Determining Structural Route Components from GPS Traces

Luk Knapen[a,*], Irith Ben-Arroyo Hartman[b], Daniel Schulz[c], Tom Bellemans[a], Davy Janssens[a], Geert Wets[a]

[a]*Hasselt University, Transportation Research Institute (IMOB) Wetenschapspark 5 bus 6 3590 Diepenbeek, Belgium*
[b]*Caesarea Rothschild Institute, University of Haifa, Israel*
[c]*Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, Schloss Birlinghoven, Sankt Augustin, Germany*

Abstract

Analysis of GPS traces shows that people often do not use the least cost path through the transportation network while making trips. This leads to the question which *structural path characteristics* can be used to construct realistic route choice sets for use in traffic simulation models. In this paper, we investigate the hypothesis that, for utilitarian trips, the route between origin and destination consists of a small number of concatenated least cost paths. The hypothesis is verified by analyzing routes extracted from large sets of recorded GPS traces which constitute revealed preference information. Trips have been extracted from the traces and for each trip the path in the transportation network is determined by map matching. This is followed by a path decomposition phase for which the algorithm constitutes the first contribution of this paper. There are multiple ways to split a given path in a directed graph into a minimal number of subpaths of minimal cost. By calculating two specific path splittings, it is possible to identify subsets of the vertices (*splitVertexSuites*) that can be used to generate every possible minimum path splitting by taking one vertex from each such subset. As a second contribution, we show how the extracted information is used in microscopic travel simulation. The distribution for the size of the minimum decomposition, extracted from the GPS traces, can be used in constrained enumeration methods for *route choice set generation*. The sets of vertices that can act as boundary vertices separating consecutive route parts contain *way points* (landmarks) having a particular meaning to their user. The paper explains the theoretical aspects of route splitting as well as the process to extract *splitVertexSuite*s from big data. It reports statistical distributions extracted from sets of GPS traces for both multimodal person movements and unimodal car trips.

Keywords: Travel behavior, Route choice, Route decomposition, Transportation modeling, GPS traces, Big data analysis, Graph theory

1. Introduction - Context

Travel demand prediction by means of micro-simulation in activity-based models results in an agenda for each individual for the simulated period of time. Such agenda consists of a sequence of episodes each one of which is defined by a period of time, an activity type, a location and the modes used to reach the location. As soon as the locations are known, the

[*]Corresponding author: Tel.:+32-11-269-126
Email address: `luk.knapen@uhasselt.be` (Luk Knapen)

traffic demand needs to be assigned to the transportation network. Thereto *route selection procedures* are required.

Such procedures are based on utility maximization where the utility depends on person and route characteristics (trip duration, route length, number of left turns etc). In this paper it is proposed to integrate an additional route feature in the choice process in order to account for the route complexity. The additional feature is the minimum number of intermediate destinations that is required to reconstruct the path from least cost (distance, travel time, . . .) components. The idea is that the traveler might have mentally constructed the path as a sequence of such intermediates (which *can* be landmarks) and then tries to reach each one of them as efficiently as possible.

Availability of big data sets of GPS traces allows for statistical analysis of the *structural* characteristics of large sets of routes. Our purpose is to use the automatically extracted information (route decomposition size and candidate landmarks) in the route choice set generation procedure in order to make it less dependent on settings determined by expert opinion.

In the remainder of the text, we deliberately use two sets of terms; they are related as follows:

Road Network Context	: Graph Theory
Node	: Vertex
Link	: Edge
Route	: Walk, Path
Network	: Graph

The minimum size of a decomposition into least cost components was determined for routes derived from two sets of GPS traces. For both of them the traces for each participant were recorded for at least one week. No user interaction with the recording device was required. Hence this data collection can be interpreted as accurate revealed preference. The traces reflect what actually happened.

In order to extract the minimal decomposition size for each path, the recorded data are processed using the following steps:

1. **Trip detection**: the sequences of GPS recordings for each traveler are broken down into subsequences. Each such subsequence corresponds to either a trip (movement) or a stop (stay at a particular location having a non-zero finite area). Stops are not relevant in this study and hence are ignored in the remainder of the paper.

2. **Map matching**: the GPS sequence for each trip is matched to a network consisting of links (road segments) and nodes (junctions). Map matching is applied to each individual trip. It associates a sequence of visited links to the trip (in general a *walk* in a graph). From the set of walks, the subset constituting simple *paths* is kept. The other walks are ignored since they are assumed not to correspond to a trip for which the user aims at maximal utility (minimal generalized cost) since those walks contain at least one node that is visited multiple times without an intermediate stop to perform an activity. Map matching of the GPS sequence for a particular trip leads to a route in the road network that links one activity location to the chronologically next one.

