Australasian Transport Research Forum 2023 Proceedings
29 November – 1 December, Perth, Australia
Publication website: http://www.atrf.info

Transport planning network and services extraction: PLANit open-source toolkit

Mark P.H. Raadsen*, Michiel C.J. Bliemer

Institute of Transport and Logistics Studies, The University of Sydney

Email for correspondence (presenting author*): mark.raadsen@sydney.edu.au

1. Introduction

Transport modelling requires a plethora of data/inputs before the actual modelling can take place. Travel demands, transport infrastructure networks, public transport services, zonings, all need to be ingested, processed, and made fit for purpose beforehand. This is a laborious task, often taking more time than the modelling itself. In this work, we contribute to reducing the efforts required for this task, with a focus on the supply side inputs of transport modelling, e.g., infrastructure networks and public transport services. We introduce an Australian developed open-source tool to extract infrastructure and (public transport) services in an easy, yet comprehensive, and highly configurable manner. While in this work we focus on leveraging Open Street Map (OSM) and GTFS data sources, the general purpose and architecture aids any data from which network and/or public transport information can be extracted. This work has been carried out as part of the Australian Transport Research Cloud initiative (ATRC, n.d.), with the aim to let researchers and practitioners focus more on their actual applications and modelling tasks by reducing the complexity of setting up their model inputs.

There exist various tools to extract information from OSM. However, simply isolating the road infrastructure from OSM data does not suffice in a planning context. To create routable networks suitable for multi-modal transport planning OSM networks need additional effort; they require additional configuration to attach, for example, link capacities, deal with tagging errors, and/or supplement missing information on lanes and speeds, etc. A few of such parsers do exist. However, they are generally designed to produce something specifically suitable to a single modelling environment, e.g., output for Aimsun (Aimsun, n.d.), VISSIM, or MATSim networks (Horni et. al., 2022). These might not be open-source, and if they are, they are often cumbersome to setup, especially for researchers, students, or professionals who do not have a computer science background. Lastly, these parsers are focused on vehicular traffic and rarely allow for flexible extraction of for example pedestrian, bicycle, and/or public transport services/infrastructure information.

In this work we address existing limitations in two ways: (i) Provide flexible open-source tools that are transport model agnostic, with various configurable data format conversions. (ii) Demonstrate the capabilities of these tools via an example application in the context of the ATRC project. The presented case study demonstrates extracting MATSim networks combined with public transport schedules from OSM/GTFS data sources. Note however that the tooling and underlying data format supports pedestrian, bicycle, and other mode extractions as well.

2. Approach

The conversions supported by the presented work are based on the notion of a “converter”. It asks the user to choose a reader, e.g., OSM network reader, as well as a writer, e.g., MATSim

network writer, see Figure 1. Readers and writers have user configurable settings to fine-tune the process without the need for complex programming. Each reader outputs a memory model in the PLANit intermediate format, which remains unseen by the end user. This memory model is then passed on to the chosen writer which, in turn, produces the result in the desired format.

An end user will not notice this 2-tiered approach, but by having an intermediate layer, additional readers/writers will be easier to create and maintain. For example, one can choose to create a writer for their own data format while reusing the existing OSM reader. The intermediate format is transport planning centric, unlike GTFS or OSM, and is therefore much easier to handle.

The tools are available in Java but also have a Python interface. Documentation, examples, and other resources on the project can be found on the PLANit website (PLANit, n.d.). It should be noted that network conversions are just one part of this project, which has a wider objective to provide a variety of open-source tools regarding (supply side) transport planning and simulation. In the remainder of this short paper, we will focus on the capabilities of our OSM and GTFS readers combined with the MATSim network/public transport schedule writer.

3. Open Street Map parser

The objective of the OSM parser is to be generally applicable across a range of transport planning applications, avoiding assumptions on how these networks will be used. Default configurations are provided while allowing users to apply overrides if needed. The parser constructs physical infrastructure networks (when modes are non-stationary, e.g. road, rail) as well as physical public transport infrastructure (when modes are stationary, e.g., poles platforms). Services are not supported, these are parsed via a compatible GTFS parser, see Section 4.

