Extracting Mass Transportation Networks from General Transit Feed Specification Datasets

Gergely Kocsis\textsuperscript{a} and Imre Varga\textsuperscript{b}

Department of IT Systems and Networks, University of Debrecen, Kassai str. 26, Debrecen, Hungary

Keywords: GTFS, Public Transport, Networks.

Abstract: In several smart-city applications the networks of the mass-transportation systems can be bases of investigations. In this paper we show how one can extract a network of connected stops from the General Transit Feed Specification (GTFS) feed of a given service provider. We have also implemented this process as a tool (gtfs2net) that is available for use at the GitHub page of the project. On of our most important finding is that since providers do not follow the specification in a coherent way regarding the use of parent stations the problem of close stops has to be manually handled. In order to show how our tool works in practice we have provided some extracted networks with their properties.

1 INTRODUCTION

Investigation of abstract networks of mass transportation providers is used frequently in smart city applications (Besenczi et al., 2021). Finding these networks however is not always trivial. There are several maps online (even some that are free and open to be used) that contain location of stops as well. From these sources it is possible to get a network of stops where the connection between them may be described by some sort of physical relation (e.g. a connecting road). In many cases however it is much reasonable to name two stops connected if there is a direct bus/tram/train trip between them.

One possible solution may be the use of the General Public Transit Feed Specification (GTFS) described timetables of transportation service providers. The General Transit Feed Specification is a current format for public transportation schedules and the related geographic information (Harrelson, 2021). Besides the opportunity of public transit agencies to publish their own transit data it lets software developers to write applications that use this data to help users in their daily lives. Numerous agencies provide their public GTFS datasets, but they create them in different manner (Wessel and Farber, 2019; Hansson et al., 2019; Kujala et al., 2018a; Sienkiewicz and Holyst, 2005; Braga et al., 2014). The native analysis of these sources is already a widely studied topic by the networks scientist community (Vuurstaek et al., 2020; Fortin et al., 2016; Wong, 2013; von Ferber et al., 2007; Gallotti and Barthelemy, 2015; Jiang, 2007; Lämmer et al., 2006; Porta et al., 2006). However the extraction of abstract networks from these sources is a much less known field. Moreover in most cases even if the extracted networks are available the process is not or only partially published (Kujala et al., 2018b).

In a simple GTFS based application a public transport stop is just a geographical location with a few meters in dimension. The feeds describe the connections of the stops as well. For a passenger however, who travels probably some kilometers, a short walk can be also included into the journey. This results in connections between close stops, which are not connected by the official datasets. As we found, it is much harder to find sources to build such networks on this basis.

Below we will show how one can extract abstract transportation networks from these feeds. In Section 2, we briefly introduce the important parts of the specification. In Sections 3 and 4 we show the process and the implementation aspects how to extract such an abstract network from a provided feed. In Section 5 we show some basic properties of some extracted networks to give an example how these networks can later be used. Finally in Section 6 we conclude our work and present our future plans.
2 GENERAL TRANSIT FEED SPECIFICATION

In the last 15 years the General Transit Feed Specification (GTFS) became the de-facto standard for describing timetables of public transportation services. The static component of the GTFS consists of comma-separated values in simple text files contained within a ZIP file. Each line of a file contains a record of the given data table, covering various required and optional fields. There are mandatory and optional files in the dataset. This section focuses on just some of the files.

- stops.txt: It describes locations (e.g. stops, stations) related to the public transportation network (PTN). Besides the official name of the place and its precise latitude and longitude other information can be stored in a record. One of these is the parent_station which can be used to define relationship among stops, platforms or boarding areas of the premise.
- trips.txt: The trips are directional sequences of stops connected by a transit vehicle during a specific time period.
- routes.txt: A transit route is a directionless set of trips concerning the same stops, so these trips are displayed to riders as a single service.
- stop_times.txt: Times when a vehicle arrives at and departs from stops for all the trips. These records connect the trips, the stops and timing information.

The records of the files are interconnected by identifiers. Stop time records contains stop_id and trip_id fields in order to connect to the respective records. The route_id field joins the routes and trips.