3. **Route Decomposition**: the map-matched route corresponds to a path in a graph in which each edge is labeled with the link travel cost. The purpose is to split the path into a minimal number of *Basic Path Components* (BPC) each one of which is either

a least cost path or a non-least-cost-edge (i.e. a single edge which is not the least cost connection between its vertices). Every two consecutive BPC's in the decomposition are separated by a *splitVertex*.

4. **Statistical analysis** of route structural characteristics.

This paper focuses on the third and fourth steps in the process. First, the problem of route choice is sketched in order to show how the size of the minimum path decomposition can be integrated in the existing models. After this motivation, the concept of splitting routes into basic path components is introduced in section 4. Definitions are given, the concept is elaborated in a mathematical way, theorems on route splitting are proved and the used algorithm is explained. Section 5 is devoted to the interpretation of the detailed route splitting results and their relevancy in travel behavior research. It also formulates research questions that can be solved using the results generated by the algorithm.

Finally, the paper reports the results extracted from the available datasets. A conclusion is presented in section 8.

2. Route Selection Context

Route selection induces a complex discrete choice problem and in general consists of two parts: a route *choice set generator* and a route *choice model*. Prato (2009) provides a comprehensive overview of solutions to the route choice problem. The traveler is assumed to select an optimal route according to personal preferences while having limited information and limited processing capacity. When predicting routes for network loading simulation, either collective of individual choice sets need to be generated for each origin-destination pair. The set of possible routes is huge and the traveler never has a mental representation of the complete set. Furthermore, the choice sets considered by the traveler and the researcher are not necessarily identical. Hence, in the route choice problem, multiple route sets are considered. Several similar schemes have been proposed to classify those sets and the one given in Kaplan and Prato (2010) is shown below: each set is derived from the one mentioned immediately above it.

	Traveler view	Researcher view
Set of all possible routes for OD-pair	Universal set	Universal set
Set of routes feasible based on a given criterion	Master set	Awareness set
Set of routes from which the individual would select	Consideration set	Viable set

For both model estimation and route prediction purposes, the consideration set is constructed algorithmically, in general by making use of non-compensatory techniques Bovy (2009). The consideration set is used in route choice models most of which are derived from MNL (multinomial logit).

Several studies investigate the quality of the choice set generators and of the overall process as well as the mutual influence of choice set generation and choice model estimation. However the effect of including or excluding a given path characteristic is not documented.

Bekhor et al. (2006) evaluate sixteen label minimization/maximization algorithms along with K-shortest path selection, link elimination, link penalty and stochastic link impedance based methods. Choice sets are generated for data collected from MIT researchers and choice models are estimated. The models are based on distance, free-flow travel time, some landmarks identified by the analyst, income indicators and time spent on government numbered routes.

The relevancy of the composition of the choice set is investigated by Prato and Bekhor (2007). The authors investigate the effect of the choice set generation technique on the choice model parameter estimates and on the generated predictions. The study creates two choice sets (one generated by the branch-and-bound technique and one that merges results from labeling, link elimination, link penalty and stochastic link trait adaptations). Six choice models are estimated using each choice set. All models use ten explanatory variables. The effect of combining choice set generation techniques and choice models is investigated. The branch-and-bound generator recursively generates paths in the network and discards paths that do not meet specific criteria (e.g. length, number of left turns at junctions etc). Fixed settings for the criteria used. The effect of the criteria is not investigated, probably because the branch-and-bound technique already outperforms the other ones with respect to *coverage*.

Prato (2012) performs a meta analysis of the effect of the choice set generation technique on the accuracy of the choice model estimates and on the link flow predictions. Deterministic (K-shortest path, link penalties, branch-and-bound) and stochastic (link impedance, combination of link impedance and travel taste, random walk biased to search for the shortest path) are considered. The choice model used is the path size correction (PSC) model. The same techniques are used to generate synthetic data and to generate objective choice sets and choice models. This is done for several parameter sets and in pairwise combinations. The choice models are restricted to account for route length, number of speed bumps, number of turns and the path size correction. The only investigated route generator that can include path attributes, is the branch-and-bound technique. Several settings for thresholds are investigated.

Since the consideration set is not observable, it can be argued that the parameters of the consideration set construction model and the choice model shall be estimated together. This is done by Kaplan and Prato (2010). The consideration set C_n for traveler n is derived from the master set using a *conjunctive heuristic semi-compensatory model*: the probability to find consideration set C_n is given by the probability that respondent n uses a specific set of thresholds for the independent values. If an independent variable is out of range w.r.t. a threshold, the corresponding route is not considered. The thresholds are unknown in advance and they are estimated using a maximum likelihood method. The authors only consider the route length and the number of turns.

Consideration set construction is the point where the research on route decomposition presented in this paper fits. Up to now the feasibility of a route for inclusion in the consideration set is assessed using several attributes (detour factor, number of left turns and others): we propose to additionally include the size of the minimum path decomposition in the assessment. The idea applies to both choice set generation an choice models (explanatory variables). It allows (i) to avoid overly circuitous routes and (ii) to avoid introducing unrealistic bias towards the shortest paths.

