Readme

Make a real Visum network ready for LinTim

Alexander Migl, M.Sc.
Prof. Dr.-Ing. Markus Friedrich

March 17, 2023

University of Stuttgart
Institute for Road and Transport Science
Chair for Transport Planning and Traffic Engineering
www.isv.uni-stuttgart.de/en/vuv/

Make a real Visum network ready for LinTim

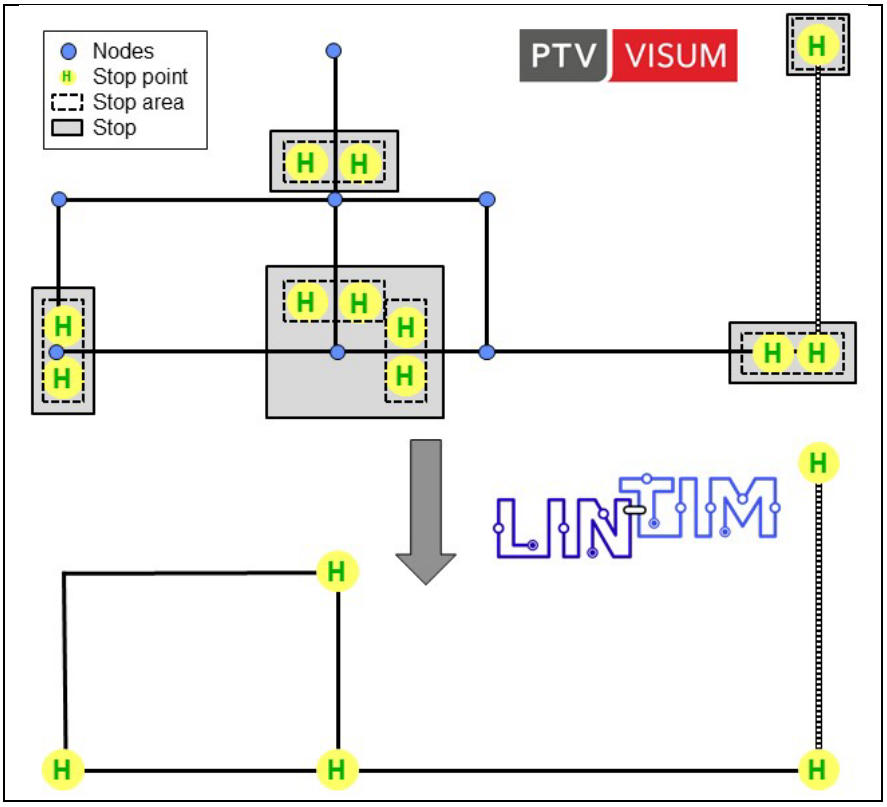

Figure 1: Graphical illustration of the process

Summary

This interface provides two PTV Visum procedure sequences and a master Access Database for converting an infrastructure from a PTV Visum infrastructure network into a LinTim (lintim.net) compatible format (see Figure 1). `VIS20_SelectLines.xml` prepares public transport lines in such a way that a new infrastructure network can be defined on the basis of line routes and their related stops. The Access Database `LineRoutesAsLinks_Master.accdb` is used to build this infrastructure network and is also accessed in that procedure sequence. Afterwards, `VIS20_LoadNetwork.xml` loads the new infrastructure from the Access Database into the PTV Visum instance again and performs a final preparation. Then, LinTim can generate solutions for that instance (see Interface_PTVVisum-LinTim). This Stuttgart public transport network was developed with the interface.

Before using this interface on your own network, we **recommend** to become familiar with the example version file `2030_Stuttgart-Cannstatt.ver` and its processing. The provided data should work for this network. For other networks adjustments might be necessary. They are marked in **purple**.

To run this example, open `2030_Stuttgart-Cannstatt.ver` and start with the procedure sequence `VIS20_SelectLines.xml`. Afterwards run the other procedure sequence `VIS20_LoadNetwork.xml`.

Folder Structure

Due to some dependencies and files generated, we recommend the folder structure as defined here on GitHub. Otherwise you have to adjust the project directories, which are set in the procedure sequences, and some internal python scripts.

