Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2017) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

The 8th International Conference on Ambient Systems, Networks and Technologies (ANT 2017)

GTFS Bus Stop Mapping to the OSM Network

Jan Vuurstaek[a,*], Glenn Cich[a], Luk Knapen[a], Ansar-Ul-Haque Yasar[a], Tom Bellemans[a], Davy Janssens[a]

[a]*Hasselt University, Transportation Research Institute (IMOB), Agoralaan, 3590 Diepenbeek, Belgium*

Abstract

Due to budget constraints *public transportation* (PT) can no longer be deployed in regions where it attracts insufficient customers. Novel techniques like *demand-responsive collective transportation* (DRT) are evaluated to cut costs. This requires detailed simulations that are able to predict travel demand and include trip execution. Simulating facilities acting as feeder services to time-table based PT services requires detailed and accurate information about the PT infrastructure on a network. However, there are no public data sources that combine network and PT infrastructure data with the preferred level of detail.

A newly developed bus stop mapping technique is presented. It uses the *OpenStreetMap* (OSM) and *General Transit Feed Specification* (GTFS) open data sources, which are maintained independently. Merging the data into a single database requires alignment. Developing bus stop mapping algorithms is challenging due to (i) inaccurate location data, (ii) inconsistent data sources and (iii) the vastly interconnected PT network and services. Due to the inaccuracy in the GTFS stop locations and in the OSM network, pure geometric considerations might lead to multiple candidate solutions to map a stop to the network. The new technique handles all GTFS trips *at once* and operates under the assumption that PT operators minimize the total distance driven to complete all trips.

Peer-review under responsibility of the Conference Program Chairs.

Keywords: OpenStreetMap (OSM), General Transit Feed Specification (GTFS), Micro-Simulation, Public Transport, Mapping Algorithm

1. Introduction

In Belgium a *bus stop* is a pole on the ground, at a specific side of the road, where people can board or alight a transit vehicle. These bus stops are serviced by a Public Transportation (PT) provider, which implies that if a road is serviced in both directions, each side of the road gets its own bus stop. The research presented in this paper attempts to match the bus stops of the only bus PT provider in Flanders (Belgium), called "De Lijn", to a road network. The bus stops are extracted from a dataset that is made publicly available by "De Lijn". This dataset follows the General Transit Feed Specification (GTFS)[1]. As for the road network, the publicly available OpenStreetMap (OSM)[2]data source was used.

* Corresponding author. Tel.: +32-11-269-111 ; fax: +32-11-269-199.
E-mail address: jan.vuurstaek@uhasselt.be

Peer-review under responsibility of the Conference Program Chairs.

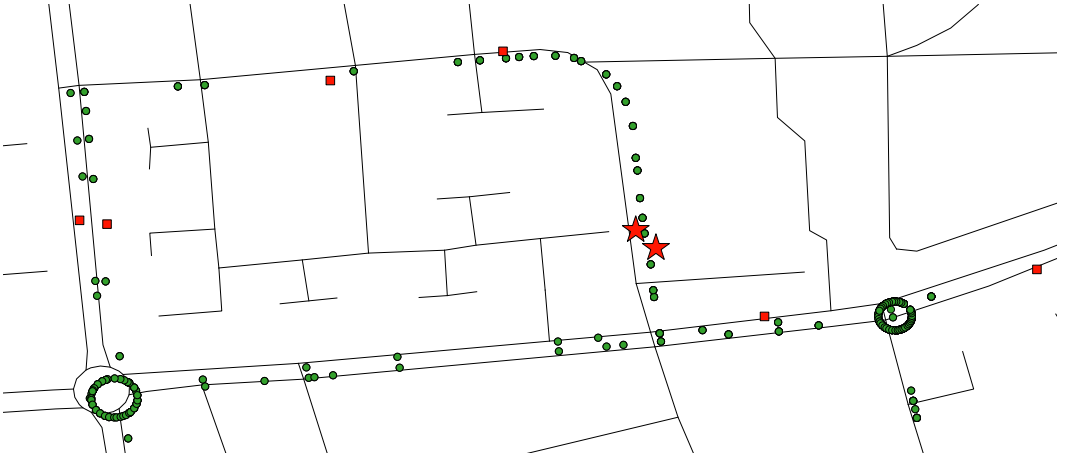


Fig. 1: Israel, Kiryat Gat: example of inaccuracy. Shapes (green circles) and bus stops (red squares/stars) in GTFS displayed on an OSM network. The stops represented by the red stars show the *wrong side of the road problem*. Note the missing roundabout at the right side and the positional errors.

