Apendix for STCP network study report

Diogo Bogas

12 December 2016

1 Data Structures

In this particular section, all the data structures used while crunching the data together will be described by chronological order. the first piece of information retrieved from the STCP website, was about the lines, so the structure BusLine follows:

- sentido
- accessibility
- code
- description
- pubcode
- LineStops

Sentido is an integer that can take the values 0 or 1 and represents the direction a bus goes through a line. For example, the 207 line can go from Campanh to Mercado da Foz, and from Mercado da Foz to Campanh.
Accessibility is an integer that dictates the what special access conditions a bus line has. It can have wheel chair access, baby stroler access for example. Its a variable that was born when the information was beeing gathered, but ended up remaining in the structure.
Code is the server way to tell bus lines from each other.
Description is a String that tells us what appears in front of a bus that if going through a line.

PubCode is the people's way to tell lines from one another.
And lineStops is a List of Strings that represents the Stops that compose a line.

After knowing each line's composing Stops, a list of all the stops was constructed, and the info for each stop was stored locally. The JAVA structure to represent a Stop is:

1. stopCode, the ID of a Stop.
2. address.
3. zone, the part of the city a Stop belongs to.
4. name, the way people know which stop is which.
5. longitude.
6. latitude.
7. totalLinesServed, a number.
8. linesServed, a list of the lines served.
9. adjacentStops, the stops it can connect to.
10. distanceBFS, an auxiliary variable to a BFS method.

The last four variables are derived, and are used as auxiliary variables to some methods. In a graph, a Stop is a node. The edges are the interactions between nodes:

1. source;
2. target;
3. desc;
4. weight;

Source and Target define the edge, as the network is a directed graph. Desc is a String composed by both stopCodes, source + - + target. The weight variable represents the number of lines that use that edge.

When grouping the information by street, the nodes of the graph(streets themselves) were represented by the BusStreet structure:

- street
- stops, a list of all the stops in that street
- neighbours
- longitude;
- latitude;

The neighbours attribute was supposed to represent the streets adjacent to a particular street, but ended up not beeing used. The edges in this type of grouping are reresented by AdressEdge:

- src
- dest
- weight
- nome

The graph is directed still, weight is the number of lines that use that particular interaction and nome is a way to diferentiate the edges, and is derived from the source and target streets.

The final way to group the stops is by code, so the Spot edge was created:

- code
- stops
- LinesServed

Code is the way to tell Spots apart, and it's derivarion is described in the main document.
Stops is a list of stops the Spot has in itself.
LinesServed is a list with all the lines that go through that Spot.

The edges in this type of grouping are represented by the SpotEdge structure:

- from
- to
- weight
- name

From and to are the source and target Spots, weight and name, as the other types of edges are the number of lines that use the edge and a way to distinguish the edges respectively.

2 Gephi source files

The first file to be described is called stops.csv , and it represents every single stop there is in this network. Five columns per stop :

- Id , is the code of each stop (the unique attribute).
- Address
- Zone

- name
- latitude
- longitude

To make this file, the method makeNodesCSV was created. It begins by printing "Id;Address;Zone;Name;Longitude;Latitude" in the first line of the output file. Then, for each stop it reads from the allStops.txt file, prints its attributes by the order sugested in the first line. As in the source file, each stop only takes one line.

The second file to talk about is called allEdges.csv, with four columns per edge:

- Source stop
- Target stop
- type which is going to be directed for every edge in this study
- weight

The generating method to this file is allEdgesCSV, and is similar to the nodes one. It also begins by printing what each column represents, (in this case "Source;Target;Type;Weight") and then for each edge it receives, prints its attributes in the order required. The source of the edges is an HashMap that comes from the getAllEdges method.
Because the stops were also grouped by address, we need two more files for that, the node file for that grouping is called streetNodes, and the edges file is called streetEdges. The first is generated by the makeStreetNodesCSV method, which in the first line of the file writes "Id;TotalStops;Longitude;Latitude", and in each subsequent line writes each streetNode's values.
The second file for this type of grouping is generated by makeStreetEdgesCSV, and this file holds, for each edge, the weight, source, target and type.The weight in this particular case represents how many bus lines use that particular edge.
The code grouping has a very similar file generation and type of information stored. SpotNodes and SpotEdges are the nodes and edges files, generated by makeSpotNodeCSV and makeSpotEdgesCSV respectively. The nodes files holds the Id and the number of total stops grouped by that code, while the

edges file, for each edge, knows the source, the target, the weight and the type. Now that all the important files and methods have been exposed, we can rely on the previously mentioned Gephi to study the network.

This ends this appendix.