Requirements

The procedure sequences are developed using *PTV Visum 20* and *Python 3.7*. Queries are based on *Microsoft Access 2019*. In addition, the following python libraries, which should be included in your Visum installation, are used:

- `Tkinter`
- `numpy`

If your PTV Visum version file uses different transport system codes for public transport than defined in the example PTV Visum version file `2030_Stuttgart-Cannstatt.ver` (see Figure 2), you have to **adapt** the queries *310* and *370 to 375* in the Access Database `LineRoutesAsLinks_Master.accdb` according to your codes. In your Visum instance, the procedure sequences can deal with different codes.

Transport systems | Modes | Demand segments

Number: 9	Code	Name	Type
1	B	Bus	PuT
2	F	Fussweg	PuTWalk
3	I	Fernbahn	PuT
4	Mex	Mex	PuT
5	NVBW	NVBW	PuT
6	R	Regionalbahn	PuT
7	S	S-Bahn	PuT
8	U	Stadtbahn	PuT

Figure 2: Transport system codes in `2030_Stuttgart-Cannstatt.ver`

Procedure sequence `VIS20_SelectLines.xml`

In PTV Visum a line can consist of more than two line routes. LinTim cannot handle that. The procedure sequence `VIS20_SelectLines.xml` tries to define exactly two line routes for each line, that should be transformed. Manual adjustment might be necessary.

The procedure sequence contains several groups (see Figure 3). The first two are just for preparation. More details are provided in the next subsections.

Number: 44	Procedure	Reference object(s)
1	Group Set Project directory	2
3	Group Pre-Settings	4 - 7
8	Group Select TSys	9
10	Group Select Lines	11 - 16
17	Group Aggregate LineRoutes	18 - 24
25	Group Select LineRoutes	26 - 33
34	Group Check Selection	35 - 38
39	Group SaveToDatabase	40 - 42
43	Group Processing in Database	44

Figure 3: Overview of procedure sequence `VIS20_SelectLines.xml`

Select TSys

A *Tkinter*-checkbutton-box pops up. Please select the transport systems you want to consider for creating the new infrastructure network. By default, `'B'=bus`, `'S'=S-Bahn` and `'U'=U-Bahn` are selected. If selected, the transport systems attribute `LINTIMEXPORT` becomes `true`.

Select Lines

This group defines, which lines are considered (line attribute `LINTIMEXPORT`) for creating the new infrastructure network for LinTim based on the selected transport systems. Default values take only lines with at least 4 departures between 7 AM and 9 AM. Other settings are conceivable. *Steps 14 and 15* **allow** manual adjustments by adapting the line attribute `LINTIMEXPORT`.

Aggregate LineRoutes

In order to consider lines in LinTim, a line may consist of a maximum of two line routes. Therefore, the existing line routes are aggregated in this group. This has to be done twice, since otherwise the amount of time profiles might be too high. Afterwards, travel times can be rounded. *Step 19* **allows** manual adjustments regarding filter criteria.

Select LineRoutes

In some cases, not all line routes can be aggregated, since the spatial course differs too much. Then, single line routes have to be eliminated. This group offers an automatic selection, which routes are kept, depending on the highest number of departures between 7 AM and 9 AM (*steps 24 to 26 and 28 to 30*).

Check Selection

In this group the selected line routes are verified. If more than one line route per direction is selected, an error message (see Figure 4) pops up. Then, **manual adjustments** should be considered. If everything seems to be fine, nothing happens.