Network parser

Notable *configurable* features for OSM network parsing are listed in Table 1. Some of the listed features allow users to address possible tagging errors that inevitably occur in OSM. Comprehensive logging is provided to identify such cases and users can act in case the default solution is not desired or cannot be derived from the context automatically.

Figure 1: (a) Conversion architecture from source to sink format via intermediate format, (b) various types of converters are available, such as for networks and services (or a combination (not depicted).

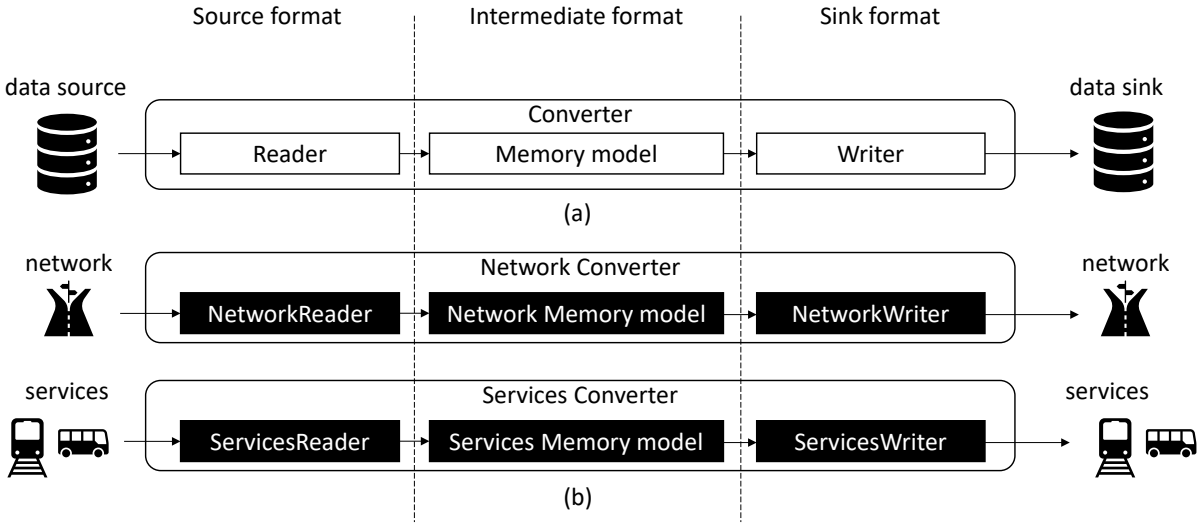


Table 1: configurable features of OSM network parser.

Feature	Notes	Example
Mode mapping	Default mode mapping is provided, but users can override this	One can map light-rail, tram, subway all to a single train mode, or keep them separate
Road mode support identification	Identify which modes are accessible to OSM ways, including bike lanes, bus lanes, and complex directional tagging information	A one-way street with an opposite direction bus lane also allowing bicycles, will result in two uni-directional links with the correct modes attached
Speed limit and lane grouping	Automatic grouping of roads by type based on speed limit, lanes, etc. Defaults applied if absent (by type) which can be overridden	Override default speed limit and/or number of lanes for any OSM way types when not tagged
Removal of unconnected subnetworks	Any dangling subnetworks in the result can be removed and size of what is considered dangling can be configured	For example, remove all dangling network with less than 10/20/100 nodes
Overwrite mode access on specific OSM ways, exclude specific OSM ways	Explicit overwrites for mode access can be provided, OSM ways that are to be removed for scenario testing, or otherwise can be listed	Remove OSM ways that are under construction, or not yet finished, allow buses on private roads where appropriate etc.
Railway and road way type (de)activation overrides	Default configurations for which OSM way types and rail types are parsed, used and logged. These can be overwritten when required	Extract only the miniature railways for children in New south Wales by switching off all rail types except miniature railways (default is that these are excluded)

For example, it might happen that a bus stop resides on a road without bus access, or a bus stop is located on the wrong side of the road, such situations will be logged and can then be addressed by for example adding mode bus to this road or attaching specific stops to specific OSM ways.

