# r-2012-12-20

František Hajnovič
ferohajnovic@gmail.com

January 19, 2013

# Contents



# 1 Emergence of timetables

## Types of timetables and underlying networks

The purpose of this thesis is to investigate the properties of timetable graphs and to create distance (or rather earliest arrival) oracles that would take advantage of the good properties to efficiently answer EAP queries. One way to understand the properties of the timetables is to *describe their creation.*

The first thing we should notice when it comes to the creation of a timetable is that there is some sort of underlying transportation network - e.g. for buses it is the road network, for trains the rail network etc... [1] This transportation network is being used by each connection, thus its properties should influence the properties of the timetable. Table 1 summarises several different types of timetables and corresponding underlying transportation networks.

| Type of timetable | Underlying network |
|---|---|
| Regional bus | road network |
| Country-wide rails | railway network |
| Underground rails | underground railways |
| Public transportation (buses) | road network (city) |
| Airlines | none (complete graph on airports) |

Table 1: Types of timetables and corresponding underlying transportation networks

*Note:* We would be able to find more types of timetables, each with at least slight differences with respect to the others. However, for our purposes, the mentioned five types are sufficient.

First thing to realize is that the **underlying transportation network** is something else then the **underlying graph** of a given timetable. The former is really a map of the infrastructure used by the carriers of the timetable (nodes are intersections, edges are roads/rails between them...) and the nodes do not have to be equivalent to stations of the timetable. The latter is an oriented graph

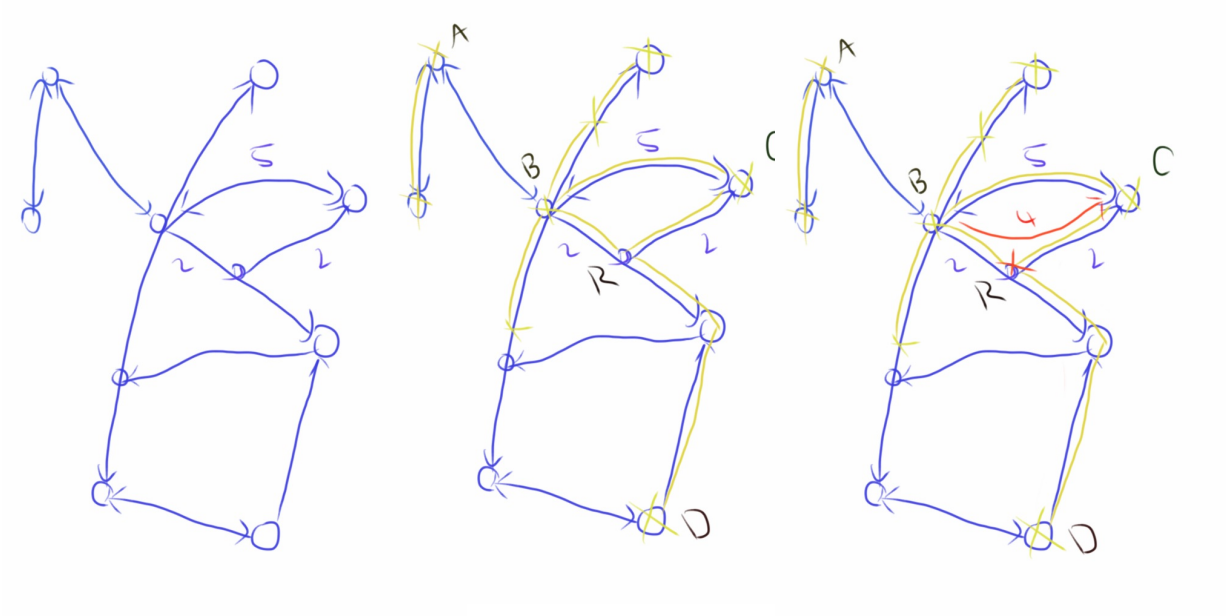
[1] Exception may be the airline timetable which does not really exploit an existing transportation network. However, we may simulate the underlying transportation network as a complete graph with nodes being the airports.

with stations being the nodes and arcs specifying from where to where there can be an elementary connection. Thus generating timetables on top of the *graph of the underlying network* is not the best approach.