Quality assessment using the minimum path decomposition size can be used (i) in the consideration set construction stage (ii) or as a posterior assessment of the generated set. In both cases, distributions for the minimum decomposition size extracted from recorded routes are used. They can be collective or individual; the latter applies to the case where sufficient longitudinal data for each participant are available.

Consideration set construction is discussed by several researchers. For the purpose of this paper the reported research efforts are subdivided in two categories according to how the proposed assessment can be integrated.

2.1. Assessment in choice set construction stage.

The research reported in the following papers allows to use additional route attributes in the consideration set construction stage.

Zijpp and Catalano (2005) present the Constrained K-Shortest Paths (CKSP) technique based on Lawler's algorithm. In case a constraint can be evaluated on the first part of a partially generated path only, a part of the search space can be discarded if the constraint is not met. The CKSP method is based on consecutive least cost path evaluations and the new *minimum path decomposition size* algorithm can be integrated with limited effort. If an upper bound for the minimum decomposition size is specified in advance, the CKSP method can discard subspaces that would deliver overly complicated paths.

Prato and Bekhor (2006) present a deterministic path generator using a branch-and-bound (breadth-first-search) technique that constructs a connection tree of candidate paths between a given origin and destination. Each time a link is to be added several constraints are verified. Given constant factors are used for partial path assessment: (i) a distance factor filters partial trips moving back to the origin, (ii) a time factor filters partial paths taking too long, (iii) a detour factor limiting the length of partial paths relative to the minimal distance between their endpoints and (iv) a similarity constraint that limits overlap between candidates. The generator is applied to a network for the city of Torino, Italy. Commuting trips were recorded using a web-tool. The *branching rules* account for behavioral constraints. The authors evaluate the quality of the choice set using the concept of *coverage*. The authors compare several generation techniques and reports that *coverage* levels attained by branch-and-bound techniques are much higher than for other techniques. Finally, several choice models are estimated using the generated choice sets. The results of our research can be used in branch-and-bound rules. The size of the minimum decomposition can be calculated for the head part each path being built; the complexity is of the same order as the evaluation of the *loop constraint* mentioned by the authors. The distribution for the minimum decomposition size extracted from GPS traces can be used to determine threshold values.

Schüssler et al. (2010) mention the difficulty of avoiding bias when establishing route choice sets for high resolution networks: either behaviorally advanced choice set generation procedures are required or large sets of routes need to be explored and reduced by considering attractivity, plausibility and similarity between routes. The authors present the Breadth First Search Link Elimination (BFS-LE) algorithm suitable for route set generation in high density networks. The minimum decomposition size of the path can be used as a selection criterion in the mentioned reduction phase.

Finally, Pillat et al. (2011) investigate how path assessment can be based on thresholds acquired from GPS recordings. The authors describe an experiment where a route choice set is generated and compared with trips recorded from GPS traces. Preferred routes were collected using a survey and saved in a database by identifying consecutive junctions on a map. From those data, the maximum detour factor as a function of the travel time is derived. In the choice set generation phase, the detour factor changes as the duration of the partially generated path grows. The resulting generated paths are compared to trips derived map-matched GPS traces.

In this paper we propose to use the information extracted from the GPS data to define the selection criteria for use during route generation instead of using it for verification only.

2.2. Posterior assessment of the generated choice set.

In some cases it is not possible to include minimum decomposition size verification in the route generation stage. In such cases, posterior assessment of consideration sets can be applied to the generated results. E.g. the distribution for the minimum decomposition size found in MATSim generated routes can be validated with the one found in GPS traces.

MATSim finds the optimal route for each traveler by micro-simulation. The movements of cars crossing links on the road network are simulated. The time to cross each link depends on the link characteristics and on the link occupation by other cars. Travelers execute their daily plan and derive the time to travel from the network. The plan gets an evaluation score at the end of the day and, under certain conditions, a new one is generated. The basic assumption is that individuals always try to minimize their travel cost (Balmer et al. (2009)) by finding new routes and by changing their departure times. No route choice set is to be determined a priori but built and maintained by the genetic algorithm as an integral part of the plan (agenda).

3. Research Objective

Each route revealed or conceived by a traveler can be decomposed into sub-routes that are least cost routes between their endpoints. The traveler is assumed to have a set of intermediate destinations in mind while constructing a route and to hop between them using least cost sub-routes. The traveler does not stop in the intermediate locations but merely uses them as anchor points in the construction process.

Graph theory is used to analyze route decomposition: definitions and theorems are presented in Section 4. Each path in a graph can be split into least cost subpaths in *several* ways. Each path has one or more *minimum* decompositions i.e. decompositions consisting of the smallest number of least cost paths from which the path can be reconstructed by concatenation.