The OSM database is fairly complete but the amount and size of positional errors cannot be ignored, as stated in the research of Haklay et al. [1] The error on GPS recordings is in the range of [15, 30] meters (example given in Figure 1). Each bus stop in GTFS is represented by exactly one coordinate pair which is assumed to have been determined by a GPS recording. As a consequence it is not always clear to which road segment a bus stop needs to be assigned. Assigning a bus stop to the wrong side of a road can induce large distance and travel time errors in reconstructed bus routes. Such errors are not acceptable in simulations of demand-responsive (private and public) collective transportation (DRT). Mutual coordination (cooperation), low volumes and hence small capacity vehicles characterize those situations. Reasoning about average flows is unfeasible due to large quantisation effects. Solutions to such problems require combinatorial optimization and are very sensitive to small changes (errors) in the input.

DRT services are used as feeders to PT services. In this context, accurate assignment of bus stops to the road network is required for modeling.

Many published GTFS datasets do not contain the optional shape file.[3] Furthermore, the shape file only provides information on the geometry but not the topology. The `shapes.txt` file in GTFS contains tuples $\langle s, x, y, o \rangle$ where s denotes the sequence identifier, x and y denote longitude and latitude respectively and o is the offset of the point in the sequence. It is observed in the Israeli data that exactly the same $\langle x, y \rangle$ pair can occur in multiple sequences. It is easy to find pairs of locations p_0 and p_1 that do occur consecutively and in both orders in multiple sequences. This means that sequences of $\langle x, y \rangle$ locations were recorded once and then used in both orders to represent trips driven in opposite directions. Hence, the shape files cannot be used to determine the bus stop locations from geometry and ordering.

Several research efforts to assign GTFS bus stops to OSM and other road networks have been reported in literature. A first category makes use of pure geometric methods which can lead to bus stops matched to the wrong side of the road due to positional errors. A second category starts from the assumption that the bus travels the minimum distance required to serve consecutive stops; those methods process a single bus trip at a time. For bus stops used in multiple trips, inconsistent results are reported by independently handling individual trips (routes). This paper describes a newly developed technique to handle all GTFS trips *at once*.

The paper is organized as follows. Section 2 provides a discussion of related work. Next, the data preparation is described in Section 3, followed by the bus stop mapping algorithm in Section 4. Then, in Section 5, a case study for Flanders (Belgium) is presented. Section 6 concludes the paper and discusses future work.

[1] `https://www.openstreetmap.org/`

[2] `https://developers.google.com/transit/gtfs/`

[3] Following GTFS data sources where checked: De Lijn (Belgium), RATP (France), Fritz Behrendt OHG (Germany), Günter Anger Güterverkehrs GmbH & Co. Omnibusvermietung KG (Germany), Haru Reisen OHG (Germany). These data sources are available on `http://www.gtfs-data-exchange.com/`.

2. Related Work

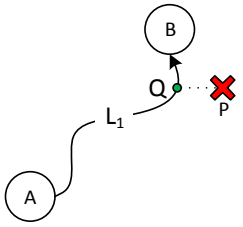
A handful solutions have been provided in the past. Lektauers et al.[2] mention a "*tool to associate stop facilities to network nodes*" without any details (in the context of an integrated MATSim-based simulation). Li[3] exploits the relational information between adjacent bus stops. Several possible projections (candidates) of GTFS stops on the road network are considered by calculating the distance to the nodes of nearby road links, instead of the links themselves. This could lead to mismatches when the correct location of a bus stop is on a long straight link that is parallel with a nearby short link. The shortest path linking all stops for a trip in the specified order is considered to be the real one. Their model only considers *single* bus trips. In case of overlapping bus trips, manual checking is required. Perrine et al.[4] propose a two step method making use of the optional shape file. In the first step the shape points are map-matched in order to determine a link sequence for each track using a multi-hypothesis technique (MHT). This is done by considering possible successive extensions of the link sequence and retaining the solution that results in the lowest distance driven. While mapping the GTFS shape points, the curvature of the road segments is not considered, which may lead to mismatches. In the second step, the links identified by the map matcher are retained as a new network. Then the sequence of bus stops is used as a virtual GPS track and map-matched against the reduced network. This is done independently on a *per route basis* so that a particular GTFS bus stop used in several routes can lead to different results. Finally, bus stops located near crossings are reported to be matched against the wrong link. The issues related to the correct road side selection and overlapping bus trips are not mentioned. Ordóñez et al.[5] developed a semi-automatic procedure to combine public bus routes information with a high resolution network as an extension to MATSim. First, a simple map-matching algorithm is applied on a per route basis. This algorithm makes use of the GPS points that are present in the optional shape file. It does not assume that the nearest link to a stop is always the correct one. Second, an automatic verification procedure including following checks is performed: (i) is the path joined, (ii) is the path without U-turns, (iii) is the path without repeated links, (iv) does every stop of the route have a stop-link relationship, (v) does every link related to a stop belong to the path, (vi) does the related links order in the path comply with the corresponding stops order in the route profile, (vii) is the nearest point between the stop point and the infinite line defined by the link inside its line segment for every stop and (viii) are the first and last links of the path related to the first and last stops of the route profile? Steps (ii), (iii) and (vii) can be disabled. If the automatic verification fails, manual editing is required. The authors report that if routes are not processed in an appropriate order, some re-work might be necessary. It required ten days to process the public transport of Singapore.