3 EXTRACTING ABSTRACT TRANSPORTATION NETWORKS FROM GTFS FEEDS

Even though the original aim of GTFS is to provide a standard for describing several aspects of public transportation for network scientists one of the most practical use of it is that an abstract transportation network can be extracted from the feed. However getting such a network is not as trivial as one would think at the first sight mostly because of two problems: i.) Local Transportation Providers sadly do not follow the specification fully in all cases or they follow it in different ways. ii.) It is not always exact what network scientists would like to mean under a node or an edge of the network.

To see the full picture let us see what is needed to build a network in an ideal case. Our desired network will have stops as its nodes and edges describing that any time in the timetable there is a direct connection between two stops. Thus to get the nodes we need to process the stops.txt file of the GTFS feed. Besides several other fields, a record containing a stop has the id of the stop (stop_id) serving as the unique identifier of it. In cases when we have only standalone stops and no groups of stops are presented this field is enough to have a node. However in many cases some of the stops are grouped because e.g. they are parts of a bigger station. This relation is described by the location type and the parent_station fields of the record. For an ordinary stop, the location type is 0. If it is a part of a bigger station its parent_station field contains a stop_id referring to the parent stop. A parent stop (parent_station) has a location type of 1. Note that the location_type of a stop can also take values of 2, 3 and 4 but as practice shows these values are much less commonly used.

Giving an answer to the first above raised question the nodes of our abstract network can usually be the stops except in cases when a stop is contained by a parent station. In this latter case the station itself can be the node instead.

As mentioned above the edges of an abstract network should be the connections between the stops. Unfortunately this information is not stored directly in the datasets. The easiest way to get it is to process the stop_times.txt file. In this file a stop of a trip of a vehicle is described in each record. After the fields describing the trip's properties the stop_id field here describes in which stop the vehicle stops. The stop_sequence tells what is the number of the stop in the trip. It is easy to understand therefore if we have read two records of the same trip with consecutive sequence numbers the stops mentioned in these records can be treated as being neighbors in a way that there is a directed edge from the stop with the smaller sequence number to the other.

The problem with the above described method of finding edges is that it does not count with the presence of parent stations. This means that for example if we have two stations with let us say three stops in both that are connected by some trips directly, we will not get a network with two nodes and two directed edges, but we will get eight nodes some edges directly between the stops of the stations and no edges between the stations themselves. Referring to the stops and stations this way is however misleading in most of the cases, so we suggest to use the parent stations of those
 Extracting Mass Transportation Networks from General Transit Feed Specification Datasets

Figure 1: The effect of using virtual stations on the abstract topology of bus stations. left: Bus stations (blue squares) around Debrecen University IT Campus and their connections (blue arrows). right: Stops merged by virtual stations (red circles) and their connections (red arrows). (Picture source: OpenStreetMap.org).

Extracting a network from a GTFS feed following the above rules means that as a result we will get a network with a number of nodes

\[ N = N_{\text{station}} + N_{\text{stop}} \] (1)

where \( N_{\text{stop}} \) means only those stops whose \textit{parent station} field is empty. Note that this number \( N \) may be lower than the original number of stops, since multiple stops may have been replaced by similar stations.

The set of edges also has a lower number of elements than the number of connections read from \texttt{stop times.txt} file. A connection described by two consecutive records of the file will describe a new edge only if there is no such edge added already to the network. Also note that if one or both ends of the edge is described by a stop having a \textit{parent station}, the edge will be added to the network so that it connects the parent station(s) of the stop(s) and not the original stop(s) as it was described above.

One can say that the above method is a reasonable way of getting the network from the feed and theoretically it is. The problem is that in many cases GTFS feed providers do not follow the proper structure in their feed (semantically), meaning that in several cases parent stations are not contained in the same way.

- In some cases there are absolutely no parent stations. Naturally this may be a real case for small enough companies.
- Other feeds contain parent stations but only for the real stations containing several stops (or platforms) in them and ignoring stops e.g. on two sides of the same road.
- Again some others say that close stops shall also be handled together by one parent station (e.g. two stops at two opposite site of the same road usually with similar names). Note however that these parent stations are not real buildings or locations in most of the cases.

It may be a topic of arguments which aspect is the proper use of the format. In some already existing solutions this aspect is not taken into account (Kujala et al., 2018b), from the point of view of getting the network from the feed however we found that maybe the best solution is to give the control to the scientists getting the network.