The size of the minimum decomposition is a structural qualifier for the path that can be used in the consideration set generation as mentioned in section 2. We, therefore, aim to establish the probability distribution for the size of the minimum decomposition from a set of trips extracted from GPS traces.

A *utilitarian trip* represents a travel where the destination is different from the origin and the traveler moves in order to perform a planned activity at the destination location, in other words, it is a travel for a given purpose. We investigate the following:

Hypothesis 3.1. *In utilitarian trips, individuals tend to construct their route as a concatenation of a small number of minimal cost routes i.e.* basic path components *(BPC).*

The individual is thought to make use of a small set of preferential intermediate locations between the origin and destination and to travel in the most efficient way between them. In order for route choice sets to be realistic, the distribution of the minimum decomposition size shall reflect the one found in recorded traces. Generalized route traversal cost is approximated by considering individual link travel cost only. Node traversal cost (e.g. *left turn* cost) is associated with pairs of links and is not covered by the algorithms presented in this research.

This paper presents the decomposition technique and the distribution for the minimal decomposition size. In a forthcoming paper we analyze the use frequency of nodes as split nodes in the routes in order to verify the hypothesis that some nodes are preferential route splitters due to traffic related characteristics of the network (like availability of traffic lights).

4. The main graph-theoretical terms, algorithms and proofs

4.1. Minimal splitting of routes into Basic Path Components

We begin with some definitions from graph-theory. Let $G = (V, E)$ be a directed graph with vertex set V and edge set E. The vertices correspond to *nodes* in a road network, and the edges correspond to *links* in the network. Each edge e has a nonnegative *cost* $c(e)$ which is the effort (e.g. time or money) required to traverse the link in the network. For a subgraph $H \subseteq G$, $V(H)(E(H))$ denote the set of vertices (edges) of H.

Definition 4.1 (walk, initial, terminal, internal vertices, internally-disjoint)**.** *A* walk *is a sequence of vertices $P = (v_0, v_1, \dots, v_l)$, not necessarily distinct, where $(v_i, v_{i+1}) \in E(G)$ for all $i = 0, 1, \dots, l-1$. Vertices v_0 and v_l are called* initial *and* terminal *vertices, respectively, of P, and vertices $v_1, \dots, v_{l-1}$ are called* internal vertices *of P. The walk P is said to be* connecting *v_0 and v_l, and it is also denoted by $P(v_0, v_l)$. A walk $Q(v_0, v_l)$, is* internally-disjoint *from P if all the internal vertices of Q, are distinct from the vertices in P.*

Definition 4.2 (path, subpath, size, cost, least cost path, least cost distance)**.** *A* path *is a walk where all its vertices are distinct. For a path $P = (v_0, v_1, \dots, v_l)$, any subsequence of vertices $v_i, v_{i+1}, \dots, v_j$, where $0 \le i, \le j \le l$ is a* subpath *of P, and is denoted by $P(v_i, v_j)$. The* size *of a path, denoted by $|P|$, is the number of edges in it (i.e. l), and the* cost *of a path, denoted by $c(P)$ is the sum of the costs of its edges. A* least cost path *between u and v is a $P(u, v$ of least cost. The* least cost distance *between u and v, denoted by $lc(u, v)$ is the cost of the least cost path connecting u and v.*

We remark that if $c(e) = 1$ for all $e \in E$ then the cost of a path coincides with its size and that vertex traversal cost is assumed to be zero.

The following lemma is easy to prove:

Lemma 4.3. *If $P = (v_0, v_1, \dots, v_l)$ is a least cost path, then any subpath of P is also a least cost path.*

Proof: Assume, by contradiction, that $P(v_i, v_j)$ is not a least cost path between v_i and v_j, for some $0 \le i < j \le l$. Let $Q(v_i, v_j)$ be a path of smaller cost. Then by replacing $P(v_i, v_j)$ by $Q(v_i, v_j)$ we get a walk connecting between v_0 and v_l of smaller cost than P. This walk contains a path connecting v_0 and v_l of smaller cost than P, since we assumed that the cost function is non-negative. This contradicts the fact that P is a least cost path. ■

The converse of Lemma 4.3 is false since it is possible that all the subpaths of $P(v_0, \dots, v_l)$ (except P itself) are least cost paths, but P is not a least cost path connecting v_0 and v_l and there is another least cost path Q connecting v_0 and v_l. This fact motivates the following definition:

Definition 4.4 (P- shortcut, fork and join vertices)**.** *Let $P = (v_0, v_1, \dots, v_l)$ be a given path. A $P(v_i, v_j)$-*shortcut *(or for brevity, P -* shortcut*, or* shortcut*), is a path $Q(v_i, v_j)$, internally- disjoint from P, where $v_i, v_j \in V(P)$, such that $c(Q(v_i, v_j)) < c(P(v_i, v_j))$. The vertices v_i and v_j are called* fork *and* join *of the shortcut, respectively. (See Figure 1).*

By Lemma 4.3 a least cost path cannot have any shortcuts.

