



A Toolchain for Generating Transit Maps from Schedule Data

Patrick Brosi

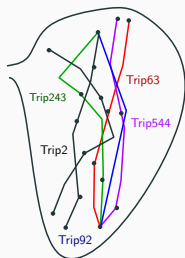
University of Freiburg

Schematic Mapping Workshop 2022 - Bochum, Germany

April 20-22, 2022

Motivation (1/2)

Goal: given some schedule dataset, render a (schematic?) transit map



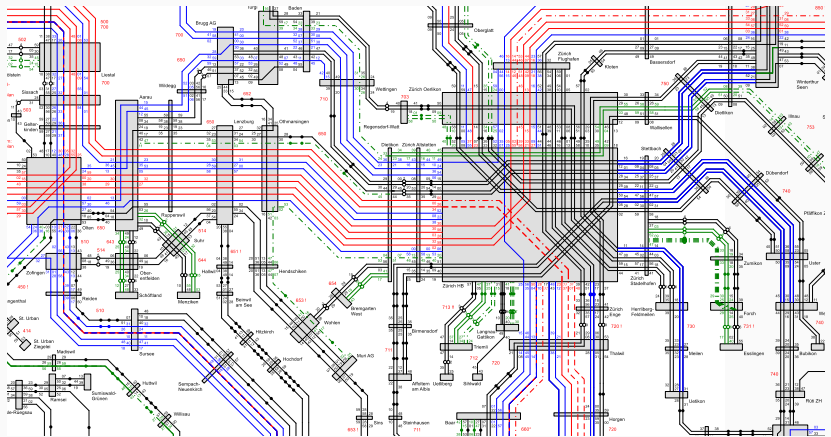
“Bag of trips”



- **Well researched:** schematization, line-ordering, labeling, esthetic aspects, ...
- **Focus today:** hands-on presentation of a **software suite** to achieve this goal.

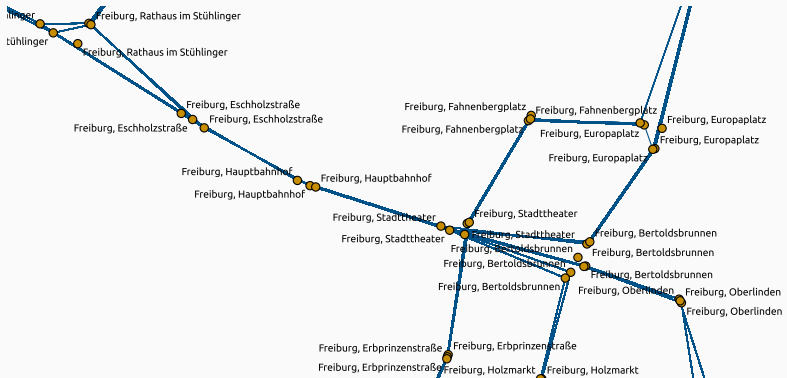
Motivation (2/2)

“Just render it automatically”



Schedule Data as Input

- Typical input data for research methods: **graph embedding** of a transit network
- Usually, **this** is what schedule data gives you:



Toolchain Overview

1. Raw schedule data		GTFS
2. Geographical line courses	standalone {	pfaedle
3. Overlap-free graph of network	LOOM {	topo
4. Line-Ordering optimization		loom
5. Optional Schematization		octi
6. Rendering		transitmap

Code: <https://github.com/ad-freiburg/pfaedle>
<https://github.com/ad-freiburg/loom>

pfaedle - Generate Geographical Line Courses

Input: GTFS data, OpenStreetMap (OSM) data

Output: GTFS data with geographical line courses

-
- Map-matching problem
 - We use a hidden Markov model based approach¹

Usage:

```
pfaedle -x freiburg.osm freiburg-gtfs
```

¹Bast and Brosi, “Sparse map-matching in public transit networks with turn restrictions”.