3. Data Preparation

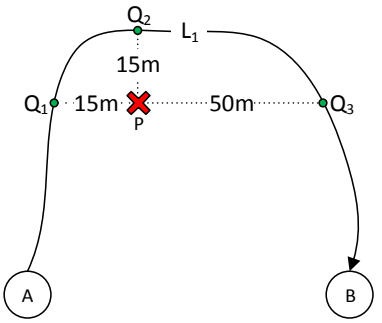
Data preparation from the publicly available OSM and GTFS sources is described by Cich et al.[6] The following procedures were developed (i) build a network derived from OSM suitable for simulations in transportation where each driving direction is represented by a separate link (directed graph), (ii) extract bus stops from GTFS while removing anomalies (details can be found in Cich et al.[6]) and (iii) find for each bus stop specified in GTFS a set of candidate network links to attach it. Note that in the road network road curvature defining shape (geometry) nodes are also included. This ensures accurate distance calculations. The actual attachment of the candidate GTFS bus stops to the OSM road network makes use of the following concepts:

1. **Projection of a point on a curve**: A point Q on the curve C_L representing a network link (road segment) L is a (geometric) projection of a point P if and only if the euclidean distance $d(P, Q)$ is minimal (Figure 2a). Note (i) that the projection can coincide with one of the end nodes of L (Figure 2c), (ii) that a point can have multiple projections on a single link (Figure 2b) and (iii) that $\overline{PQ}$ is not necessarily perpendicular to C_L (Figure 2c).

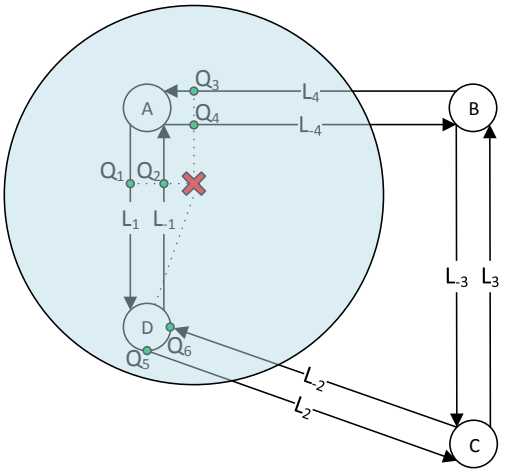
2. **Projection of a point on network**: Each GTFS bus stop P is projected to the set of all network links $L \in \mathcal{L}$ for which the distance between the GTFS stop and its projection $d(P, Q(P, L))$ does not exceed a given threshold. A particular GTFS stop can have a projection on multiple links. Remember that all links are unidirectional. The projections of a single point P on multiple links can geometrically coincide; this occurs when the respective projections all map to a node shared by those links. (Figure 2c)



(a) A projection of a point on a (curve) link.



(b) Point P has three projections on link L_1. Projection Q_3 is discarded. Since both projections Q_1 and Q_2 are at the minimum distance to P, one of them is randomly selected.



(c) A projection of a point on a network. The threshold is indicated by the blue circle. The projection of point P on link L_2 and L_{-2} coincides with link node D.

Fig. 2: Examples of projections of a point.

Multiple projections of a point P to the geometry of a particular link L can exist (an example is shown in Figure 2b). Both the algorithm described in this paper and the simulations into which the bus stops are fed are indifferent to which of the alternative projections of P to L is chosen. Only the projections at the minimum distance to P are considered because those are assumed to have the highest probability to be the ones searched for. One of them is chosen randomly.

A projection is a candidate to represent the GTFS bus stop from which it was derived (hence the name *projected stop* used in the remainder of the paper). Since a projection is associated with a unidirectional link (road segment) the next node in the network reached by a bus serving the projected stop is unambiguously known. Hence, it determines the side of the street to which the candidate applies.

4. Bus Stop Mapping

4.1. Principle of Operation

A technique to handle all GTFS trips *at once* has been developed. Let S_G denote the stops found in GTFS and let S_P denote the set of projected stops. A GTFS stop $g \in S_G$ can result in several projections $p \in S_P(g)$ located on multiple links. For each $g \in S_G$, exactly one $p \in S_P(g)$ has to be assigned to g.

The assignment is computed by optimization. It is assumed that the PT operator minimizes the total distance driven to complete all trips. Hence, the number of times a particular trip is serviced during a day is used as a multiplicative weight factor.

Each $p \in S_P(g)$ constitutes a *degree of freedom* (DOF) for g, i.e. the degree of freedom for g equals $|S_P(g)|$. The number of possible injections $S_G \Rightarrow S_P : g \mapsto p$ is huge (see Section 5). Assigning a projection $p \in S_P(g)$ to a GTFS stop g is called *fixing* g.