Assume $e = (u, v)$ is an edge in P whose cost is larger than the least cost path connecting u and v. Then e is called a *non-shortest-edge* .

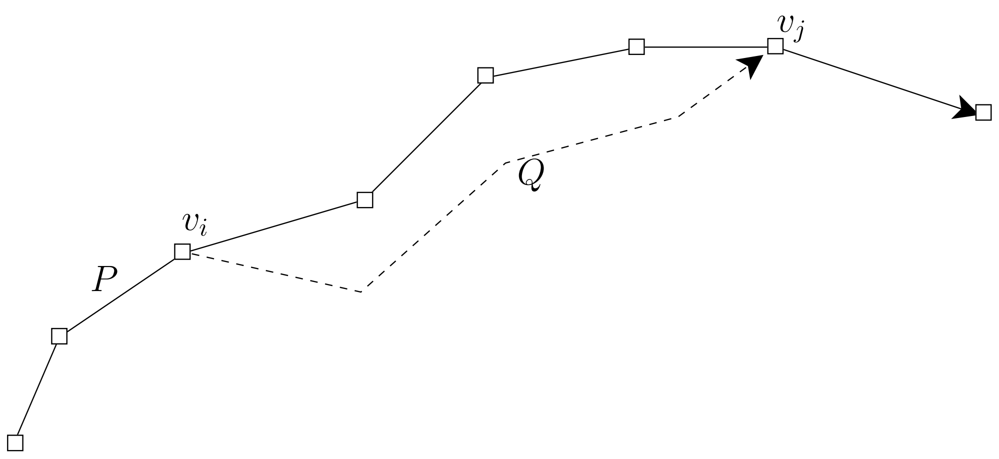


Figure 1: A path P with a $P(v_i, v_j)$-shortcut. v_i is a fork vertex and v_j is a join vertex

Definition 4.5 (Basic Path Component (BPC))**.** *Given a path P, a subpath of P is called a* Basic Path Component*, or for short, a BPC, if it is either a least cost path connecting its endpoints, or P consists of a single non-shortest-edge.*

Claim 4.6 (path splitting)**.** *Let $P = (v_0, v_1, \ldots, v_l)$ be any path connecting v_0 and v_l. Then $E(P)$ can be partitioned into BPC's.*

Proof: The trivial partition into edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{l-1}, v_l)$ is an example of such a partition. ■

There are many ways to do path splitting, the trivial partition is among them. We are interested in finding a path splitting with a *minimum number* of basic path components. Such a path splitting is called *minimum path splitting.* Each non-shortest-edge is a part in each minimum path splitting since it constitutes a BPC. If we remove the set of non-shortest edges in a path (each of which is a BPC), we are left with a set of disjoint paths, each of which contains no non-shortest-edges.

From now on we will assume that P does not contain any non-shortest-edges.

We will address the following problem:

Problem 4.7 (minimum path splitting)**.** *Given a path $P = (v_0, v_1, \ldots, v_l)$ with origin v_0 and destination v_l, and assume P does not contain any non-shortest-edges. Find efficiently a minimum path splitting of P .*

A vertex separating two consecutive BPC is called a *splitVertex.* To solve problem 4.7 we note that a minimum path splitting will contain a minimum number of splitVertices (since, by definition, any two consecutive basic path components are separated by a splitVertex). Hence, an equivalent formulation of Problem 4.7 would be to find a minimum number of splitVertices in P, denoted by $v^s_{i_1}, v^s_{i_2}, \ldots v^s_{i_k}$, such that any subpath connecting consecutive splitVertices will be *least cost.*

Lemma 4.8. *Let $P = (v_0, v_1, \ldots, v_l)$ be a path connecting v_0 and v_l. Assume P is not least cost, and let $Q(v_i, v_j)$ be a shortcut in P. Then any path splitting of P will contain at least one internal vertex in the path segment $P(v_i, v_j)$.*

Proof: By contradiction. If no internal vertex in $P(v_i, v_j)$ is a splitVertex, then $P(v_i, v_j)$ is a least cost path, contrary to the fact that $Q(v_i, v_j)$ is a shortcut in P. ■

We are now ready to describe an efficient algorithm for partitioning a given path P into a minimum number of basic path components. The algorithm begins with the initial vertex of P, v_0, and finds a maximal least cost path beginning with it. This is done using Dijkstra's least cost path algorithm (Dijkstra (1959)). Assume v_{j_1} is the first vertex on P for which $P(v_0, v_{j_1})$ is not least cost, then the algorithm marks v_{j_1} as a join vertex and continues with the subpath of P beginning from the vertex prior to v_{j_1} on P, looking for the next join vertex in $P(v_{j_1-1}, v_l)$. We continue until no more join vertices are found. The pseudo code is given below. See also Figure 2.