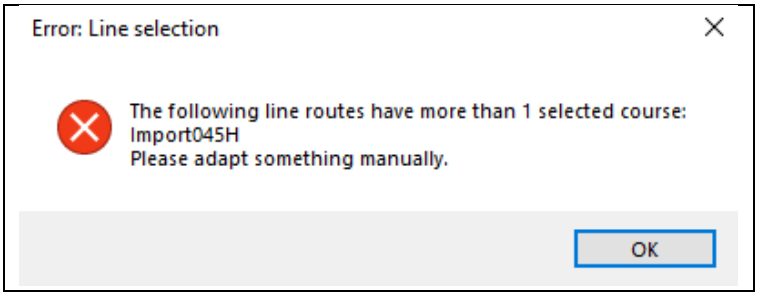

Figure 4: Potential error message

Export to Database

Manual adaption of stops: It might be useful, to not consider all stops from the original infrastructure network (one way streets, stops are only served in one direction, different stops (not stop points) for different transport systems etc.). A recoding table is defined in `LineRoutesAsLinks_Master.accdb` (see Figure 5). The stops, which should be merged, should be defined in the Access-Table `STOP_Recoding`. E.g. in the example version file `2030_Stuttgart-Cannstatt.ver` the stop “Veielbrunnenweg” of line route “45_R” is called “Elwertstraße” in the opposite direction “45_H”. Since “Veielbrunnenweg” is served also from Line “Import056” in both directions, it is useful, to merge “Elwertstraße” to “Veielbrunnenweg”. Furthermore, the stops “Bad Cannstatt”, “Bad Cannstatt Wilhelmsplatz” and “Badstraße (Wilhelmsplatz)” might be merged, to enable better transfers.

To merge stops, **define** the stops and the direction of merging in the Access-Table `STOP_Recoding`. The stop, which should be replaced, is defined in the columns `NO_Original` and `Name_Original`. The stop, that is used for the replacement, should be defined in the columns `NO` and `NAME`.

STOP_Recoding

NO	NAME	NO_Original	Name_Original
2332	Veielbrunnenweg	2358	Elwertstraße
6333	Bad Cannstatt	6332	Bad Cannstatt Wilhelmsplatz
6333	Bad Cannstatt	6334	Badstraße (Wilhelmsplatz)

Figure 5: Access-Table `STOP_Recoding` in `LineRoutesAsLinks_Master.accdb`

In this group, the procedure sequence exports the relevant data to a new access data base `LineRoutesAsLinks_EditedLinks.accdb`, which is based on the master data base `LineRoutesAsLinks_Master.accdb`.

Afterwards, most network objects in the Visum instance are removed.

Processing in Database

The new access data base `LineRoutesAsLinks_EditedLinks.accdb` contains tables for all relevant information for the transformation into a LinTim compatible version file:

- `Stops`
- `Nodes`
- `StopAreas`
- `StopPoints`
- `Links`
- `Lines`
- `LineRoutes`
- `LineRouteItems`
- `TimeProfiles`
- `TimeProfileItems`
- `VehJourneys`
- `VehJourneyItems`
- `VehJourneySections`

There are several queries, which lead to the transformed infrastructure. All these queries are called by the macro `00_Run_All_Queries`. That macro is called automatically in *step 43*. While you call it, the database has to be closed.

Procedure sequence `VIS20_LoadNetwork.xml`

After the processing with Microsoft Access, please use the other procedure sequence `VIS20_LoadNetwork.xml`. It is used for loading the new network into the PTV Visum instance and for doing some final preparation.

This procedure sequence contains several groups too (see Figure 6). More details are provided in the next subsections.

Number: 122	Procedure		Reference object(s)
1	Group Load Infrastructure ...	...	2 - 4
5	Group PreSettings ...	...	6 - 7
8	Group Symmetrize ...	...	9 - 17
18	Group CreateConnectors ...	...	19 - 31
32	Group Change Stop-TypeNo ...	...	33 - 35
36	Group Turns symmetrical ...	...	37 - 38
39	Group Load Lines ...	...	40
41	Group Delete empty nodes/stop	...	42 - 47
48	Group Set Stop TSys ...	...	49 - 50
51	Group Simplify TimeProfiles ...	...	52 - 62
63	Group Create periodic timeprofil	...	64 - 76
77	Group Demand Data ...	...	78 - 79
80	Group Set Walking ...	...	81 - 83
84	Group Edit cordon-connectors ..	...	85 - 90
91	Group Change VehUnits ...	...	92 - 95
96	Group Lines for optimizing comp	...	
97	Group Simplify TimeProfiles ...	...	98 - 106
107	Group Create periodic timeprofil	...	108 - 120
121	Group Save version ...	...	122

Figure 6: Overview of procedure sequence `VIS20_LoadNetwork.xml`

Load Infrastructure

First of all, the new infrastructure (without lines) is loaded to the Visum instance from `LineRoutesAsLinks_EditedLinks.accdb`.