Let us take e.g. the road network - there are quite efficient algorithms to find shortest paths in road networks. Their functionality depends on several good properties of road networks. If we would like to show that these properties are present in the graphs representing e.g. regional bus timetables (which have road network as the underlying transportation network), we would have to show the preservation of the properties at two places:

$$\textit{Underlying network} \xrightarrow{a)} \textit{Underlying graph} \xrightarrow{b)} \textit{Timetable}$$

You can think of this as of a game with two levels of difficulties. The more difficult version is the following: We have an underlying network of some timetable that we do not know. Our task is to choose stations on this network and then build a timetable connecting these stations so that it reflects the original timetable as much as possible. A bit easier version is the one, when we already know the stations and the pairs of stations between which there are elementary connections.
So in the **step *a)*** we first have to think where do we want to have stations and which of them should be connected in the timetable. The *stations may be placed* in the nodes of the network, or on edges, in which case they cut the edge in two and create a new node. Then we *specify the infrastructure* which is simply marking the arcs that will be used. Next, we *remove* from the graph each *node on which we have not placed a station*, and add new edges between each pair of its neighbours to preserve their connectivity and distance. And finally, we keep for each set of arcs from $X$ to $Y$ only the one with minimum length. After this, we have an underlying graph for a timetable with stations as we chose them and with elementary connections possible according to the arcs of the graph. See picture 1 for demonstration.




(a) Underlying network (b) Chosen stations and infrastructure (c) Removed non-station nodes

Figure 1: Step *a)*. First we have the underlying transportation network *(a)*. Then we choose the stations and infrastructure (marked yellow, *(b)*). Note that there will be no connection between $A$ and $B$ in the timetable since they are not connected. Finally we remove node $R$, since it is a non-station node and we insert a new arc from $B$ to $C$ to preserve the distance *(c)*. Similarly we would do between $B$ and $D$ and $C$ and $D$.

In the **step *b)***, we have to build a timetable respecting the locations of the stations and the connections between them. We will discuss this later. In what follows, we will be interested, if by restricted selection of stations and infrastructure, we can preserve the good properties through step *a)* - that is from underlying network to the underlying graph.

## Step a) - discussion

A question is - how do we choose stations on the transportation networks? In a real-world scenarios, the placement of stations is simply determined by the necessity of transportation from a given place - thus there will most probably be many stations near (or in) the big cities, holiday resorts etc. Unless we have a geographic representation of the transportation network and we consult external source of information, we are unable to deduce the right placement of the stations.

However, even the mere graphical representation of a transportation network provides some clues. E.g., if we see a **cluster of many nodes** connected by short roads, we most probably found a city (or another important place). The size of a cluster would be influenced by the size of the city, which is usually proportional to the number of stations found within.

Another idea is to take nodes which form **separators** (or are part of small separators) in the network. These nodes are likely to be important as they connect different parts of the map and we could expect a station being built in such node, though a station of low importance (e.g. a station at the borders of two states, used for transfers). Note: a separating *edge* is intuitively unlikely to carry a station, especially a longer edge, as it is usually connecting bigger components (e.g. a road in a desert between Las Vegas and Los Angeles) .

We could also try to place stations based on the **betweenness centrality measure** of a node, which is equal to the number of shortest paths passing through the specific node. In cities, these are likely to be used as bus stations and generally, these nodes would be ideal for transit stations.

## Step b) - discussion

As for the step *b)*, there are numerous decisions that influence the properties of the resulting timetable. Let us list the most important ones:

- **Usage of express lines**. Express lines were defined in *r-12-11-08* and if we allow them, we can add an elementary connection also between two stations *not* connected by an edge in the underlying graph. However, we have to be careful about this, as excessive usage of express line could completely hide the structure of the underlying graph. One possibility is to assign levels to stations (a plausible option would be such that the distribution of the levels would be scale-free) and make express lines only between pairs of stations with the same level (or with a difference of 1 in levels).
- **Specifying length of connections**. Here we could parametrize, how much do we hold on to the cost of the edges of the underlying graph. For example, if the costs are in *km*, we can find highways that, though longer, provide quicker connection times then some shorter regional roads. However, generally a several times longer road would also yield several times longer connection times.
- **Overtaking connections**. Allowing overtaking connections usually complicates the resulting timetable and is meaningful only if we consider also other aspects of the travelling then the total time (such as price, quality of services...).
- **Time-expanded degree of a node**. Time-expanded degree of a node is the number of events (departure or arrival of a connection) happening in a node with respect to the timetable. Important stations are likely to have high time-expanded degree.

# 2 Data analysis

Following is the simple analysis of the data. Results were obtained by running *ttblazer* application.