Algorithm 4.1 Algorithm for finding a partition of a path into BPC's

```
  Input Graph $G$, edge costs $c$, $P = (v_0, v_1, \ldots, v_l)$ containing no non-shortest edges
  $start \leftarrow 0$;
  $k \leftarrow 1$
  while ($P(v_{start}, v_l)$ is not a least cost path) do
      Find the first vertex $v_{j_k}$ in $P(v_{start}, v_l)$ such that $lc(v_{start}, v_{j_k}) < c(P(v_{start}, v_{j_k}))$
      $start \leftarrow j_k - 1$
      $k \leftarrow k + 1$
  end while
  return join vertices $v_{j_1}, v_{j_2}, \ldots, v_{j_{k-1}}$
```

Theorem 4.9. *Algorithm 4.1 finds a partition of path P into a minimum number of BPC's.*

Proof: Given the output of the algorithm, we will partition the given path P into exactly k subpaths . Define the splitVertices to be the vertices preceding the join vertices on P, i.e. $v_{s_i} = v_{j_i-1}$ for $1 \leq i \leq k-1$. Each subpath begins in a splitVertex and ends in the next splitVertex, except for the first subpath which begins in v_0 and the last subpath which ends in v_l. In other words, the subpaths are: $P(v_0, v_{s_1}), P(v_{s_1}, v_{s_2}), P(v_{s_2}, v_{s_3}) \ldots, P(v_{s_{k-1}}, v_l)$. By the algorithm, it is clear that each of these subpaths are minimum cost, and hence are BPC's. We are now left to prove that no other partition exists with fewer than k BPC's. Assume, by contradiction, that such a partition exists, with $k-1$ BPC named $\overline{P_1}, \overline{P_2}, \ldots, \overline{P_{k-1}}$. Then, by the pigeon hole principle, at least one BPC, say $\overline{P_i}$ contains at least one splitVertex of P, say v_{s_t} as an internal vertex of $\overline{P_i}$ and $v_{s_{t-1}}$ (or v_0 when $t = 1$) is also included in $\overline{P_i}$. This implies that the join vertex v_{j_t} is included in $\overline{P_i}$, and hence $\overline{P_i}$ is not a least cost path, which is a contradiction. ■

We have described an algorithm of splitting a path into a minimum number of BPC's, and finding splitVertices that separate between these basic path components. The algorithm is efficient since it uses Dijkstra's algorithm no more than k times, where k is the minimum number of BPC's used to partition P. Two natural questions arise here; is the partition into a minimum number of BPC's unique, and are the splitVertices unique? Since the splitVertices may denote some point of interest for the traveler (otherwise a minimum cost path would have been chosen), we are interested to find efficiently other splitVertices and partitions into BPC's.

We use Algorithm 4.2 to find another partition of P into BPC's, starting from the end of the path, and a collection of fork vertices in the given path P. This is done by 'going backwards' from v_l to v_{l-1} etc. and finding the first vertex v_{f_1} such that the subpath $P(v_{f_1}, v_l)$ is not a least cost path, but $P(v_{f_1+1}, v_l)$ is a least cost path. The algorithm to

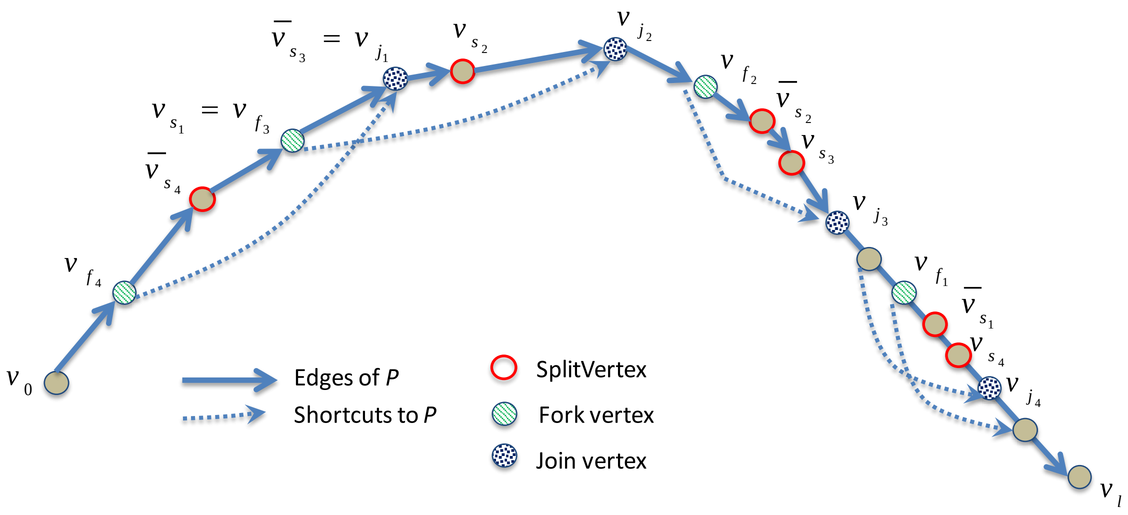


Figure 2: A path P with join and fork vertices and basic path components.

produce fork vertices is similar to Algorithm 4.1 , except that we run it on the "reverse graph" of G, obtained by reversing all the edges in G. Note that when all the edges in G are reversed, the first vertex v_0 of the " reversed path " $\overleftarrow{P}$ corresponds to the last vertex v_l of the original path P. We include the algorithm for completeness.

