Worldwide Subway Systems: Data Extraction, Topology, and Resilience

Xiaoqian Sun\textsuperscript{1,2} and Sebastian Wandelt\textsuperscript{1,2,*}

Abstract

Understanding and improving urban transportation networks is one of the key challenges in the 21st century, since the economy of a region largely depends on its accessibility. Existing studies on urban transport usually collect data by hand or directly receive them from network operators; making the data inaccessible for other researchers. This has three consequences: First, researchers spend a significant amount of time to obtain the data. Second, experiments often cannot be reproduced without having the same dataset. Third, results obtained for one network cannot be transferred to other networks easily.

In this study, we use public available data from Openstreetmap to extract the subway networks for more than 150 cities worldwide, and propose several techniques to solve data inconsistency problems. Moreover, we investigate the potential of this data for urban complex network research and provide a preliminary comparison of the topology and the resilience of subway networks. Our work contributes towards understanding and improving cities’ infrastructure from a complex network point of view.

Keywords: Subway networks; Data extraction; Resilience

1 Introduction

The understanding and improvement of urban transportation systems is one of the global challenges, since an efficient transportation network is one prerequisite for the well-functioning of a region. There are several modeling mechanisms for urban transportation systems, one of them is the view as complex networks [Von Ferber et al. 2009, Wandelt et al. 2015, Sun et al. 2017]. Complex network techniques have become very popular during the last decade, giving ground-breaking research on artificial and human-made systems modeled as a network of nodes interconnected by links [Newman2003, Barabási2016, Costa et al. 2007]. Existing studies on the analysis of urban transportation networks often rely on hand-curated data or data from network operators [Huynh et al.m 2016, Fielbaum et al. 2016]. In the former case, the preparation of data is rather time-consuming and it is often difficult to transfer results from one urban region to another, simply because of different data modeling and processing methodologies. In the latter case, one needs to rely on the cooperation of network operators to publish their infrastructure and schedule data online. While there has been a great progress in this area recently, following the Open Big Data movement, it is still difficult to obtain consistent and comparable transportation datasets for different urban regions. This limitation of data availability has two major consequences. First, the unavailability of data makes it very difficult to reproduce results of studies and use them in other scenarios. Second, with more data available to the public, more researchers will attempt to solve real-world problems with this data. Some work already geared towards making data for single regions available for research purposes [Gallotti and Barthelemy2014].
The objective of this study is to address these data availability and data management challenges by investigating the possibility to extract urban transportation complex networks from Openstreetmap (OSM), a community-based effort to create a freely available map of the world, ran by a large number of volunteers [Haklay and Weber2008]. Since 2006, the year of the establishment of the Openstreetmap Foundation, the quality of the data, as well as the coverage, have increased significantly, particularly in regions with high population densities [Hochmair et al. 2013]. This makes it possible to extract data for urban regions at a large scale and compare structural similarities and differences of networks.

The paper is structured as follows. Section 2 introduces our algorithms and implementations for extracting a useful complex network representation of subway systems and Section 3 reports our initial results for comparing more than 150 urban subway networks, with a particular emphasis on the resilience analysis. With Section 4 we summarize our results and discuss some future research directions.

2 Methodology

A naive method for extracting the subway network of an urban region does not provide sufficiently accurate and cleansed data for sound complex network analysis. Therefore, in the following section, we introduce a few data management and cleansing techniques, which allow us to extract a useful and representative snapshot of the subway network inside a region. Our evaluation on several urban regions worldwide (Barcelona, Beijing, Berlin, London, and New York) shows that the methodology yields accurate and stable results for distinct parts of the world.