Let $G(V, E)$ denote the bus stop connection graph defined by the GTFS trips. Then $V = S_G$ and two vertices are connected by an edge if and only if they appear as consecutively serviced stops in GTFS. G is built by adding edges one after another. Finding an optimal assignment is computationally efficient for an acyclic digraph. Hence, the algorithm decomposes G into mutually independent acyclic components. Whenever the addition of an edge introduces one or more cycles, the set C of vertices that *individually* can break all newly induced cycles is determined. Exactly one of those is to be chosen as a cycleBreaker: $v \in C$ is chosen so that $|S_P(v)|$ is minimal (see Figure 5 for an example). In several stages of the algorithm cycleBreakers can become redundant (in which case they are removed).

The basic idea of the solution is to fix stops in mutually independent bounded acyclic subgraphs of G. Such subgraphs are isolated by boundary vertices. A vertex is a *boundary vertex* only if it is (i) a *source*, (ii) a *sink*, (iii) a previously *fixed* stop or (iv) a *cycleBreaker*. All possible assignments for the boundary vertices need to be examined; hence it is important to select the boundary $B \subseteq S_G$ so that $\prod_{b \in B} DOF(b)$ is small. For each assignment $\mathcal{A}_B$ to the boundary set B, the lowest cost assignment $\mathcal{A}_I$ for the set of *internal* vertices I in the acyclic component is computed. If a particular $p \in S_P(g)$ occurs in each assignment $a \in \mathcal{A}_I$, then g can be fixed to p. On the other hand, if p is not used in any $a \in \mathcal{A}_I$ then p can be discarded as a candidate for the final assignment. Smart selection of the bounded subgraphs results in an efficient procedure. E.g. if the subgraph is a linear sequence of GTFS stops, B contains the two end nodes g_0 and g_1. The shortest paths between each pair $\langle p_0, p_1 \rangle \in S_P(g_0) \times S_P(g_1)$ represents an assignment $\mathcal{A}_B$ and can be computed very efficiently because the corresponding stop projections constitute a layered graph.

4.2. Algorithm

The bus stop mapping procedure is summarized in algorithm 4.1. The concepts presented in the algorithm will be explained using the interconnected trips that are shown in Figure 3.

Algorithm 4.1 Determination of optimal assignment of projected stops S_P to GTFS stops S_G.

```
1:  S_P ← projStops(S_G)
2:  fixTrivial(S_G)                                   ▷ Assign GTFS stops having a single projection
3:  ⟨G_G, G_P⟩ ← graphFromBusStopSequences()           ▷ Introduces cycleBreakers
4:  repeat
5:      removedAtLeastOneCandidate ← false
6:      handleTriples(⟨G_G, G_P⟩)                     ▷ Vertex in the middle has inDegree = outDegree = 1
7:      handleNonBifurcatingMaximalSequences(⟨G_G, G_P⟩)   ▷ Internal vertices have inDegree = outDegree = 1
8:      handleStars(⟨G_G, G_P⟩)
9:      reduceCycleBreakers(⟨G_G, G_P⟩)   ▷ Removes cycleBreakers that became redundant by fixing some vertices
10: until ¬removedAtLeastOneCandidate     ▷ Either by explicit discarding or as a consequence of fixing
11: components ← decompose(⟨G_G, G_P⟩)
12: for all c ∈ components do
13:     assign(c)                                     ▷ Assignment by solution enumeration
14: end for
```

Line 3 builds a graph G_G using GTFS stops based on the sequences found in GTFS. Furthermore, it builds a graph G_P using projected stops so that each $p \in S_P(g)$ for a given $g \in S_G$ is connected to all the projected stops of the neighbours of g. Hence, each pair $\langle g_0, g_1 \rangle \in S_G \times S_G$ of neighbours defines a distance matrix (between the respective stop projections). This is graphically represented in Figure 4 where $P(g_C)$ and $P(g_D)$ make up the distance matrix induced by GTFS stops g_C and g_D. This step includes the marking of some GTFS stops as *cycle breakers*. Figure 5 shows an example of cycle breaking. Subgraphs delimited by cycle breakers are directed acyclic graphs (DAG). Cycle breakers contribute to the DOF size of the irreducible components generated in line 11 and therefore need to be chosen carefully.

Line 6 considers *triples* $\langle g_0, g_1, g_2 \rangle$ for which the set of neighbours of g_1 is $\{g_0, g_2\} = B$ (the boundary of the subgraph). Determine all shortest paths between $p_{0,i} \in S_P(g_0)$ and $p_{2,j} \in S_P(g_2)$ for all i and all j. Drop the projected stops $p_{1,k}$ for all k that are *not* part in *any* shortest path. If and only if for a given k the projected stop $p_{1,k}$ appears in *all* shortest paths, assign it to g_1. Then g_1 is said to be fixed. Figure 6 shows an example of a triple.

Line 7 performs a similar technique on *maximal non bifurcating sequences*. Actually, (handleTriples) is a special case of this one that is implemented to increase efficiency. Figure 4 shows an example of a maximal non bifurcating sequence.