Namely, while processing GTFS data of a given transportation provider (stops, stop times, parent stations) we say that not only explicitly described parent stations take the place of stops of a station but also, if the distance between stations is smaller than a so called merging limit \( r \) we replace these stations by a \textit{virtual station} acting as the parent station of the affected stops.

The effect of the merging of stops by virtual stations is illustrated on Figure 1 using the map of bus stations around the IT Campus of University of Debrecen. Note that by the introduction of virtual stations the number of nodes in the network strongly decreases (from 17 to 8), while the number of edges shows a much consolidate decrease (from 14 to 13 – counting bidirectional edges as 2). This transaction also has an effect on the connectedness of the graph, since separate clusters of stations may be merged in the resulted network.

Defining the above method more precisely, a new virtual station will be used for all those stops \( s_a \in S \)
for which
\[ \exists s_b \in S \rightarrow d(s_a, s_b) < r \] (2)
where \( d(s_a, s_b) \) is the geographical distance calculated using the Haversine formula (Rosetta Code Community, 2021) that is based on the longitudinal and latitudinal positions of the stops that can be read from the stops.txt file of the GTFS feed. One may note that as a result of the transitive property of this relation, two stops that are more far than \( r \) can be contained by the same virtual station if they both have the same station in between a distance of \( r \).