PreSettings

Since the previous connectors cannot be used anymore, they have to be created from scratch. For zones in a selected area (in `2030_Stuttgart-Cannstatt.ver` the main zone *"Stuttgart - Bad Cannstatt (Ost)"*), it is recommended to generate them automatically. For cordon zones (in `2030_Stuttgart-Cannstatt.ver` zones with *TypeNo="9"*) a manual assignment seems to work better.

For the **manual assignment**, it is necessary, to define the node the zone should be connected with in the zone-attribute `name`. In `2030_Stuttgart-Cannstatt.ver` the assignment is already set for the cordon zones.

Symmetrize

LinTim requires, that the transport systems and the travel times on a link are the same on the reverse link. To ensure this, the values are symmetrized.

If required values are missing, e.g. since additional links were added manually, these values are set based on the poly-length.

CreateConnectors

If not all zones shall be considered, there is an **option** for deletion here.

The group is adapted for `2030_Stuttgart-Cannstatt.ver`. In your network, the numbers and type-numbers of zones are probably different. Therefore, you should make **manual** adjustments.

Connectors should only refer to a node with an outgoing link. Therefore, a filter is implemented here. As mentioned above, the connectors inside a specific area can be generated automatically (*step 22 and 23*). Manual connector assignments take place in *steps 26 to 28*. The manual allocation is defined in the zone-attribute `name` (in `2030_Stuttgart-Cannstatt.ver` already set). Another option could be the deletion of some cordon zones.

Change Stop-TypeNo

Classification of the stop types depending on the outgoing links.

Turns symmetrical

The turns have to be symmetrical.

Load Lines

After the preparation of the infrastructure is completed, the lines and the timetables are loaded from `LineRoutesAsLinks_EditedLinks.accdb`.

Delete empty nodes/stops

Nodes and stops, which have been merged for the link-network, are still in the infrastructure network. Here they are deleted.

Set Stop TSys

The transport systems of the stop points are defined here, according to the line routes serving the stop point.

Simplify TimeProfiles

LinTim can only optimize the supply of one transport system. With a *Tkinter*-radiobutton-box you have to specify, which transport system should be optimized later (in `2030_Stuttgart-Cannstatt.ver` by default `'B'=bus`). Lines of other transport systems are specified as fixed.

For considering a line as fixed in LinTim, the line has to serve every stop along its line route. This is achieved in the *steps 56 to 62*.

Create periodic timeprofile for fixed lines

Moreover, a fixed line in LinTim needs a periodic timetable. This group generates a periodic timeprofile for each fixed line by determining the first departure in the original timetable (Attribute “AddVal1”) and by determining a headway (Attribute “AddVal2”) from the number of

departures between a specific time interval (in `2030_Stuttgart-Cannstatt.ver` 7 AM till 9 AM).

You might also create a periodic timetable for the non-fixed-lines, to allow a better comparison between the original solution and solutions from LinTim. Two groups with a similar structure to the fixed-lines-processing are defined in the end of this procedure sequence (by default not active).

Demand Data

Names and codes of transport systems can be changed, added or deleted here. Especially the code for the transport system type “PuTWalk” is renamed to *“Walk”.*

Set Walking

Since walking can be allowed in the network, the transport system “Walk” is added to the Visum model. Travel times for walking are calculated by default with a walking speed of 4 km/h. Moreover, transfer walk times between stop areas are set (default value: 180 seconds).

Edit cordon-connectors

It might make sense, to set the length of the cordon-zone-connectors to 0 km. If so, the travel times should be 0 too.

Change VehUnits

LinTim can only consider one vehicle type per transport system. Here, for each transport system in the network one vehicle unit and one vehicle combination is defined. The exact values can be changed via the Tkinter-surface. Old units and combinations are removed beforehand.

Lines for comparison with LinTim

see *Create periodic timeprofile for fixed lines*

Save version

The Visum-version-file is saved.

Now the network should be ready for generating PuT-supplies with LinTim, if a demand matrix for public transport is defined. For the Visum-to-LinTim-interface, check Github.