Line 8 again performs a similar technique on *stars* (i.e. a GTFS stop having an inDegree and/or outDegree larger than one). Let $S_P(c)$ denote the projected stops for the *core* of the star. If $p \in S_P(c)$ is used in each possible assignment for the neighbours, c is fixed to p. If p is not used in any assignment, it is discarded as a candidate. Figure 8 shows an example of a star.

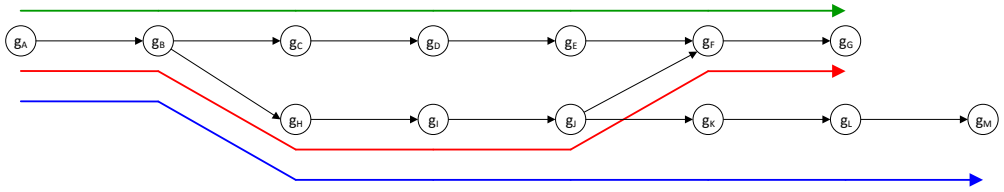


Fig. 3: A collection of three (red, green, blue) interconnected bus trips. Each vertex represents a bus stop on a trip.

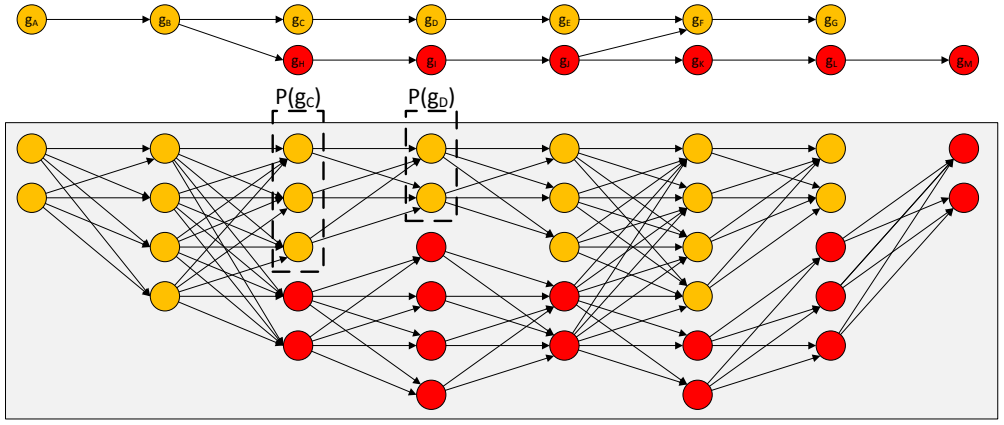


Fig. 4: GTFS stops (upper) and associated stop projections (lower) graphs. The orange/red *string* in the upper part corresponds to the orange/red *layered graph* in the lower part. Each pair of consecutive vertices $\langle g_0, g_1 \rangle \in S_G \times S_G$ corresponds to a complete bipartite subgraph in the lower part.

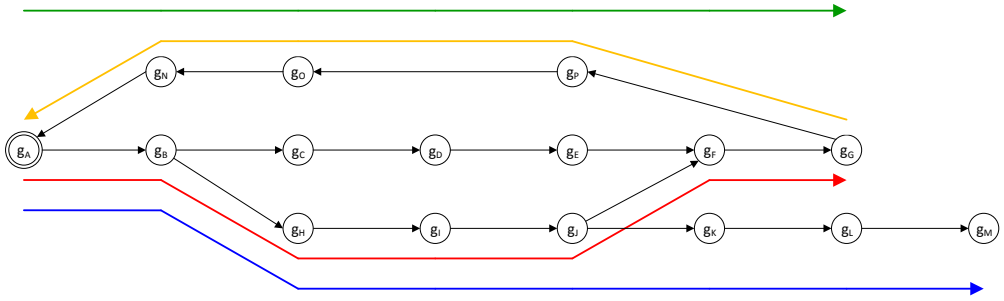


Fig. 5: Example of a cycle breaker. Cycle breaking makes a graph acyclic. Cycle breakers are indicated by double circle vertices. In this example GTFS stop g_A is chosen as a cycle breaker. Other candidates to break the same cycle are: g_B, g_F, g_G, g_N, g_O and g_P.

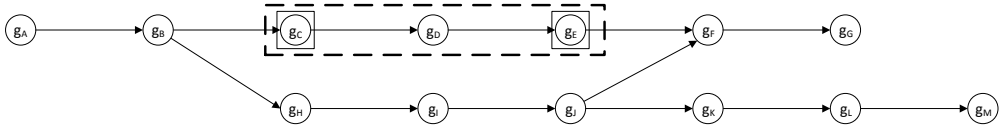


Fig. 6: Example of a triple. Each vertex that has only one incoming and one outgoing edge can be considered as the center of a triple. The vertex at the incoming and outgoing edge are the boundary vertices of the triple. Together they constitute a subgraph. The figure depicts only one triple. Boundary vertices are indicated by a square around the vertex. The subgraph is indicated by a dashed box. All triples present in this figure are: $\{\langle g_B, g_C, g_D \rangle, \langle g_C, g_D, g_E \rangle, \langle g_D, g_E, g_F \rangle, \langle g_B, g_H, g_I \rangle, \langle g_H, g_I, g_J \rangle, \langle g_J, g_K, g_L \rangle, \langle g_K, g_L, g_M \rangle\}$.