### 4 IMPLEMENTATION ASPECTS OF THE EXTRACTION

Providing an out of the box tool to do the extraction we have implemented an out of the tool in Java that can process unzipped GTFS feeds and results a network as a .txt file described by the edges of the network. Namely in each line of the file an edge is listed by containing the starting and arriving stops separated by a comma.

```java
GTFSTools
+readStops(String input, Map<String, Stop>)
+readStopTimes(String input, stops, Map<String, GTFSTools.Stops>)
+printStopsAsNetworkToFile(String input, Map<String, GTFSTools.Stops>, outFile)

class Stop
    +name: String
    +lonpos: double
    +latpos: double
    +hasParentStation: boolean

public interface GTFSTools
    +registerCloseStopsAsOne(Map<String, Stop>, r: int)
    +registerCloseStopsAsOne(Stops, r: int)

Algorithm 1: Algorithm to merge close stops by virtual stations.

1. Input: Set of stops and parent stations, \( r \)
2. Output: Set of stops and virtual parent stations
3. BottomIdx = the last stop’s index in the list
4. for actIdx ∈ 0..stops.size do
5. if distance(stops[actIdx], stops[bottomIdx]) < \( r \) then
6. Add stops[actIdx] to the set of close stops
7. end if
8. for each stop in the list of close stops do
9. repeat steps 4, 5, 6
10. A natural outcome of this algorithm is that by increasing the value of the checked distance \( r \), we get less nodes in our resulted network. An interesting question is that what may be a reasonable value for \( r \) in order not to lose too many stops but still eliminate the cases when e.g. stops at two sides of the same road are handled as being independent (since maybe no common parent station has been added to them).

In order to be able to give a hint how to select the value of \( r \) we have checked the dependence of the number of nodes \( N \) on the distance \( r \) for several GTFS feeds (A detailed description of each feed and the sources of them are available at the GitHub of the project’s data (Kocsis and Varga, 2021b)).

Our findings are summed up on Figure 3. As one may observe increasing the merging distance \( r \) at the first some meters does not affect much the number of nodes but still elimi-
a station). As the trends show in most cases however somewhere around 100 − 200 meters the number of stops \( N \) starts to fall down rapidly showing that this distance may be the desired one describing close stations that may be merged in order to see a more valid picture of the network of mass transportation systems.

This finding is consistent to the intuitive guess based on studying mass transportation maps, that the distance of such stations should be somewhere around 150 meters. The exact value however is to be decided based on the actual data since special local properties may affect it.

One may also note on Figure 3 that while most of the curves move together following an inverse logistic shape, there are some of them that seem to have different behavior. Examining these cases however soon reveals that these exceptional cases are for transportation feeds describing non-local or non-exclusively-local transportation networks, like train, ferry and inter-city bus networks. Nevertheless drawing these data on a longer scale shows that their qualitative behavior do not differ (see Figure 3 inset). We see similar falling of the curves as before for these cases as well, just the place of it is around 2-4 kilometers. It has to be noted however that merging stops more than 2 kilometers far from each other may not have any practical use especially knowing that luckily in the case of train stations it is really rare that companies register the two directions as individual stops (not even grouped by a parent station).

5 INVESTIGATION OF SOME RESULTED NETWORKS

In order to see how our tool works on extracting networks from GTFS feeds in practice we have used it for several transportation feeds available online. We have collected these sources under the GitHub page of our project (Kocsis and Varga, 2021a), (Kocsis and Varga, 2021b) and also we have uploaded there the resulted abstract networks in .txt format. A concluding table of these extracted networks together with some basic properties of them are shown on Figure 4. Note that there are huge differences in the number of nodes of the networks for different sources. Some service providers have only a bit more than 100 nodes, while some of them have ten-thousands.

In most of the cases the union of close nodes (merging stops within \( r = 150 \)m distance by virtual parent stations) decreases the number of nodes almost to the half of the original node number in case of \( r = 0 \)m. This confirms the assumption that stops at the opposite side of roads, or being close to each other in different ways are quite frequent in the source data sets (note again the position of bus stops on the real map shown on Figure 1). These related stops are handled independently by most of the GTFS feeds however from the passengers’ point of view they usually mean the same location during their journey on the urban public transport system. Contrarily, there are some systems (like New York City subway or the Hungarian train system e.g.) that are almost unaffected by our algorithm, namely the merging process does not change too much the number of nodes (in case of \( r = 150 \)). These are however mainly railway systems, where the opposite direction traffic use the same platform or if there are multiple platforms, they are contained by parent stations already in the original transportation feed.

Some extracted networks in case of \( r = 0 \)m contain only one cluster of nodes, meaning that all the nodes are available from any nodes via a link sequence. This practically means that passengers can travel from a stop to any other by the given public transport system. Nevertheless a significant portion of the analyzed systems fall apart several isolated clusters, where there is no connection at all between the nodes of these separate clusters. It seems strange if a public transport systems does not provide any service to connect different regions, but we will see that usually this is just a side effect of the missing parent stations of close stops. Usually but not exclusively one of these clusters is much larger then others, so beside the giant cluster there are several minor clusters of stops and stations.
Figure 4: Basic properties of networks extracted from some example GTFS feeds for merging distance \( r = 0 \text{m} \) and \( r = 150 \text{m} \).

Note the huge level of change implied as a result of merging close stops (see the last two columns). \( N \): number of nodes in the network, \( L \): number of edges in the network, \( N_c \): number of clusters, \( S_g \): size of the giant cluster.

Our algorithm can transform this network by the merging of near stops to create a new virtual station. Our results show that using \( r = 150 \text{m} \) distance implies that almost all networks compose only one or two clusters. For a passenger this means that a few steps long walk can dramatically improve the connectivity of the network. It should be mentioned, that the only exception is the GTFS feed of Zürich, where the service provider operates services not exclusively in Zürich, but in further places as well resulting in small independent clusters. The numerical results can be seen on Figure 4.

6 CONCLUSIONS

In this paper we have given a basic introduction to the structure of General Transit Feed Specification highlighting the most important properties of it from the aspect of extracting a network of connected stops from the source feeds. Besides describing the process we have even implemented the extraction in a free to use tool.

We have put a special focus on the use (or not use) of parent stations in GTFS feeds. As a possible solution to handle the problem of close stops in the extracted networks we used the merging distance \( r \) to describe how close stops are to be handled as being stops of the same so called “virtual station”. In some example networks extracted from various GTFS feeds we have investigated the effect of increasing \( r \).

We have found that the intuitive value of \( r = 150 \text{m} \) is a reasonable choice also from the aspect of our numerical investigations. We have presented some basic properties of these networks at the end.

Our ongoing research now has a focus on the use of the tool providing more mass-transportation networks to be used by other scientists (with the description of these networks). Integrating the API of OpenMobilityData would be e.g. a very useful extension.

We also plan to further upgrade the tool to make the extracted network depend on more parameters. And also we would like to provide a simple Graphical User Interface to make the tool even more easy to be used. We have found that building a web service for this aim would not be worthy.

ACKNOWLEDGEMENTS

This work was supported by the EFOP-3.6.1-16-2016-00022 project.
The project is co-financed by the European Union and the European Social Fund.
REFERENCES


Ogami, Y. (2021). Fast algorithms for particle searching and positioning by cell registration and area comparison. Trends in Computer Science and Information Technology, 007-016(6(1)).

