Analisys of the STCP transport network

Diogo Bogas

DCC/FCUP

December 12, 2016

Contents

1 Abstract

This theme aims to do an initial analisys of the complex network that represents all the bus lines belonging to STCP. The student will start by creating the network, with automatic data extraction from sources like the website stcp.pt. Then, with all the gathered data, there should be a preliminary data analisys with focus on node centrality, degree distribution and charateristic patterns.

2 Introduction

A network is a catalog of a system' s components often called nodes or vertices and the direct interactions between them, called links or edges. Some good examples of networks, are social networks like Facebook and Twitter. This article uses networks to represent the services offered by a bus company operating in Porto, with the intent of answering some questions like : how many bus stops are there, which street has the most lines going through it and how many times do i have to change bus so i can go from one side of the network to the other.

3 Data Gathering and Processing

3.1 Data source

The first decision made in this study wasn't what to study about the network, was to know were to get the data. The STCP website was a great place to start, as it has all the information on bus lines and bus stops. So, how do we get the information programatically? The best way to do it, is to read the answer the server gives the client when a request is made. Luckily, this answer is in form of JSON objects, which are easily parsable. The information the site gives about bus stops and bus lines is enough for the common user, but we need more. There are some aspects we need to derive if we want to do a complete study on the proposed network. Therefore, every piece of information was locally stored.

3.2 Data storing

While reading the information from the server answers, i noticed that each time i wanted to read the data i had to , programatically,connect to the site ,and that took too long to be viable.

A simple answer was to create two .txt files with the information gathered.

One of the files (AllLines.txt) had all the information about each busline. Each bus line is represented in a single line by a code, a direction, an integer that tells us the total number of stops in that line, and finally, all the stops that compose said line.

The other file (AllStops.txt) has all the information about all the stops. Each stop is represented in a single line by a stop code, an address, a zone, a name and a pair of floating point numbers that represent it's geographical location.

All of this information can be refreshed, as there are methods that do so.

There is the need for some structures in which to represent all this data we have, so it can be easily used to answer some of the questions raised above.

Using the JAVA programming language, and after analising the data, two structures came up as obvious:

- Stop, composed by:
  - a stopCode
  - an address

    - a zone
    - a name
    - a longitude
    - a latitude

  The stopCode is the ID of a stop,as it is unique. Zones in this network are parts of the city, this will be clear on the visual part of this study.

- BusLine, composed by:
    - direction
    - code
    - description
    - pubcode
    - LineStops

  The direction tells us which way a bus goes trough the line. The code and pubcode are different things. The pubCode is, as the name sugests, public, and it is how we humans distinguish lines from one another. The code variable however, is the server way to distinguish bus lines. The description is a sentence that indicates were the line begins and were it ends. LineStops is the group of stops a certain line serves.

After some processing , and some thinkering with gephi,there was a need to group all the stops we had, to facilitate the study. These were grouped by two different attributes. Their address and their code.

The address grouping was simple, as adresses are unique in this network. The structure BusStreet follows:

- Street, which is simply a String.
- stops, which is a list of all the stops in that street.
- longitude
- latitude

In a graph where a street is a node, we also need a group of edges, so we have the structure AddressEdge:

- src , which is the source street
- dest, the destination street
- weight, represents the number of lines that use this edge
- name

The code grouping requires some thought, as it is related to the direction a bus goes through a line. For instance, there happen to be two stops which codes are TSL1 and TSL2. This happens, because one serves buses that go in a direction, and the other serves buses in the opposite direction. Therefore, as the direction a bus goes in does not interest us, we can group these two stops into only one, and here a new structure arises. Introducing the Spot structure:

- code, is generated using the same logic from the example above
- stops, is an hashmap that contains all stops under the code.

Similarly to streets, there is also a need for a structure that represents interactions between Spots, therefore, SpotEdge:

- from is the source Spot
- to is the other Spot in the connection
- weight is the number of lines that use this connection
- name

These are the structures used to simplify this study.
Some files, six to be more precise, were not included in this subsection, as they were generated with JAVA to be used by gephi, and the generating methods will be described next.