There is also support for country specific defaults. Mode access for example is governed by the type of OSM road, e.g., no pedestrians on motorways. This logic is applied when explicit tags are absent. A similar approach exists for speed limits which also have defaults by OSM road types. However, these defaults differ per country and are not part of the OSM data but are disseminated on OSM's wiki pages. The parser has embedded country specific mappings conforming to theses OSM specifications for Australia and The Netherlands, while others can be provided via configuration files. If the OSM data originates from a not yet supported country, global defaults will be applied. This holds for railways as well as for roads.

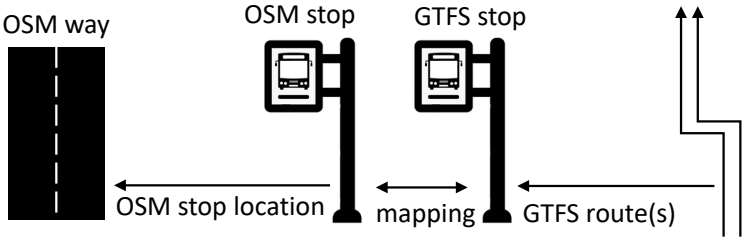
Public transport infrastructure parser

The OSM parser is also capable of parsing public transport stops. The benefit of OSM based stop/platform data over GTFS stops is the fact they can contain information on *how* the stop attaches to the road network via specialist tags. GTFS stop information only provides a location of the pole/platform and requires algorithmic based snapping to the network which is inevitably more error prone. By matching GTFS stops to OSM stops, when present, and then using the OSM mapping to the network, the quality of the mapping can be improved, see Figure 2.

Public transport services, however, are not sourced from OSM because quality and availability is inferior to GTFS. Table 2 lists the main configurable features of the public transport infrastructure parser aspect of the presented approach.

Besides configurable options, the parser supports flexible geometries for stop locations, e.g., poles are parsed as points, platforms as lines, or polygons, and for complex platforms with escalators, we extract the outer geometry. In case the output format does not support this level of detail, e.g., MATSim, these geometries will collapse to points.

Figure 2: Schematic impression of how leveraging OSM stop locations can help snapping of GTFS stops to the physical network.



OSM has various ways of supporting train stations, ranging from a single "station" node, to intricate platform configurations. In the former case, the parser will infer platforms by searching for the closest by track and then places platforms on all tracks running parallel to the reference track within a reasonable distance. As an end user this means that the result always provides a platform per rail track per station, even if the platform is not explicitly coded in OSM. These types of "salvaging", or intelligent parsing options, have been applied throughout to minimise the need for manual adjustments after the fact.

4. GTFS parser

The GTFS parser comprises three main aspects: (i) generic file parsers for the various GTFS files, (ii) conversion from raw GTFS entities to the intermediate format, (iii) integration with existing public transport infrastructure available in the intermediate format, e.g., fuse with a network and infrastructure obtained via the OSM parser.

(i) Generic file parsers

In case a user just wants to parse GTFS files, or a subset, the provided collection of parsers can be used without any need to adopt the PLANit intermediate format at all. They parse raw CSVs and provide in memory versions of the GTFS data in a convenient way. The user can configure which columns to extract or leave out.

Table 2: Configurable features of OSM public transport infrastructure parser.

Feature	Notes	Example
Configurable search radii	Defaults are applied when snapping poles and platforms to OSM ways (roads/tracks), these can be overwritten	When no explicit mapping to a road/rail is provided, the parser will search around the pole/platform – up to the distance threshold – to find the closest mode matching road
Exclude stops, stations, platforms by node or way ids	Allow user to exclude certain stops, platforms, or stations from parsing if they are problematic, outdated, or for other reasons	OSM contains stops that are under construction, in the wrong location, and/or not accessible to the public, on the wrong side of the road etc. These can be excluded (or mapped differently)
Overwrite stop location on road to waiting area (pole/platform) mapping	The stop location is the location on the network where a transit vehicle services a waiting area (pole/platform). Sometimes there is a tagging error in this mapping. Using this override, the user can force the mapping to which waiting area this stop location is matched, without issuing a warning.	A stop location might be located on the wrong side of the road in a complex bus station with many platforms, if upon inspection the mapping between pole and location is valid (but the pole cannot be moved), this can be used to avoid this warning and indicate their relation as valid)
Overwrite waiting area (pole/platform) OSM way (stop location) mapping	Inverse of the above, when no stop location is tagged on a waiting area (pole/platform), the algorithm might identify a problem in identifying a valid stop location on a nearby road, this option allows the user to inspect and force where the parser should place the stop location	This might be needed when the stop is placed too far from the road it services, or it is matched incorrectly to a minor road that is spatially closer (but would trigger a warning with the stop being on the wrong side)

