A Toolchain for Generating Transit Maps from Schedule Data

Patrick Brosi
University of Freiburg
Freiburg, Germany
brosi@cs.uni-freiburg.de

***Abstract*—We give a quick overview over *LOOM*, a software toolchain which aims to render transit maps from raw schedule data fully automatically. The maps can either be geographically correct or schematic (using e.g. octilinear, orthoradial, or hexalinear layouts). To ensure extensibility, the toolchain is implemented as a Unix pipeline, with individual tools performing well-defined subtasks (e.g. extracting the network graph from schedule data, avoiding overlapping segments, finding a schematic layout, finding the optimal line ordering on the network segments, and rendering the map as an SVG file). An auxiliary tool to add missing geographical line courses to schedule data (*pfaedle*) is also presented.**

***Index Terms*—transit maps, schedule data, graph drawing**

I. Overview and Objectives

The problem of automatically drawing transit maps (or metro maps) following various layouts has been extensively studied in the past two decades (see [1] for a recent survey). However, a full toolchain to generate such maps automatically has been missing so far. The tools presented here aim to fill this gap. We demonstrate tools to (1) enrich schedule-data with geographical line courses (if missing), (2) extract a line-labeled network graph from this schedule data, (3) generate a network graph without segment overlaps, (4) find schematic drawings of the network graph, following ortholinear, octilinear, hexalinear, or orthoradial layouts, (5) find segment line-orderings which optimize the (weighted) number of line crossings and/or separations, and (6) render the map as an SVG graphic which can be directly used in print or as a map overlay. Each step in this toolchain is optional, allowing e.g. for geographically accurate transit maps if step 4 is omitted.

II. Methodology

To generate missing geographical line courses for schedule data, we use *pfaedle*, a tool for map-matching schedule data first published in [2]. *LOOM*, the software toolchain for generating transit maps from schedule data, consists of the tools *gtfs2graph*, *topo*, *loom*, *octi*, and *transitmap*. The tools *gtfs2graph*, *loom*, and *transitmap* are based on methods first published in [3] and [4]. *octi* is based on methods published in [5] and [6].

All tools have been made publicly available on Github[1][2] and expect the schedule data to be given in the GTFS format [7].

[1]https://github.com/ad-freiburg/pfaedle
[2]https://github.com/ad-freiburg/loom

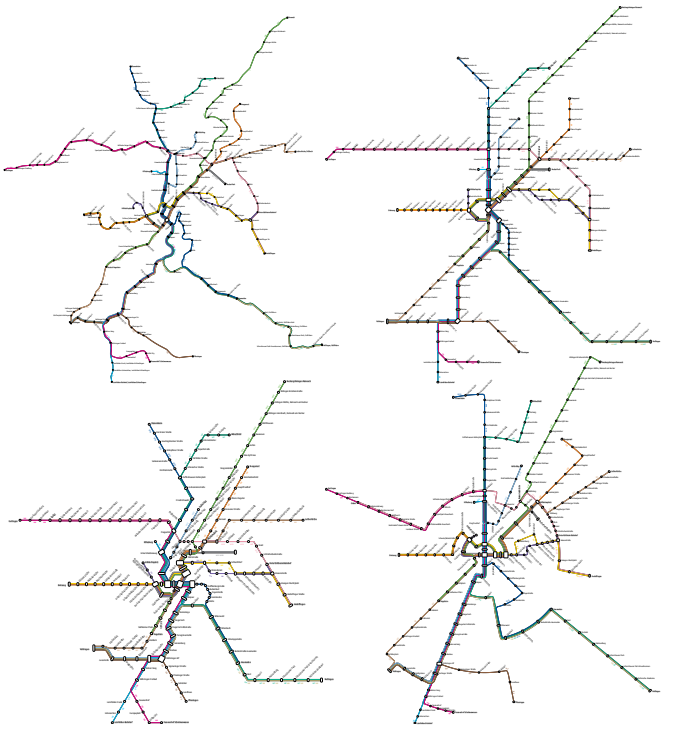

Fig. 1. Geographically correct, octilinear, hexalinear, and orthoradial transit maps rendered from schedule data using our toolchain.

The individual tools and their usage will be briefly described in the following sections. For details on their inner workings, we refer to the publications given above. For installation instructions, we refer to the respective `README` files.