Algorithm 4.2 Algorithm for finding a partition of a path into BPC's and fork vertices.

```
**Input** Graph $G$, edge costs $c$, $P$ containing no non-shortest edges
 Reverse the edges in $G$; Assume the reversed path is $\overleftarrow{P} = (v_l, v_{l-1}, \dots, v_0)$
$start \leftarrow l$;
$k \leftarrow 1$
**while** ($\overleftarrow{P}(v_{start}, v_0)$ is not a least cost path) **do**
    Find the first vertex $v_{f_k}$ in $\overleftarrow{P}(v_{start}, v_0)$ such that $lc(v_{start}, v_{f_k}) < c(\overleftarrow{P}(v_{start}, v_{f_k}))$
    $start \leftarrow f_k + 1$
    $k \leftarrow k + 1$
**end while**
**return** fork vertices $v_{f_1}, v_{f_2}, \dots, v_{f_{k-1}}$
```

Now define another set of splitVertices $\overline{v_{s_i}} = v_{f_i+1}$ for $1 \le i \le k-1$, which are the vertices following the fork vertices on P found in Algorithm 4.2. It is easy to see, as in the proof of Theorem 4.9, that the k subpaths $P(v_0, \overline{v_{s_{k-1}}}), P(\overline{v_{s_{k-1}}}, \overline{v_{s_{k-2}}}), \dots, P(\overline{v_{s_1}}, v_l)$ are all BPC's, and hence are a minimum partition into BPC's. (See Figure 2).

4.2. Selecting splitVertices for minimum path splitting

We have seen in section 4.1 that there are at least two sets of splitVertices which break up the given path P into k BPC's. One set was obtained by looking for maximal shortest paths starting from the beginning of P, these vertices were labeled by $v_{s_1}, v_{s_2} \dots, v_{s_{k-1}}$, and the other was obtained by looking at maximal shortest paths starting from the end vertex of P. These were labeled as $\overline{v_{s_1}}, \overline{v_{s_2}}, \dots, \overline{v_{s_{k-1}}}$. Each such set breaks up P into exactly k BPC's. Denote by S_i the sequence of consecutive vertices on P in which $\overline{v_{s_{k-i}}}$ is the first one and v_{s_i} is the last one, for each $1 \le i \le k-1$. We call each such sequence *splitVertexSuite (SVS)*. Then any partition of P into a minimum number of BPC's is obtained by choosing

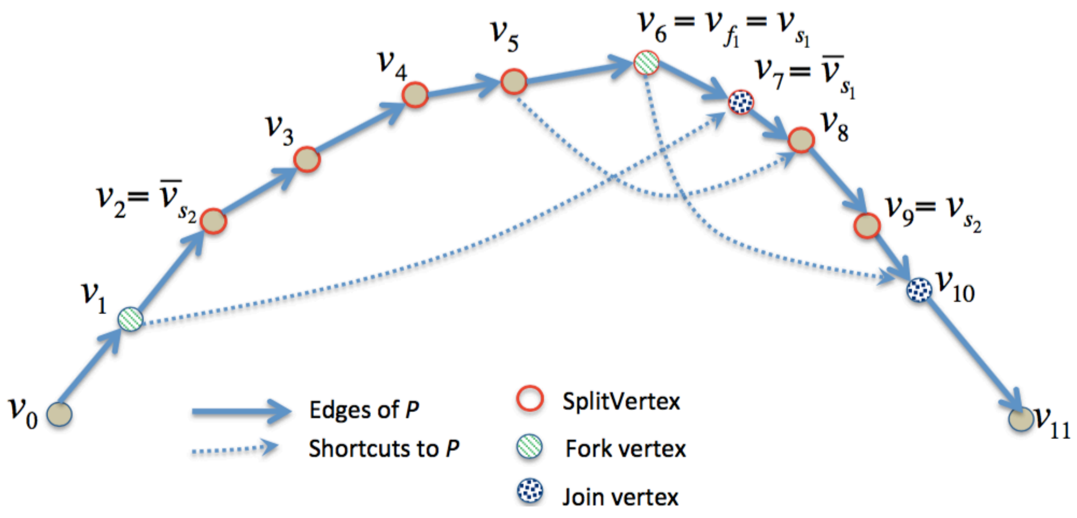


Figure 3: A path P and an "invisible" shortcut ($v_5 \to v_8$) to P.