2.1 Extraction of the base network

In a first step, we extract the base physical infrastructure network from OSM, by identifying a super set of required elements first, in order to carefully remove redundant and unnecessary elements afterwards. Initially, we iterate all relations within the region of interest (\(R_{\text{osm}}\)) and identify all relations with either \(\text{route}=\text{subway}\) or \(\text{subway}=\text{yes}\), denoted as \(R_{\text{subway}}\). For all relations in \(R_{\text{subway}}\), we extract their member way ids (added to \(W_{\text{subway}}\)) together with member node ids (added to \(N_{\text{subway}}\)). Furthermore, for all member ids of the relation, we assign a derived mode by propagating subway mode information from relations to ways and from ways to nodes. Next, we iterate over all ways in the region (\(W_{\text{osm}}\)) and identify all ways tagged with either \(\text{route}=\text{subway}\) or \(\text{subway}=\text{yes}\), plus all ways occurring in \(W_{\text{subway}}\). For each such way, we check, whether it is annotated as a station. If yes, we extract the center point of the way polygon and add this node to \(S_{\text{subway}}\). If not, we extract all way segments of the way and add them to \(L\). In the latter case, for all node member ids of the way, we assign a derived mode and added the relevant nodes to \(N_{\text{subway}}\). Finally, we iterate the subset of nodes in \(N_{\text{subway}}\) plus all nodes tagged as subway stations, and extract their geo-coordinates together with tag dictionary. If the node is tagged as a station, the node is added to \(S_{\text{subway}}\). In a cleanup step, we first remove all links and stations from the network, which are being tagged as under construction (having \(\text{disused}=\text{"yes"}\) or "construction" as a key of the dictionary). Second, we remove all stations from the network, whose derived mode is not equal to subway. After this pre-processing, we have the following data:

- \(N_{\text{subway}}\): A set of nodes which build the subway network by providing way points.
- \(L_{\text{subway}}\): A set of links which constitute the base subway network after extraction.
- \(S_{\text{subway}}\): A set of stations which are candidates for being subway stations in the network.
2.2 Aggregating links and adding disconnected stations

With the base network from the previous section, given by $N_{subway}$, $L_{subway}$, and $S_{subway}$, we address the problem of improperly/disconnected stations next. As discussed above, these stations are mainly introduced by mapping errors, where links are not accurately connected to a station, or when redundant stations are inserted after the initial layout of the infrastructure network was finished. In general, we want to attach stations to nearby links, whenever it is appropriate. In order to achieve this, we follow some simple rules:

- We identify a set of connectable stations. All these stations are disconnected (i.e. the degree is 0) and there is no other station within a given distance threshold. The rational is as follows: If there is a nearby connected station, then the disconnected station is very often redundant, i.e. has a similar name and location.
- For each link, we use an approximation to find the connectable stations and connect the station if it is within a given perpendicular distance. We use the latter distance, since way segments often pass by a station very close, while the spanning nodes of the segments are not close.
- A naive solution would be to compare all links with all connectable stations and check the distance. However, in order to avoid a quadratic time complexity, we use a KD tree [Error! Reference source not found.] over all connectable stations to find station candidates near way segments.

In Algorithm 2.1, we formalize this method. The distances were chosen empirically, by analyzing datasets for different regions. These results showed that stations should be considered connectable, if there is no other connected station within 200m. Moreover, we found that the perpendicular distance between disconnected stations rarely exceeded 50m. Finally, we remove all stations which are still disconnected after this step.

### Algorithm 1 Aggregation of links to nearby stations

```plaintext
Algorithm 1 Aggregation of links to nearby stations

- **Input:** Set of nodes $N$, set of stations $S$ and set of links $L$
- **Output:** Set of revised nodes $N_r$, set of stations $S$ and set of aggregated links $L_r$

1. Let $S_d$ be the disconnected stations in $S$ (i.e., with degree equal to 0)
2. $candidates = \emptyset$
3. Let $KDT_c$ be a kd tree over $S \setminus S_d$
4. for all $s \in S_d$ do
5.   use $KDT_c$ to find nearest connected station to $s$, store in $s_c$
6.   if $distance(s, s_c) \geq 200m$ then
7.     $candidates = candidates \cup \{s\}$  \hspace{1cm} $\triangleright$ Check if no other connected station nearby
8.   end if
9. end for
10. Let $KDT$ be a kd tree over $candidates$
11. $L_r = \emptyset$  \hspace{1cm} $\triangleright$ Loop over all links and connect nearest stations
12. for all $(a, b) \in L$ do
13.   use $KDT$ to find nearest station to $a$, store in $s_a$
14.   use $KDT$ to find nearest station to $b$, store in $s_b$
15.   if $s_a \equiv s_b$ then
16.     let $dist$ be the perpendicular distance from $s_a$ to the line between $a$ and $b$
17.     if $dist \leq 50m$ then
18.       add $\{(a, s_a, t), (s_a, b, t)\}$ to $L_r$  \hspace{1cm} $\triangleright$ Rewire links within aggregation distance
19.     else
20.       add $(a, b, t)$ to $L_r$
21.     end if
22.   end if
23. end for
24. Let $N_r$ be the subset of nodes from $N$ which occur in $L_r$
25. return: $(N_r, S, L_r)$
```