A. pfaedle - Map-Matching GTFS Data

A problem with real-world schedule data is that the geographical line courses (shapes) are often missing. This is not only a problem when geographically accurate transit maps are generated, but may also impede the bundling of lines sharing a common course in schematic maps. *pfaedle* map-matches GTFS data to OpenStreetMap (OSM) map data and outputs a new schedule dataset with shapes Once *pfaedle* is installed, shapes for an example GTFS feed of Freiburg in folder `freib` can be extracted from an OSM file `freib.osm` (holding the OSM data for the Freiburg region) as follows:

```
pfaedle -x freib.osm freib
```

B. gtfs2graph - Extracting Network Graphs from GTFS Data

To extract a network graph from the schedule data, *LOOM* offers the tool *gtfs2graph*. Given a GTFS feed `freib`, the following command extracts a graph for the contained tram network and writes it as a GeoJSON file to `freib.raw.json`.

```
gtfs2graph -m tram freib > freib.raw.json
```

C. topo - Extract Free Network Graphs

Graphs generated by *gtfs2graph* have edge segments for every trip contained in the schedule data. These segments typically show great overlap. Making this graph overlap-free can be considered a map-construction problem. *LOOM* offers the tool *topo* for this. The following command generates an overlapping-free graph `freib.json` from the input graph given in `freib.raw.json`.

```
topo < freib.raw.json > freib.json
```

D. loom - Line-Ordering Optimization

A key aspect of the readability of a transit map are the line orderings on the network segments. The tool *loom* can be used to optimize these orderings for a giving input network as follows:

```
loom < freib.json > freib.opt.json
```

E. octi - Schematization of Network Graphs

Maps used in print are typically schematic. The tool *octi* reads a network graph and outputs a schematic version:

```
octi < freib.opt.json > freib.octi.json
```

The default layout is octilinear, but *octi* can also follow ortholinear, hexalinear, or orthoradial layout. Layouts can be selected via the `-b` (*b*ase graph) parameter:

```
octi -b orthoradial < freib.opt.json
```

octi may also approximate the geographical line courses in the schematic drawing by setting the parameter `--geo-pen` to a valuate greater than zero. It can also consider obstacles given as polylines or polygons in a GeoJSON files specified via the `--obstacles` parameter:

```
octi --obstacles obst.json < freib.opt.json
```

Example maps generated by *octi* using various different layouts can be found online[3].

F. transitmap - Transit Map Rendering Engine

The network graphs can be rendered into an SVG map with the *transitmap* tool:

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

It is possible to manually add CSS styling to individual lines in the GeoJSON file. The line width and line spacing may be specified via `--line-width` and `--line-spacing`. Labeling (still preliminary) can be added via the `-l` option.

[3]https://octi.cs.uni-freiburg.de/

G. Full Pipeline

Given a GTFS feed in some folder `freib`, an octilinear map of the contained tram network can thus be generated with the following command:

```
gtfs2graph -m tram freib | topo | loom | octi
    | transitmap > freib.svg
```

H. Web Maps

The SVG maps can be directly used as overlays in web map libraries like Leaflet[4]. Because of the SVG format, it is also very easy to make them interactive (for example, to add hover effects or add click listeners to individual lines and stations). We provide examples under https://loom.cs.uni-freiburg.de. The code of this website is also publicly available[5].

III. Outcomes and Benefits

We hope that the toolchain briefly introduced above might provide a basis for further research and implementation work. It would for example be easy to add a tool to locally enlarge network graphs before schematization to arrive at a more uniform map density. For professional end-users and designers of transit maps, we hope that the tools will be useful for fast map-prototyping, or for the generation of geographically correct web maps. The generated SVG graphics can be edited easily and might thus also form the basis for further polished manually designed maps.

[4]https://leafletjs.com

[5]https://github.com/ad-freiburg/loom-eval

References

[1] H. Wu, B. Niedermann, S. Takahashi, M. J. Roberts, and M. Nöllenburg, "A survey on transit map layout - from design, machine, and human perspectives," *Comput. Graph. Forum*, vol. 39, no. 3, pp. 619–646, 2020. [Online]. Available: https://doi.org/10.1111/cgf.14030

[2] H. Bast and P. Brosi, "Sparse map-matching in public transit networks with turn restrictions," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*. ACM, 2018, pp. 480–483. [Online]. Available: https://doi.org/10.1145/3274895.3274957

[3] H. Bast, P. Brosi, and S. Storandt, "Efficient generation of geographically accurate transit maps," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*. ACM, 2018, pp. 13–22. [Online]. Available: https://doi.org/10.1145/3274895.3274955

[4] ——, "Efficient generation of geographically accurate transit maps," *ACM Trans. Spatial Algorithms Syst.*, vol. 5, no. 4, pp. 25:1–25:36, 2019. [Online]. Available: https://doi.org/10.1145/3337790

[5] ——, "Metro maps on octilinear grid graphs," *Comput. Graph. Forum*, vol. 39, no. 3, pp. 357–367, 2020. [Online]. Available: https://doi.org/10.1111/cgf.13986

[6] ——, "Metro maps on flexible base grids," in *Proceedings of the 17th International Symposium on Spatial and Temporal Databases, SSTD 2021, Virtual Event, USA, August 23-25, 2021*. ACM, 2021, pp. 12–22. [Online]. Available: https://doi.org/10.1145/3469830.3470899

[7] "GTFS Reference," https://developers.google.com/transit/gtfs/reference, accessed: 2022-04-12.