## UG

| Name | Type | # nodes | # arcs | Load time |
|---|---|---|---|---|
| ug_cpza.ug | Regional bus | 1128 | 2034 | 0.4s |
| ug_cpru.ug | Regional bus | 877 | 1784 | 0.4s |
| ug_zsr.ug | Country-wide rails | 233 | 588 | 0.1s |
| ug_london.ug | Underground rails | 321 | 732 | 0.1s |
| ug_svk.ug | Road network | 181386 | 425829 | 7.754s |

Table 2: Underlying graphs - main properties

| Name | # nodes | # arcs | Avg deg. | Min deg. | Max deg. | Degrees | Analysis time |
|---|---|---|---|---|---|---|---|
| ug_cpza.ug | 1128 | 2034 | 1.80319 | 0 | 24 | 0: 78x, 1: 536x, 2: 297x, 3: 115x, 4: 48x, 5: 25x, 6: 10x, 7: 7x, 8: 5x, 9: 2x, 10: 1x, 12: 1x, 14: 1x, 15: 1x, 24: 1x | 0.0s |
| ug_cpru.ug | 877 | 1784 | 2.03421 | 0 | 17 | 0: 54x, 1: 405x, 2: 202x, 3: 86x, 4: 55x, 5: 34x, 6: 17x, 7: 11x, 8: 3x, 9: 2x, 10: 1x, 11: 1x, 12: 1x, 13: 2x, 14: 1x, 16: 1x, 17: 1x | 0.0s |
| ug_zsr.ug | 233 | 588 | 2.52361 | 0 | 12 | 1: 42x, 2: 119x, 3: 28x, 4: 14x, 5: 14x, 6: 6x, 7: 3x, 8: 1x, 9: 1x, 12: 2x | 0.0s |
| ug_london.ug | 321 | 732 | 2.28037 | 0 | 7 | 0: 3x, 1: 25x, 2: 230x, 3: 25x, 4: 27x, 5: 5x, 6: 3x, 7: 3x | 0.0s |
| ug_svk.ug | 181386 | 425829 | 2.34764 | 0 | 6 | 0: 96x, 1: 46469x, 2: 34998x, 3: 90084x, 4: 9586x, 5: 150x, 6: 3x | 0.25s |

Table 3: Underlying graphs - degrees

| Name | # nodes | # arcs | Connected | # of conn. comps | Component sizes | Analysis time |
|---|---|---|---|---|---|---|
| ug_cpza.ug | 1128 | 2034 | false | 21 | 1108, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1 | 0.2s |
| ug_cpru.ug | 877 | 1784 | false | 7 | 871, 1, 1, 1, 1, 1, 1 | 0.1s |
| ug_zsr.ug | 233 | 588 | true | 1 | 233 | 0.0s |
| ug_london.ug | 321 | 732 | false | 4 | 318, 1, 1, 1 | 0.1s |
| ug_svk.ug | 181386 | 425829 | false | 871 | 178187, 134, 105, 30, 29, ... | 4.478s |

Table 4: Underlying graphs - connectivity

| Name | # nodes | # arcs | Strongly conn. | # of str. conn. comps | Str. comp. sizes | Analysis time |
|---|---|---|---|---|---|---|
| ug_cpza.ug | 1128 | 2034 | false | 663 | 253, 37, 22, 11, 11, 10, 10, 9, ... | 01s |
| ug_cpru.ug | 877 | 1784 | false | 520 | 245, 16, 11, 10, 9, 8, 7, 7, 6, 6, 5, ... | 0.1s |
| ug_zsr.ug | 233 | 588 | false | 9 | 225, 1, 1, 1, 1, 1, 1, 1, 1 | 0.0s |
| ug_london.ug | 321 | 732 | false | 4 | 318, 1, 1, 1 | 0.1s |
| ug_svk.ug | 181386 | 425829 | false | 1192 | 177667, 128, 105, 105, 47, 45, 30, 29, ... | 4.478s |

Table 5: Underlying graphs - strong connectivity. Obtained with Tarjan's Algorithm

## TT

| Name | Type | # el. conn. | Load time |
|---|---|---|---|
| tt_cpza | Regional bus | 61747 | 4.451s |
| tt_cpru | Regional bus | 38540 | 2.275s |
| tt_zsr | Country-wide rails | 932052 | 66.6668s |

Table 6: Timetables - main properties

# TE

| Name | Type | # nodes | # arcs |
|---|---|---|---|
| te_cpza | Regional bus | 59276 | 119857 |
| te_cpru | Regional bus | 36458 | 74070 |
| te_zsr | Country-wide rails | 1706077 | 2637896 |

Table 7: Time-expanded graphs - main properties

# TD

| Name | Type | # nodes | # arcs |
|---|---|---|---|
| td_cpza | Regional bus | 1108 | 2839 |
| td_cpru | Regional bus | 871 | 2487 |
| td_zsr | Country-wide rails | 233 | 588 |

Table 8: Time-dependent graphs - main properties

# 3 Open points

- What to focus on during X-mas?

# 4 To do

**Theory:**

- Properties propagation in step a), step b)...

**Practice:**

- United airlines extract data, California road netw., public transport - buses
- Analyse properties: HD, separator, betweenness, scale-free distr., planarity, avg. distance, degrees for TE
- Building timetables from UG
- Machine learning