(ii) GTFS parser to intermediate format

Built on top of the generic parser are GTFS readers that process the GTFS data and produce stops and platforms, a service network, and the routed services in the intermediate format. This format preserves the schedule information and contains detailed geometries of the stops and platforms. It also creates an additional service network on top of the physical network representing the legs between stops as a graph, where each leg comprises one or more physical link segments. The GTFS routes and trips then are routed on the service network (rather than the physical network), based on the shortest paths between used stops.

(iii) Mapping of GTFS stops to physical/service network

In case the user has parsed public transport stops and platforms from OSM (or elsewhere), then they are available in the intermediate format when commencing the parsing of the GTFS. In that case, the parser will attempt to map the GTFS stops to existing stops whenever this makes sense (utilising contextual information such as platform names, locations, etc.). If no viable match is available, a new stop/stop location will be created. Hence, in absence of any public transport infrastructure the parser will just create new ones. When the mode at hand is bus, it will create the stops on the correct side of the road depending on the driving rule of the country, while for rail-based modes it is assumed both directions of the tracks will be served by the same stop.

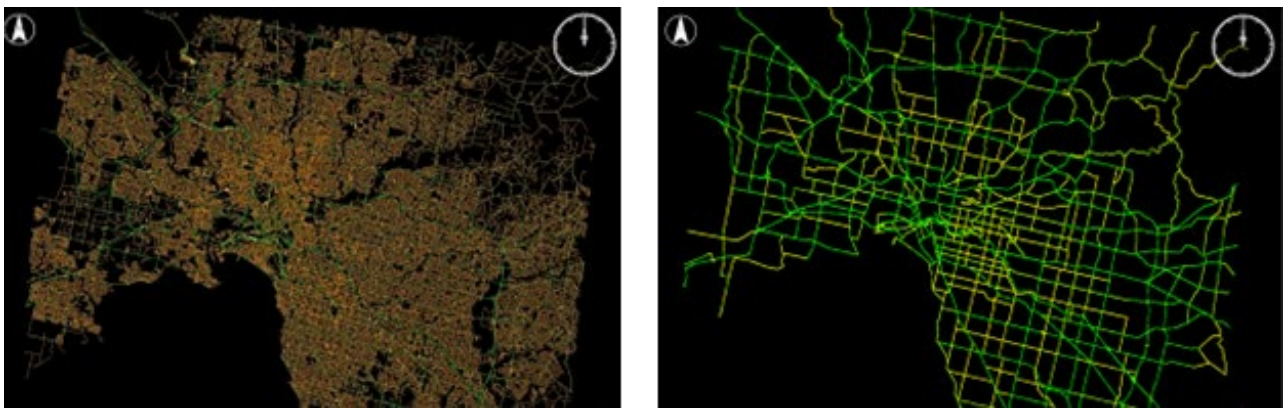
Any GTFS data that is parsed but falls outside the area of the underlying physical network, will be automatically cropped to the portion that overlaps with the chosen underlying network. In case services exit and re-enter the network, they will be split into separate routes such that they are not lost.

4. Case study/ATRC integration

The current version of the OSM/GTFS parser is showcased in the ATRC project, with the particular purpose of converting OSM/GTFS networks/schedules into MATSim networks/schedules with minimal configuration required by the user. It exposes high-level configuration options such as the preferred fidelity of the network, a chosen bounding box (to limit the area to parse), and options such as include/exclude rail and/or public transport infrastructure. Figure 3 provides an impression of a high and lower fidelity example of a MATSim network extracted from OSM using this method.