Line 9 removes *cycleBreaker* markings that became redundant by stop fixations. If an assignment was found for GTFS stop g_B in Figure 5, a cycle breaker is no longer needed, since g_B then acts as a boundary vertex.

Line 11 decomposes the graph so that each component is maximal and delimited by GTFS stops that are sources, sinks, assigned (fixed) or cycleBreakers. Figure 9 shows an example of a component.

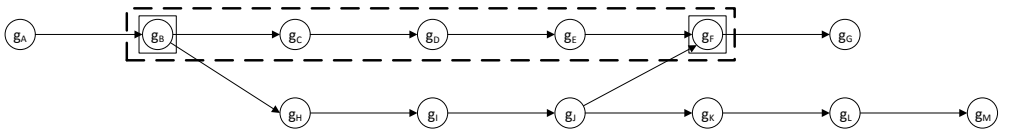


Fig. 7: Example of a maximal non bifurcating sequence. A maximal non bifurcating sequence is a sequence that is delimited by sources, sinks, cycle breakers, assigned vertices or vertices that have more than one incoming or more than one outgoing edge. The intermediate vertices of the sequence have exactly one incoming and one outgoing edge. Together they constitute a subgraph. The figure depicts only one maximal non bifurcating sequence. Boundary vertices are indicated by a square around the vertex. The subgraph is indicated by a dashed box. All maximal non bifurcating sequences present in this figure are: $\{\langle g_A, g_B\rangle, \langle g_B, g_C, g_D, g_E, g_F\rangle, \langle g_F, g_G\rangle, \langle g_B, g_H, g_I, g_J\rangle, \langle g_J, g_K, g_L, g_M\rangle\}$

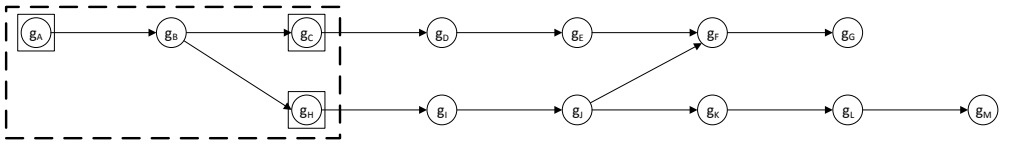


Fig. 8: Example of a star. Each vertex that has more than one incoming and/or more than one outgoing edge can be considered as the center of a star. The vertices at the incoming and outgoing edges are the boundary vertices of the star. Together they constitute a subgraph. The figure depicts only one star. Boundary vertices are indicated by a square around the vertex. The subgraph is indicated by a dashed box. All stars present in this figure are: $\{\langle g_A, g_B, g_C, g_H\rangle, \langle g_E, g_J, g_F, g_G\rangle, \langle g_I, g_J, g_F, g_K\rangle\}$.

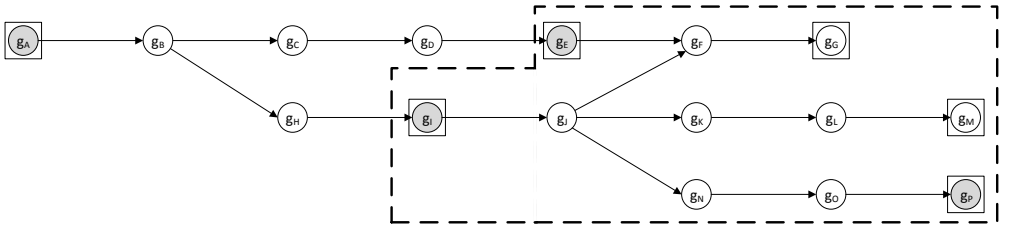


Fig. 9: Example of a component. A component is the largest possible subgraph that can be extracted without including any assigned vertices or cycle breakers as intermediate vertices. The sources, sinks, cycle breakers and assigned vertices make up the boundary vertices of the component. The figure depicts only one component. Boundary vertices are indicated by a square around the vertex. Assigned vertices are indicated by a grey colour fill. The subgraph is indicated by a dashed box. All components present in this figure are: $\{\langle g_A, g_B, g_C, g_D, g_E, g_H, g_I\rangle, \langle g_E, g_F, g_G, g_I, g_J, g_K, g_L, g_M, g_N, g_O, g_P\rangle\}$.

5. Results and Discussion

After assigning a stop projection to each GTFS stop, the bus trips were reconstructed by connecting consecutive stops in trips by the shortest path in the road network. Figure 10 shows an example.

