Master ICT - USTH
BC2.04b - Data Visualization – Week2
2024-2025

Visualisation of shortest paths and communities in a network graph

K. Bertet

<u>Objective :</u> The objective of this practical work is to visualize shortest paths obtained with Dijkstra algorithm, and communities obtained with the Louvain detection method.

1) The graph of the Parisian metro

The file *metro.graph* contains data describing the graph of the Parisian subway :

- Each node corresponds to a station for a given line (for example, République [line 3] and République [line 5] are two different nodes). Each node is associated with the name of the station (character string) and the position of the station on a map (scale: 1~25.7m).
- An oriented and attributed edge connects two nodes if the metro directly connects the corresponding stations. The graph is not symmetrical because of some « one-way » lines, for example at the Porte d'Auteuil station. The edges are attributed by the estimated travel time in seconds (assuming an average speed of 10m/s, or 36km/h).
- Two symmetrical edges connect two nodes if it is possible to walk without changing tickets between the corresponding stations. These nodes are then attributes by an estimate of the average travel and waiting time (120s).

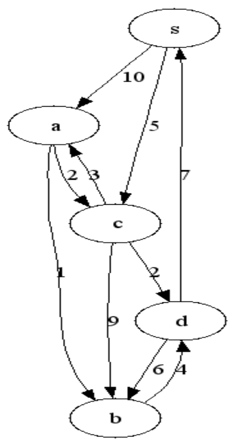
Exercice 1. Create the graph of the Paris subway, with :

- Nodes attributes : position of the stations, name (String)
- Edges attributes : travel time

Improve the visualization with :

- a coloration of the edges according to the line of the Parisian metro
- a size of the nodes according to the degree of the stations, according to a given number of sizes.
- A coloration of station using the graph coloring algorithm :
    - https://networkx.org/documentation/stable/reference/algorithms/coloring.html#
    - https://en.wikipedia.org/wiki/Graph_coloring

Exercice 2. Create the following graph, and compute its shortest from using the *shortest_paths* method implemented in the nextworkx packages : https://networkx.org/documentation/stable/reference/algorithms/shortest_paths.html



Exercice 3. Improve the visualization of the Parisian metro so that shortest path between to given stations is colored.

2) Communities detection

```
!pip install partition-networkx
!pip install python-louvain
!pip install matplotlib

import partition_networkx
from community import community_louvain

%matplotlib inline
import matplotlib.pyplot as plt
import networkx as nx
import networkx.algorithms.community as nx_comm
import numpy as np
```

Exercice 4. We will use the *planted_partition_graph(l,k,p_in,p_out)* function, which generates a random graph with l groups, each composed of k vertices. Two vertices in the same group are connected with a probability of p_in, and two vertices from different groups are connected with a probability of p_out.
The code below generates a graph with 40 vertices divided into 4 groups with probabilities of 0.7 and 0.03. The chosen probabilities should make the 4 groups visible.

```
l = 4 # number of groups
k = 10 # number of nodes per group
p_in = 0.7 # intra-group probability of connection
p_out = 0.03 # between-group probability of connection
G = nx.generators.planted_partition_graph(l, k, p_in, p_out)
nx.draw_networkx(G)
```

Exercice 5. In this example, we can consider these 4 groups as ground truth of communities in the graph. The following code allows extracting these 4 groups. Add the instructions to color them with 4 distinct colors. Explain the link between the two probabilities and the community structure of the graph.

```
l = 4 # number of groups
k = 10 # number of nodes per group
gt_part = [set(range(k*i, k*(i+1))) for i in range(l)]
print(gt_part)
```

Exercice 6. Implement a function my_modularty (G) for the calculation of modularity of a graph. Then give the modularity of the previous graph, and verify the correctness of your implementation by comparing its result to the modularity function implemented in Networkx. You must obtain 0.6175491898148149
https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.quality.modularity.html

Exercice 7. Write code to plot a curve that will calculate the modularity of this ground truth as a function of p_out, using 4 groups of 10 vertices with an intra-group connection probability of 0.8.

Exercice 8. Write a function that detects communities using the Louvain method and that color then. Compare with the groubd truth for the graph with 40 vertices divided into 4 groups with probabilities of 0.7 and 0.03.
https://python-louvain.readthedocs.io/en/latest/api.html?highlight=best_partition#community.best_partition

Exercice 9. For a better comparison of the modularity of Louvain communities to that of the ground truth, plot both modularity curves for graphs composed of 4 groups of 32 vertices each and an internal probability (p_in) of 0.7. Only the external probability (p_out) will vary.