IMPERIAL COLLEGE LONDON, DEPARTMENT OF MATHEMATICS

Scientific Computing (M3SC)

Peter J. Schmid

February 9, 2017

1 CLASS PROJECT 1: TRAFFIC FLOW THROUGH A SIMPLIFIED STREET NETWORK

We will use the street network of Rome, used in one of the class tutorials, to simulate the traffic flow through the city during rush hour where each car is making choices according to the local optimal route as determined by its GPS-navigation system. (Download the modified RomeEdges-file.)
We model the traffic flow as a time-discrete process (minute-per-minute; each iteration represents one minute) where we store and update the number of cars at each of the 58 nodes in the city map. We inject 20 cars per minute at node 13 in the figure (St. Peter's Square; `python` index 12) and move the cars according to the following rules:

1. cars at each node choose the currently optimal (fastest) route to node 52 in the figure (Coliseum; `python` index 51),

2. during each minute (time step), only 70 % of the cars move to the next node, the remaining 30 % of the cars stay behind at the original node,

3. only integer number of cars move through the network (so the 70 % - 30 % - split can only be enforced approximatively), (caution: be particularly careful with rounding the number of cars to the nearest integer to not lose any cars as they move through the network; the number of cars has to be conserved)

4. at node 52 (Coliseum; `python` index 51), cars leave the network, but this time with 60 % of cars left behind (due to slow traffic),

5. after all cars have moved to their new position, the weights of all segments w_{ij} in the network, connecting node i to node j, are updated according to the rule

$$w_{ij} = w_{ij}^{(0)} + \xi \frac{c_i + c_j}{2} \tag{1.1}$$

where $w_{ij}^{(0)}$ denotes the original weights of the edges, as determined from `RomeEdges`, and c_i, c_j stand for the number of cars at nodes i and j, respectively. The parameter ξ is set to $\xi = 0.01$.

We simulate the traffic flow through the city for 200 iterations (200 minutes), where cars at the rate of 20 per minute are injected (at node 13) only for the first 180 iterations (minutes); afterwards, no more cars enter the network, even though the cars already in the network keep on moving.

- Determine for each node the maximum load (maximum number of cars) over the 200 iterations.
- Which are the five most congested nodes?
- Which edges are not utilized at all? Why?
- What flow pattern do we observed for parameter $\xi = 0$?
- An accident occurs at node 30 (`python`-index 29) which blocks any route to or from node 30. Which nodes are now the most congested and what is their maximum load? Which nodes (besides node 30) decrease the most in peak value, which nodes increase the most in peak value?

2 WHAT TO SUBMIT

Produce a brief write-up of your work (using the supplied LaTeX-template) that outlines your solution strategy and details about your `python` implementation. Add generous comments to your code to explain critical concepts and steps. Make a zip-file (or tar-file) of your write-up (pdf-file only) and your `python`-code and upload it to the blackboard course web-page **at the latest by 17:00 on 23. February 2017**. Late submissions will not be accepted without prior permission.

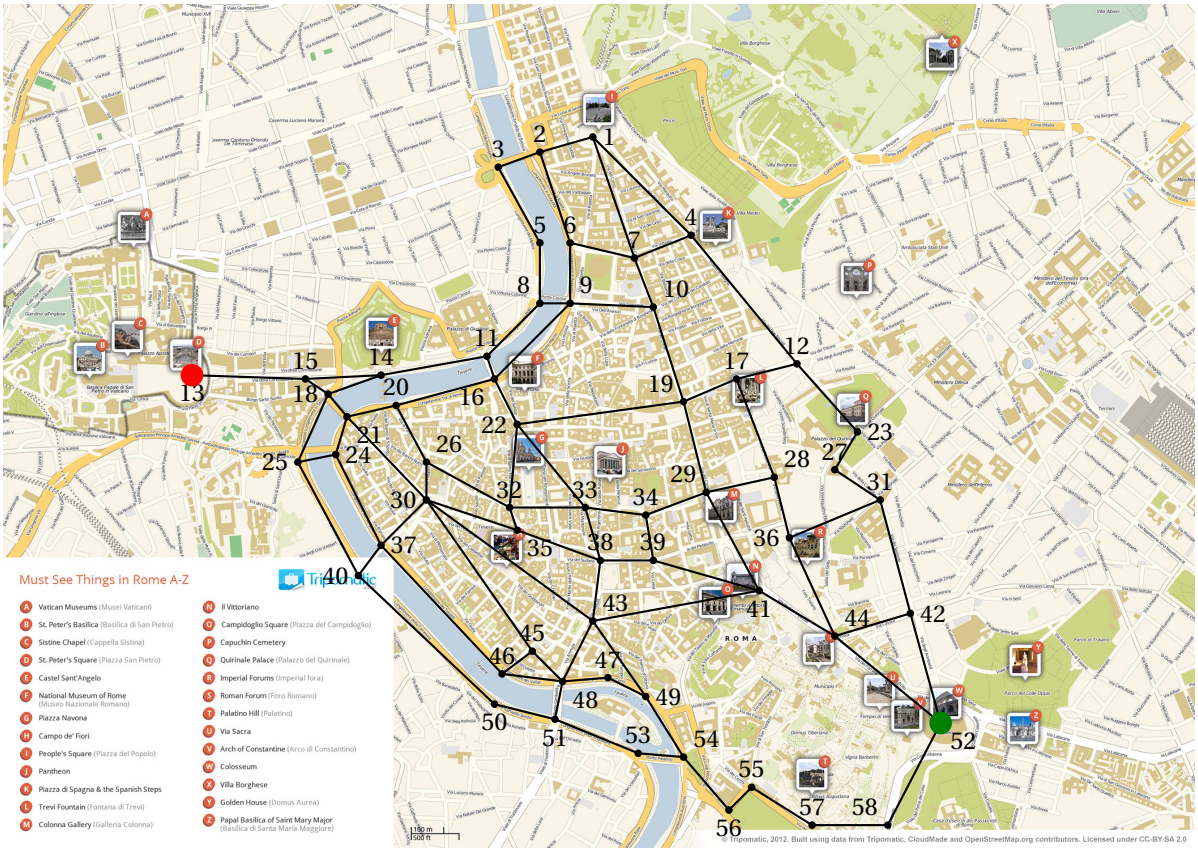

Figure 2.1: Map of Rome with a simplified traffic network consisting of 58 vertices and 156 edges (counting bi-directional edges double).