Validation of a small part of the resulting assignments was done by visual inspection of known bus stops and reconstructed bus routes similar to the validation described by both Li [3] and Perrine et al. [4] All inspected bus stops were assigned correctly.

Because interactive visual inspection only covers part of the results, an automated validation method was developed. Timing for the trips is taken from GTFS and trip length is taken from the reconstructed bus trips. We computed (i) the average speed of each complete trip (so-called *commercial speed*, distribution shown in Figure 11a) and (ii) the speed between consecutive stops (distribution shown in Figure 11b). Outliers designate errors in the assignment; they are harmful to travel simulations.

The GTFS dataset contains 6 402 unique trips. One percent of these trips have a commercial speed lower than 16.2[km/h] and one percent of these trips have a speed higher than 53.7[km/h]. The average speed equals 32.0[km/h]. The target commercial speed values for the buses of “De Lijn” are separated into 4 categories: 25[km/h] for suburb and city services, 30[km/h] for feeder services, 35[km/h] for main services and 50[km/h] for express services [7]. The obtained average speed values for the individual trips fit well in these categories.

The GTFS dataset contains 7 780 728 trip segments. A trip segment is the passage of two consecutive stops. Several of these passages can belong to different executions of the same trip. For 83 of those the speed equals 0[km/h] and 896 943 trip segments have a zero duration. Speeds equal to 0[km/h] occur in segments which have a zero distance.

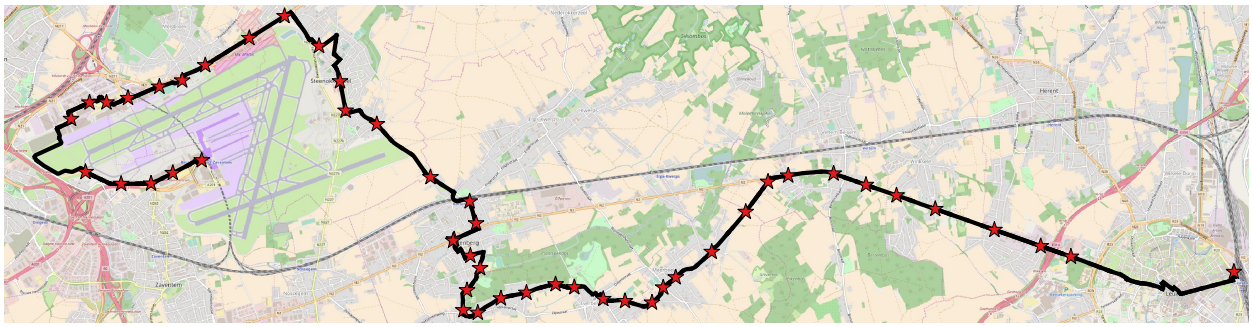
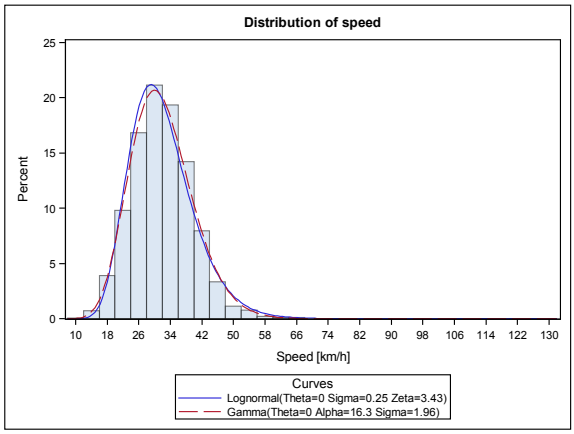
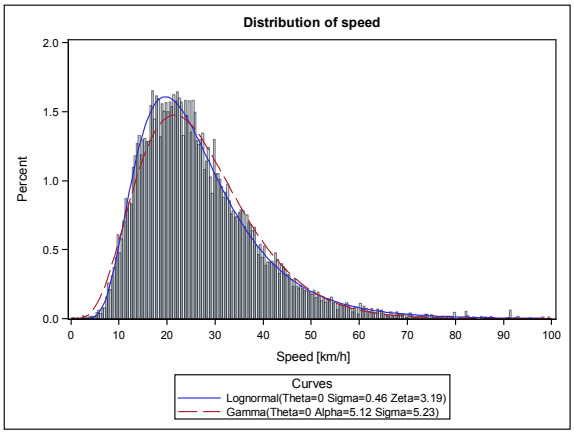


Fig. 10: Visual inspection of a line going from Leuven (right) to Zaventem Airport (left). The black line represents the followed path to serve the GTFS bus stops (red stars).



(a) Distribution of the speeds of all unique trips.



(b) Distribution of the speeds of the segments of all trips. Only speeds lower than or equal to 100[km/h] are considered.

Fig. 11: Speed distributions.