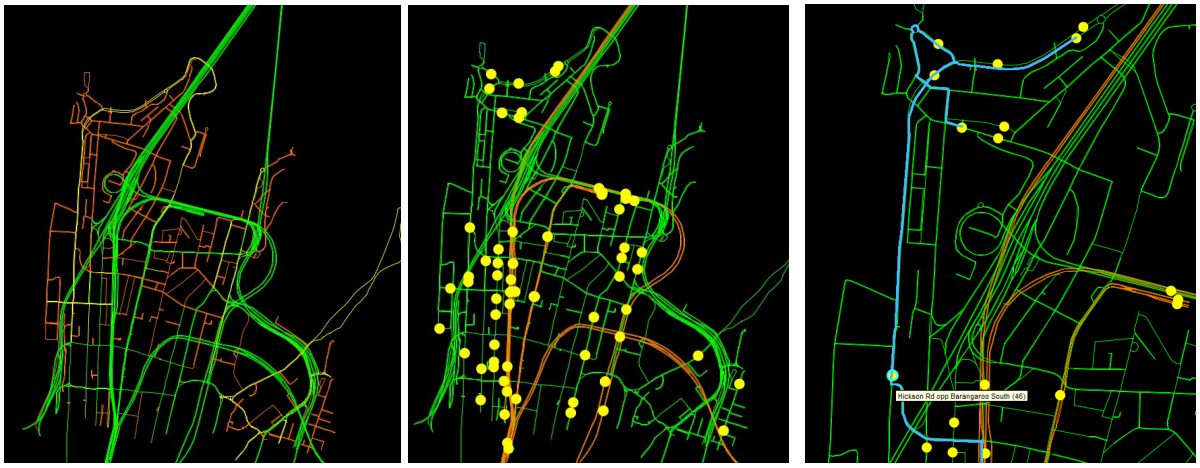
In this context, it is provided as a cloud-ready Docker image which can be run in stand-alone fashion either locally or remotely, i.e., via github actions, or some other cloud-based system.

Figure 3: (a) high fidelity result of parsing OSM network of Melbourne in MATSim format, (b) lower fidelity result of same network conversion.



(a) (b)

Figure 4: (a) Sydney CBD road/rail network in MATSim format, (b) with rail/light-rail and pt infrastructure, (c) same network with GTFS attached.



(a) (b) (c)

When PT infrastructure is enabled, the produced networks will include rail/light-rail/tram/metro infrastructure as well as the waiting areas, i.e., bus poles, train platforms, etc. In case a group of platforms form a station, these will be captured in a group in the intermediate format. An example of the Sydney CBD with stops is provided in Figure 4(b), while Figure 4(c) shows an example result with the GTFS services based on the presented fusing of OSM pt infrastructure with GTFS stop and schedule information.

5. Discussion and future work

In this work we presented a new generic open-source OSM/GTFS parser for transport planning purposes. Unlike existing parsers, it can leverage OSM public transport information to improve the GTFS parsing accuracy. It is not dedicated to a particular data format for its input or output. Based on its 2-tiered conversion approach it allows for easy extensions where only part of a converter, e.g., only a reader, or only a writer, needs to be implemented, after which all existing readers/writers can be reused to create new converter combinations. An example application – part of the ATRC initiative – demonstrated the OSM/GTFS readers and MATSim writer. Additional readers and writers supporting shape files and geopackages are planned to be added in the near future.

Acknowledgments

The Australian Transport Research Cloud project received investment (https://doi.org/10.47486/PL104) from the Australian Research Data Commons (ARDC), and technical support from AURIN. The ARDC and AURIN are funded by the National Collaborative Research Infrastructure Strategy (NCRIS).

References

Aimsun, n.d.. User manual, https://docs.aimsun.com/next/22.0.1/UsersManual/OSMImporter.html

ATRC, n.d., Australian Transport Research Cloud project, https://doi.org/10.47486/PL104

Horni, A, Nagel, K and Axhausen, K W (eds.) 2016 The Multi-Agent Transport Simulation MATSim. London: Ubiquity Press. DOI: http://dx.doi.org/10.5334/baw

PLANit, n.d. Planning Assignment, Networks, Integrated Toolkit open-source initiative, http://www.goplanit.org.