topo - Extract Overlap-Free Line Graph

Input: Line-labeled graph (line graph) of transit network

Output: “Free” line graph without overlaps

-
- Map-construction problem (typical use case: extract road network from collection of GPS traces)
 - We use a constructive approach similar to an approach by Cao and Krumm²

Usage:

```
topo < freiburg.raw.json > freiburg.json
```

²Cao and Krumm, “From GPS traces to a routable road map”.

loom - Line-Ordering Optimization

Input: Line graph

Output: Line graph with line label orderings which minimize (weighted) line crossings / separations

-
- Combinatorial optimization problem (MLCM), we only allow crossings at nodes (MLNCM)
 - Optimization approaches described here³

Usage:

```
loom < freiburg.json > freiburg.opt.json
```

³Bast, Brosi, and Storandt, "Efficient Generation of Geographically Accurate Transit Maps".

octi - Line Graph Schematization

Input: Line graph

Output: Schematic variant of the line graph

-
- Uses a grid-based schematization approach, currently supports ortholinear, octilinear, hexalinear, and orthoradial layouts
 - Optimization approaches described [here](#)⁴ and [here](#)⁵

Usage:

```
octi < freiburg.opt.json > freiburg.octi.json
```

⁴Bast, Brosi, and Storandt, “Metro Maps on Octilinear Grid Graphs”.

⁵Bast, Brosi, and Storandt, “Metro Maps on Flexible Base Grids”.

transitmap - Line Graph Rendering

Input: Line graph

Output: SVG drawing of the line graph

-
- SVG allows for easy edit, high-quality print, and use in web maps
 - Allows easy styling via CSS (which can be included in the GeoJSON file)
 - Can also be used in web maps
 - Approach: see here⁶

Usage:

```
transitmap < freiburg.octi.json > freiburg.octi.svg
```

⁶Bast, Brosi, and Storandt, "Efficient Generation of Geographically Accurate Transit Maps".

Pipeline Approach

- We took the **chain** metaphor seriously: LOOM tools can be plugged together via Unix pipes
- Between each step, the tools output and read a GeoJSON graph
- Allows for easy modification / extension

```
gtfs2graph <gtfs> | topo | loom | octi | transitmap > map.svg
```

Who (do we hope) is this for?

1. **Researchers** who want to try their own methods
2. **Map designers** for fast prototyping
3. **Schedule planers** for visualization of schedules

Rough Edges - What doesn't work

1. pfaedle only accepts XML OSM files
2. All tools only accept **extracted** GTFS feeds - not ZIPs.
3. Labeling is **preliminary** - for schematic maps, labeling should be part of the schematization
4. Enlargement of high-density areas in schematic maps
5. Schematic maps are based on a heuristic approach, which is a bit unstable - small changes to the input may result in a completely different layout.
6. Very large networks (London, New York) often show constraint violations, ILP approach takes too long (hours)
7. Tariff zones?
8. **Holy grail**: interactive map editor

The background of the slide features a complex network of multi-colored lines (blue, green, yellow, orange, purple, pink) that branch and merge. Small white circles are placed at various points along these lines, representing nodes or data points. Some lines have white rectangular markers with diagonal slashes. The overall style is clean and modern, suggesting a technical or scientific theme.

Thank you!

<http://loom.informatik.uni-freiburg.de>

<http://octi.informatik.uni-freiburg.de>

Activity Ideas

If you are **familiar** with installing Unix tools:

1. Go to <http://ad-research.cs.uni-freiburg.de/smw> and follow the instructions, try out some examples
2. Installation can also be done via Docker for minimal dependencies

If you are **not familiar** with Unix tools:

1. You can browse the render example at <https://octi.cs.uni-freiburg.de/> and <https://loom.cs.uni-freiburg.de/>
2. There you also play around with different layouts or methods
3. There is a JS implementation of the *octi* approach (by Tim Janiak) here: <https://ruhr-uni-bochum.de/schematicmapping/janiak>
4. You can play around with the results of a *pfaedle* run in Germany (and other areas) here: <https://travic.app>

Also feel free to ask me any questions!