This is the result of coinciding projections of different GTFS stops. The time resolution used by "De Lijn" is one minute. Some bus stops are separated by a small distance so that the traversal time is less than a minute, which results in a zero duration. These segments were removed from our data. One percent of the segments have a speed lower than 8.2[km/h] and one percent of the segments have a speed higher than 74.1[km/h]. The average speed equals 27.3[km/h]. The amount of unique segments that have a speed lower than 8.2[km/h] or higher than 74.1[km/h] is equal to 1 031. The amount of unique stops involved in these segments equals 1 785, i.e. 5.8 % of all GTFS stops. Next, we discarded 27 708 of the 6 883 702 speed segment with values larger than 100[km/h], i.e. we removed the values that were obvious outliers. The resulting segment speed distribution is shown in Figure 11b. Further research includes the investigation of the aforementioned outliers. The source of the outliers will be investigated, i.e. do they emerge due to our optimization assumption, due to the underlying network or due to the data contained in the GTFS dataset. The latter is a possible candidate because the GTFS transit feed validator [4] reports an excessive travel speed ($\geq$ 100[km/h]) on 445 of all trip executions in the GTFS dataset.

Computation results for the Flanders (Belgium) case study are summarized in Table 1. Run times hold for Ubuntu Linux and Java7 on Intel(R) Xeon(R) CPU X5570 @ 2.93GHz. This computation server was used for the data preparation algorithm as well as the bus stop mapping algorithm.

[4] `https://github.com/google/transitfeed/wiki/FeedValidator`

Table 1: Characteristics and Results of the Experiment for Flanders.

Quantity	Value	Quantity	Value
# Network Nodes	641 901	# Network Links	1 627 258
# GTFS stops	30 654	# Projected stops	127 705
Max projStop / GtfsStop	20	# Stop pairs (dist. matrices)	36 836
# Sequences (GTFS trips)	6 477	# Trivial stops (DOF=1)	147
# Cycle breakers	982	$\log_{10}$(complexity)	16 176.502
$\log_{10}$(complexityCycleBreakers)	394.246	# Iterations	28
# Components	688	Duration iterations [sec]	1 546
Dur [sec] components solving	14	Max # DOFs for component	14
Total computation time	24[min] 20[sec]		

6. Conclusion

We presented a bus stop mapping algorithm that is capable of handling all bus trips at once and is able to determine the side of the road of the bus stops. This was a challenging task since the PT network is vastly interconnected. The assignment of the bus stops is computed by optimization, under the assumption that the PT operators minimize the total distance driven to complete all trips. Visual inspection of known bus stops shows correct assignment of these stops. Validation based on average segment speeds shows that less than 6 % of the stops require interactive attention and those stops can be identified automatically in order to correct the data.

Acknowledgements

The research reported was partially funded by the IWT 135026 Smart-PT: Smart Adaptive Public Transport (ERA-NET Transport III Flagship Call 2013 “Future Traveling”).

References

1. Haklay, M., Basiouka, S., Antoniou, V., Ather, A.. How Many Volunteers Does it Take to Map an Area Well? The Validity of Linus' Law to Volunteered Geographic Information. *The Cartographic Journal* 2010;**47**(4):315 – 322. doi:10.1179/000870410X12911304958827.
2. Lektauers, A., Petuhova, J., Teilans, A., Kleins, A.. The Development of an Integrated Geosimulation Environment for Public Transit Analysis and Planning. *Information Technology and Management Science* 2012;**15**(1):200 – 205. doi:10.2478/v10313-012-0026-3.
3. Li, J.Q.. Match bus stops to a digital road network by the shortest path model. *Transportation Research Part C: Emerging Technologies* 2012; **22**:119 – 131. doi:10.1016/j.trc.2012.01.002.
4. Perrine, K., Khani, A., Ruiz-Juri, N.. Map-Matching Algorithm for Applications in Multimodal Transportation Network Modeling. *TRB Research Record* 2015;**2537**:62 – 70. doi:10.3141/2537-07.
5. Ordóñez, S.A.. Semi-Automatic Tool for Bus Route Map Matching. In: Horni, A., Nagel, K., Axhausen, K.W., editors. *The Multi-Agent Transport Simulation MATSim*; chap. 18. London: Ubiquity Press. ISBN 978-1-909188-76-1; 2016, p. 115–122. doi:10.5334/baw.
6. Cich, G., Vuurstaek, J., Knapen, L., Yasar, A.U.H., Bellemans, T., Janssens, D.. Data Preparation to Simulate Public Transport in Micro-Simulations Using OSM and GTFS. *Procedia Computer Science* 2016;**83**:50 – 57. doi:10.1016/j.procs.2016.04.098; the 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016) / The 6th International Conference on Sustainable Energy Information Technology (SEIT-2016) / Affiliated Workshops.
7. De Lijn, , Tritel, , Goudappel-Cofeng, , Mint, . Mobiliteitsvisie De Lijn 2020. 2009.