{d1} & w_{n2} & w_{n3} & \dots & w_{nn} \end{bmatrix}$.

2. Create boolean array to *used* = {*False,False,...False*}
3. Take *i-th* station and mark it as used.
4. Try to build Halminthon path in each iteration pick the edge with smallest price. Then move to its end vertex and mark it as used. Keep doing it until all vertices are used or you can not pick a new vertex.
5. If we found Halminthon path and it has less weights than the previous path, update minimum path
6. Repeat steps *2,3,4,5* for all n stations

2.3 Brute force algorithm

For our data set we can use brute force algorithm with complexity $O(n!)$,because we have only *6 vertices*:

1. Create graph adjacency matrix with minimal weights of edges between each two vertices: $\begin{bmatrix} x_{11} & w_{12} & w_{13} & \dots & w_{1n} \\ w_{21} & w_{22} & w_{23} & \dots & w_{2n} \\ \multicolumn{5}{c}{\dotfill} \\ w_{d1} & w_{n2} & w_{n3} & \dots & w_{nn} \end{bmatrix}$.
2. Create boolean array to *used* = {*False,False,...False*}
3. Take *i-th* station and mark it as used.
4. Create all permutation of the Halminthon path with the start station i. Then pick the path with the smallest weight.
5. If we found Halminthon path and it has less weight than the previous path, update minimum path
6. Repeat steps *2,3,4,5* for all n stations

2.4 Results

Price of the travel: 1927.18
Trains is used: [1171, 1140, 919, 1156, 1148]
Order of visiting stations [1909, 1921, 1902, 1981, 1937, 1929]
Costs of each trip: [1407.37, 114.12, 130.6, 140.98, 134.11]
Greedy algorithm time: 0.00046
Brute force algorithm time: 0.00179

3 Best by time path

We consider our trip starts from first train trip, so let's compose the task.

3.1 Formal task

We have directed multi graph. Elements of vertices set V are all stations and elements of edges set E are *all voyages* between two stations where weight of the edge is *travel time + waiting time* of voyage. Our task is to find Hamiltonian path which has the smallest sum of edges weights.

3.2 Greedy algorithm + brute force

We can get approximate results, but with complexity only $O(n^2)$. Lets take a look at steps of this algorithm:

1. Create graph adjacency matrix with without weights of edges between each two vertices: $\begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{d1} & x_{n2} & x_{n3} & \cdots & x_{nn} \end{bmatrix}$.
2. Create boolean array to *used* = {*False,False,...False*}
3. Take *i-th* station and mark it as used.
4. Try to build Halminthon path. For each stations which is connected to our current vertex pick the edge with smallest weight between them. Then move to its end vertex and mark it as used. Keep doing it until all vertices are used or you can not pick a new vertex.
5. If we found Halminthon path and it has less weight than the previous path, update minimum path
6. Repeat steps *2,3,4,5* for all n stations

3.3 Full brute force

In this algorithm we simply look for all Halminthon path permutations and choose the fastest one. Unfortunately it is too slow, but gives exact answer.

3.4 Results

Greedy algorithm + brute force:
Price of the time: 54:10:0
Trains is used: [1167, 1811, 1342, 1365, 1413]
Order of visiting stations: [1981, 1902, 1921, 1909, 1929, 1937]
Time: 0.950183629989624

Full brute force:
Best time: 52:31:0
Price of the time: 52:31:0
Trains is used: [1156, 1098, 1382, 1035, 1342]
Order of visiting stations [1981, 1937, 1902, 1929, 1921, 1909]
Time: 398.7164878845215
Error value: 3.14%