2.2 Aggregating links and adding disconnected stations

With the base network from the previous section, given by $N_{subway}$, $L_{subway}$, and $S_{subway}$, we address the problem of improperly/disconnected stations next. As discussed above, these stations are mainly introduced by mapping errors, where links are not accurately connected to a station, or when redundant stations are inserted after the initial layout of the infrastructure network was finished. In general, we want to attach stations to nearby links, whenever it is appropriate. In order to achieve this, we follow some simple rules:

- We identify a set of connectable stations. All these stations are disconnected (i.e. the degree is 0) and there is no other station within a given distance threshold. The rational is as follows: If there is a nearby connected station, then the disconnected station is very often redundant, i.e. has a similar name and location.
- For each link, we use an approximation to find the connectable stations and connect the station if it is within a given perpendicular distance. We use the latter distance, since way segments often pass by a station very close, while the spanning nodes of the segments are not close.
- A naive solution would be to compare all links with all connectable stations and check the distance. However, in order to avoid a quadratic time complexity, we use a KD tree [Error! Reference source not found.] over all connectable stations to find station candidates near way segments.

In Algorithm 2.1, we formalize this method. The distances were chosen empirically, by analyzing datasets for different regions. These results showed that stations should be considered connectable, if there is no other connected station within 200m. Moreover, we found that the perpendicular distance between disconnected stations rarely exceeded 50m. Finally, we remove all stations which are still disconnected after this step.
2.3 Merging stations

Following the previous steps, we have a subway network with all links passing through nearby stations. However, many stations are in fact redundant, because they have been introduced by different mappers or for modeling distinct purposes. From a complex network point of view, we are merely interested in one station per node. Therefore, we want to merge nearby stations into one node, as long as we have sufficient evidence that these distinct stations model the same object. The major question is how to decide whether two nearby stations should be merged. We follow a two-faced merging strategy here, based on two observations:

1. The closer two stations are, the more likely they represent the same real-world object.
2. Stations with similar names are more likely to represent the same real-world object.

The proximity of two stations can be measured by their geographical distance (using the Haversine formula \[[Error! Reference source not found.]\]). In order to measure the similarity of two names, we exploit the relative edit distance between the two names encoded as strings: The edit distance, also known as Levenshtein distance \[[Error! Reference source not found.]\], between two strings, \(string_1\) and \(string_2\), is defined as the minimum number of add, remove, replace operations on symbol level necessary to transform \(string_1\) into \(string_2\). For instance, the edit distance between 'Friedrichstraße' and 'Friedrichstrasse' is 2, since the 'ß' in 'Friedrichstraße' can be replaced by one 's' and the other 's' has to be added in addition. The edit distance is computed by standard algorithms following a dynamic programming model. The relative edit distance between \(string_1\) and \(string_2\) is then defined as \(\frac{ED(string_1, string_2)}{\max(|string_1|, |string_2|)}\). For example, the relative edit distance of 'Friedrichstraße' and 'Friedrichstrasse' is \(\frac{2}{16} = 0.125\), which means the strings are rather similar (with 0 indicating identity and 1 maximum dissimilarity).

Based on the concept of relative edit distance, the overall algorithm for merging stations is introduced next. First, we build a KD tree over all stations. Afterwards, we iterate over all stations and find the nearest neighbor stations (within 500m distance). We compute the spatial distance and relative edit distance, and merge the two stations only if \(\text{relativeEditDistance} \leq \frac{10}{\sqrt{\text{spatialDistance} \times 1000}}\). This formula has been verified by experiments on
many urban city networks and, as required, decreases the likelihood of merging with an increasing distance. For each station pair, we record whether they are mergeable or not. In a second step, we decide which stations to be merged (we prefer to keep stations with names over unnamed stations, and prefer stations tagged as station). Finally, we iterate all links and replace stations by their representative station after merging. The algorithm is formalized in Algorithm Error! Reference source not found..