3.3 Data processing

As mentioned before, Gephi was used to study the network. Gephi is a preety powerfull tool, but it requires some form of input. In this particular case .csv files. This subsection describes the methods that process the data (stored in any way described in the previous subsection) and generate the input gephi needs.
Still on the previous subsection, it was mentioned that there were 6 files left to describe. Three of these are groups of nodes, and the other three, groups of edges.

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 comes from another method.
The method makeAllLinesCSV was built with the intent of beeing able to make the .csv files for the edges of all bus lines. This idea was abandoned, but the method was repurposed. Besides making those files, it also returns an hashmap with all the edges it has seen, beeing the source of this second file.
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.

4 Results

This first image is the very first piece of information that our tool gave us, a visual representation of all 2416 stops in the network(not using any type of grouping), colored by zone.

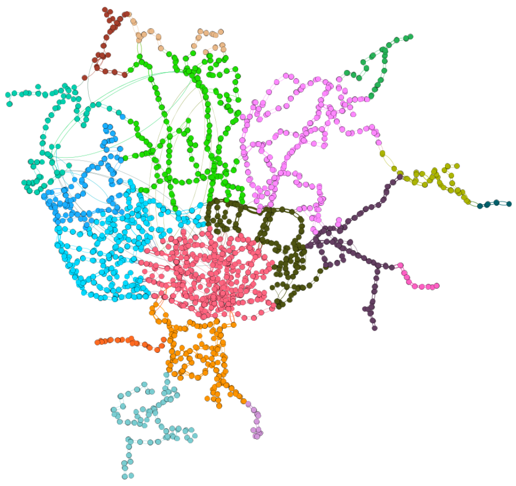
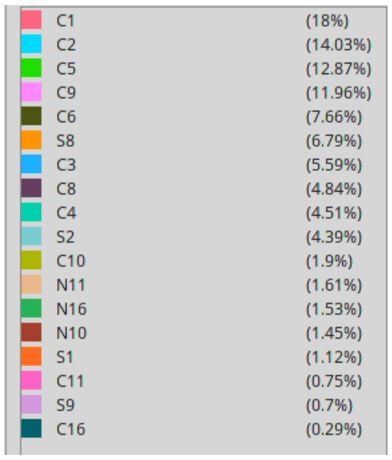


Figure 1: The network, without grouping, with stops colored by zones and shown according to geographical coordinates

The following image is the pallete used to color the network shown above, and it ends up giving us how large is each zone relatively to it.



🟥	C1	(18%)
🟦	C2	(14.03%)
🟩	C5	(12.87%)
🟪	C9	(11.96%)
🟫	C6	(7.66%)
🟧	S8	(6.79%)
🟦	C3	(5.59%)
🟪	C8	(4.84%)
🟩	C4	(4.51%)
🟦	S2	(4.39%)
🟨	C10	(1.9%)
🟫	N11	(1.61%)
🟩	N16	(1.53%)
🟫	N10	(1.45%)
🟧	S1	(1.12%)
🟪	C11	(0.75%)
🟪	S9	(0.7%)
🟦	C16	(0.29%)

Figure 2: A table with all the zones, sorted according to size,displaying each zone's color

The second question raised in the Introduction section of this report was "Which Stop has the most lines going through it and how many times?". For this, it's better to group the stops by code, and answer a diferent question : "Which Spot has the most lines going through it?", as a Spot is much more influential than a simple Stop. Te answer to this, is in the table below,extracted from spotNodes.csv were the 5 most populated Spots are represented:

Id	linesServed	totalStops
TRD	21	6
BCM	20	5
BS	18	8
AAL	17	6
CMO	17	4

Checking the Id's in the AllStops.txt file, we get, by ascending order: Carmo,Aliados,Bom Sucesso,Casa da Música in Boavista and Trindade topping the list.