**Algorithm 3 Extraction of logical network**

<table>
<thead>
<tr>
<th>Region</th>
<th>N</th>
<th>L</th>
<th>AvgDeg</th>
<th>Density</th>
<th>CC</th>
<th>ASPL</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>377</td>
<td>448</td>
<td>2.3</td>
<td>0.0063</td>
<td>0.09</td>
<td>19.2</td>
</tr>
<tr>
<td>Seoul</td>
<td>322</td>
<td>365</td>
<td>2.2</td>
<td>0.0071</td>
<td>0.02</td>
<td>19.2</td>
</tr>
<tr>
<td>Paris</td>
<td>303</td>
<td>356</td>
<td>2.3</td>
<td>0.0078</td>
<td>0.03</td>
<td>12.6</td>
</tr>
<tr>
<td>Shanghai</td>
<td>297</td>
<td>351</td>
<td>2.3</td>
<td>0.0080</td>
<td>0.02</td>
<td>14.4</td>
</tr>
<tr>
<td>Beijing</td>
<td>294</td>
<td>325</td>
<td>2.2</td>
<td>0.0075</td>
<td>0.02</td>
<td>16.6</td>
</tr>
<tr>
<td>London</td>
<td>257</td>
<td>302</td>
<td>2.3</td>
<td>0.0092</td>
<td>0.04</td>
<td>15.4</td>
</tr>
<tr>
<td>Madrid</td>
<td>249</td>
<td>284</td>
<td>2.2</td>
<td>0.0092</td>
<td>0.02</td>
<td>14.7</td>
</tr>
<tr>
<td>Tokyo</td>
<td>219</td>
<td>270</td>
<td>2.4</td>
<td>0.0113</td>
<td>0.06</td>
<td>10.5</td>
</tr>
<tr>
<td>Berlin</td>
<td>169</td>
<td>184</td>
<td>2.1</td>
<td>0.0130</td>
<td>0.02</td>
<td>14.1</td>
</tr>
<tr>
<td>MexicoCity</td>
<td>165</td>
<td>179</td>
<td>2.1</td>
<td>0.0132</td>
<td>0.03</td>
<td>12.5</td>
</tr>
<tr>
<td>Moscow</td>
<td>164</td>
<td>221</td>
<td>2.7</td>
<td>0.0169</td>
<td>0.17</td>
<td>8.9</td>
</tr>
<tr>
<td>New-Delhi</td>
<td>157</td>
<td>161</td>
<td>2.0</td>
<td>0.0131</td>
<td>0.01</td>
<td>18.3</td>
</tr>
<tr>
<td>Guangzhou</td>
<td>152</td>
<td>168</td>
<td>2.2</td>
<td>0.0146</td>
<td>0.01</td>
<td>12.8</td>
</tr>
<tr>
<td>Barcelona</td>
<td>151</td>
<td>167</td>
<td>2.2</td>
<td>0.0146</td>
<td>0.01</td>
<td>11.3</td>
</tr>
<tr>
<td>Duesseldorf</td>
<td>129</td>
<td>129</td>
<td>2.0</td>
<td>0.0156</td>
<td>0.00</td>
<td>15.5</td>
</tr>
</tbody>
</table>
Table 1: An overview over the Top 15 subway networks, according to the number of nodes in the giant component. Number of nodes (N), number of links (L), average degree (AvgDeg), density, average clustering coefficient (CC), average shortest path length (ASPL).

### 2.4 Creating the logical network

The previously described transformations have led us to a cleansed infrastructure network, with stations connected appropriately and redundant connections being removed. However, we still have many nodes modeling the detailed infrastructure, i.e., the physical layout of routes in between stations. From a logical, complex network point of view one is (usually) not interested in these, but rather like to have only links between stations. In order to create such a network, we extract the logical network, given the cleansed infrastructure network. Intuitively, we prefer to remove all non-station nodes, i.e., all nodes which are only inside the dataset for modeling way points. Simply removing these nodes from the network will disconnect all stations. Therefore, we introduce our method for transforming the physical infrastructure network into a logical network below.

First, we iterate over all stations in the network and find their direct neighbors, since we want to identify station pairs which are directly connected, without any other station on the way. Then, for each pair of stations, we compute the shortest geodesic path in the network. From the path, we extract the total length in km, as induced by the way segments spanning up the path. In general, we could add this link between these two stations to the logical network, weighted by the computed distance. In our experiments, however, it turned out that too many stations are connected. The reason is that railway-based vehicles can only follow infrastructure lines with physical limitations. In particular, rail-based systems will never turn on acute angles, but rather can only follow the infrastructure in limited curves. Based on our experiments with real urban networks, we decided to set the threshold for rejecting routes at 45°, which eliminates these spurious railway track joins. The complete algorithm is formalized in Algorithm Error! Reference source not found.. Please note that if the nearest stations to the nodes spanning up a link are not identical (checked in Line 15 of Algorithm Error! Reference source not found.), we do not attach either of the stations. In our evaluation, we have not found any problems with this approach. However, another way would be to iterate over stations and assign them to closest links, using a MX-CIF tree or similar data structures [Error! Reference source not found.].
Figure 1: A visualization of the Top 15 subway networks, according to the number of nodes in the giant component. The layout of the networks exhibits widely distinct structural properties.

TC "1 Topological properties of worldwide air transportation networks at six different aggregation levels." \l 1

3 Preliminary Evaluation

We evaluate our extraction algorithm on a set of cities obtained from Mapzen-provided Metro-extracts\(^3\). At the time of our download (June 18th, 2016), the list consisted of 713 cities around the world. We obtained the OSM files in PBF format, which is a compressed representation of the original XML-based OSM files, based on Google Protocol Buffer. The algorithms proposed in this study were then executed on each of the files separately, using the Python package imposm\(^4\). Please note that not all these 713 cities have subway networks. In our experiments, we obtained a total of 156 subway networks. For the sake of comparison, and in order to be able to compute particular network metrics, we report only results for the largest component in each of the subway networks.

---

\(^3\) The term "Metro" refers to metropolitan area and is not related to our goal of extracting a subway network. Mapzen provides a bounding-box based subset of OSM only, which is more convenient to use than the whole planet file, since the whole planet file is 40 GB. Link: [https://mapzen.com/data/metro-extracts/](https://mapzen.com/data/metro-extracts/)

\(^4\) Link: [https://imposm.org/](https://imposm.org/)
In Figure 2, scatter plots are presented showing the number of nodes versus the number of links (left) and average shortest path length (right). We can observe a strong linear correlation between the number of nodes and links, with the outlier being Moscow. There is a wide spread among average shortest path length for all subway networks, even for networks with comparable sizes.

In order to analyze the extracted networks, we investigate a set of standard network properties, which have been introduced in the literature. We revisit the definition of these properties below, for convenience. For an introduction into the science of complex networks, we refer the reader to several excellent surveys/textbooks [Error! Reference source not found., Error! Reference source not found., Error! Reference source not found.].

- **Average Shortest Path Length**: The average shortest path length of a network is the sum of the lengths of all shortest paths divided by the number of node pairs. This number represents the average number of steps it takes to get from one node to another.
- **Clustering Coefficient**: The clustering coefficient of a node is the ratio of existing links connecting a node’s neighbors to each other to the maximum possible number of such links.
- **Degree**: The degree of a node is the number of its direct neighbors.
- **Density**: The density of a network is defined as $\frac{2L}{N(N-1)}$, i.e., the ratio of the number of links $L$ to the number of possible links in the network.

In Table Error! Reference source not found., we provide an overview over the Top 15 subway networks, according to the number of nodes (=stations) in the network. In our experiments, New York has the largest number of stations (377) and largest number of links (448), which is more than...
twice as much as the subway network ranked number 10 (Mexico City). The average degree of the nodes in the network (the degree measures the number of neighbors for a node) is between 2 and 2.7. The density of all network is rather low, indicating that we deal with very sparse networks: The probability of two random chosen nodes being connected is rather small. Similarly, the average (local) clustering coefficient, a measure for transitivity of a graph, is very low. This shows that the majority of the networks is line-based, compared to triangle-based. The average shortest path length (average number of stops between two randomly chosen nodes) is below 20; and increases with the number of stations in the network. To sum up, these subway networks have very small density and clustering coefficient, but a significantly varying average shortest path length.

A visualization of the Top15 subway networks is shown in Figure Error! Reference source not found.. We can identify many similarities between the networks: All networks are clearly build around a center. The major difference between the networks is the coverage of the center. For instance, New Delhi and Duesseldorf only have a very few nodes in the center (main station), from which different subway lines leave into all directions. Cities like Paris and Beijing cover a much wider area in the center, where separate subway lines are only visible with a considerable distance from the most central node. On the other hand, the networks of Barcelona and Mexico-City do not reveal any clear network center.

Next, we analyze all the networks, not only the Top15. In Figure 2 (left), we analyze the relationship between the number of nodes and the number of links in the network. It can be seen that, with the exception of Moscow, all points form a nice line. In fact, the ratio between the number of links and the number of nodes averages neatly around 1.19, i.e., for each station in the network, in average 1.19 links are added.

In Figure 2 (right), we plot the average shortest path length versus the number of nodes in the network. The average shortest path length can be seen as a measure for the efficiency in the network: The shorter the length, the fewer hops a passenger needs to take between origin and destination. Interestingly, there is a wide spread in the average shortest path length, even for networks of similar sizes. This suggests that the arrangement of links in the network differs significantly between cities. An interesting example are the networks of New Delhi and Moscow: While both networks have a similar number of nodes, Moscow has ca. 60 links more. This small difference increases the efficiency of the network remarkably, as the average shortest path length in New-Delphi is two times higher than for Moscow.
One important property of complex networks is their robustness against random failures and targeted attacks of nodes in the network, which received much interest recently [Error! Reference source not found.]. Most networks are rather robust against random failures, but very vulnerable when the order of failing nodes is not at random. In particular, studies on the resilience of networks often use two network metric-attacking strategies to simulate failure of nodes in the network: Degree (number of neighbors of a node) and Betweenness centrality (how central is the node). In Figure 3, we show the results for the resilience analysis of the Top15 networks. Several interesting conclusions can be drawn. First, all networks break down rather fast: After the failures of 5–20 stations, the networks are broken into disconnected parts, and cannot provide reasonable transportation services anymore. Second, few networks are particularly vulnerable, e.g., New-Delphi and Duesseldorf. These networks have in common that they are built around few central main stations, whose failures will immediately disconnect large parts of the network. Other networks, e.g. New-York and Shanghai, are more robust against node failures. Third, the networks are more resilient to betweenness-based attacks than to degree-based attacks, which is not the case for many other transportation networks. The reason is that betweenness-based attacks preferably remove nodes in the center of a network, while degree-based attacks preferably remove transfer stations.

4 Conclusions

In this study we automated the extraction of worldwide subway networks from Openstreetmap. We showed that a naive extraction method does not lead to satisfying results, given inaccuracies during mapping and different mapping styles throughout the world. After applying appropriate data cleansing and repair techniques, the quality of the network increased significantly and, closely resembling the real networks, are useful for urban transportation research. Our preliminary evaluation on more than 150 subway networks around the world revealed interesting structural properties. Moreover, we found that subway networks are very susceptible to intentional failures; some of them being completely malfunctioning after only disabling a few stations.

This study gears towards the development of a global mobility dataset, accessible and manageable for all researchers, and eventually, a vision of shared public data among scientists working with urban transportation data [Error! Reference source not found.]. We hope that our extraction methodology can be improved further and yield satisfying results across different transportation modes. Another challenge is the extraction of networks for very large regions, which needs to addressed by more careful use of data management techniques [Error! Reference source not found.].

Acknowledgement

This study is supported by the National Natural Science Foundation of China (Grant No. 61650110516 and Grant No. 61601013).

References