a unique splitVertex from each splitVertexSuite S_i. Hence the number of ways of splitting a path into a minimum number of BPC's, N_P is bounded above by

$$N_P \leq \prod_{1 \leq i \leq k-1} |S_i| \tag{1}$$

The following example (see figure 3) shows why in some cases, this is a strict upper bound, i.e. the actual number of partitioning P into a minimum number of BPC's is smaller than this bound. In figure 3 our algorithms discover the join vertices v_7 and v_{10} and fork vertices v_6 and v_1. The SVS's are $S_1 = \{\overline{v_{s_2}} = v_2, \ldots, v_{s_1} = v_6\}$ and $S_2 = \{\overline{v_{s_1}} = v_7, \ldots, v_{s_2} = v_9\}$. However, if for example, we choose the split vertices v_3 and v_9 then we do not break up P into 3 BPC's since the middle subpath, $P(v_3, \ldots, v_9)$ contains a shortcut between v_5 and v_8, which was not discovered by our algorithm, and hence is not a BPC. In fact, in this example any splitting of P into 3 BPC's must not avoid the vertices v_6 or v_7 . We conclude that formula 1 is an upper bound for the number of ways to split a path into a minimum number of BPC's. In the sequel paper we will show how to avoid this problem of an "*invisible shortcut*" and compute precisely the number of ways to split a path into a minimum number of Basic Path Components.

5. Interpretation of route decomposition

In Section 4 we described an algorithm for finding a partition of a path into a minimum number of BPC's, and finding a collection *splitVertexSuites* which separate between the

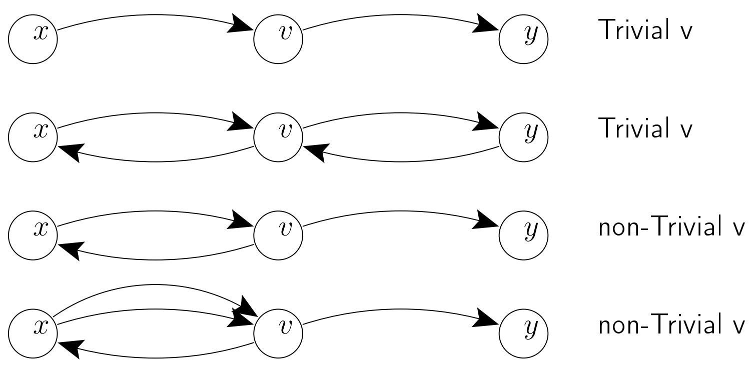


Figure 4: Examples of trivial and non-trivial nodes having exactly two neighbors in the road network.

basic path components. From this information, the researcher knows how many intermediate destinations, but not which particular combination, the traveler had in mind while composing the route.

5.1. SplitVertexSuite*s and the detection of*

traveler intentions. It is reasonable to assume that at least one vertex (a *splitVertex*) in each *splitVertexSuite*, or one of its incident edges belonging to the path, has a special meaning to the traveler; otherwise there would be no reason for a rational traveler to visit the subpath determined by the *splitVertexSuite* and not take the shortcut. We do not know, however, which vertices in the *splitVertexSuites* the traveler had in mind as special vertices acting as intermediate destinations. The larger the size of a *splitVertexSuite*, the more uncertain we are about the motivation of the traveler. The value of N_P given by inequality (1) is a measure for the uncertainty about the set of road landmarks motivating the selected route.

5.2. Network Normalization

In order to calculate the uncertainty value, a preprocess step of *network normalization* is required. In road networks, both directed (one-way) and undirected (bidirectional) links coexist. The corresponding digraph is verified in advance not to contain any sources or sinks (to assess data quality). Parallel edges (i.e. two different edges sharing a single ordered pair (v_a, v_b) of vertices) are allowed in graphs representing road networks.

A *trivial vertex* v is defined as a vertex with exactly two neighbours, each of which is connected to v by at most one edge in each direction and in addition $indegree(v) = outdegree(v)$. (see Figure 4 for examples).

In digital maps, trivial vertices (nodes) are used to subdivide road segments into parts at locations where a road segment attribute value changes (e.g. speed limit, number of lanes, municipality name, road owner, pavement type, etc.). These changes by themselves are assumed to be irrelevant to the traveler while selecting a route. It is however obvious that an *aggregated* measure (e.g. the fraction of the road length having good quality pavement) can be a reason to select a particular link.

In a path of trivial vertices, either all or none belong to the same *splitVertexSuite*, since no trivial vertex can be a fork or a join vertex. This observation allows for *network normalization* by link contraction which removes all trivial vertices. Network normalization does not change the number of *splitVertexSuite*s neither does it affect the set of *splitVertices* since it is reasonable to assume that trivial vertices are not *splitVertices.* This leads to an unambiguous estimation of the uncertainty about the *splitVertex* selection. Figure 5 depicts

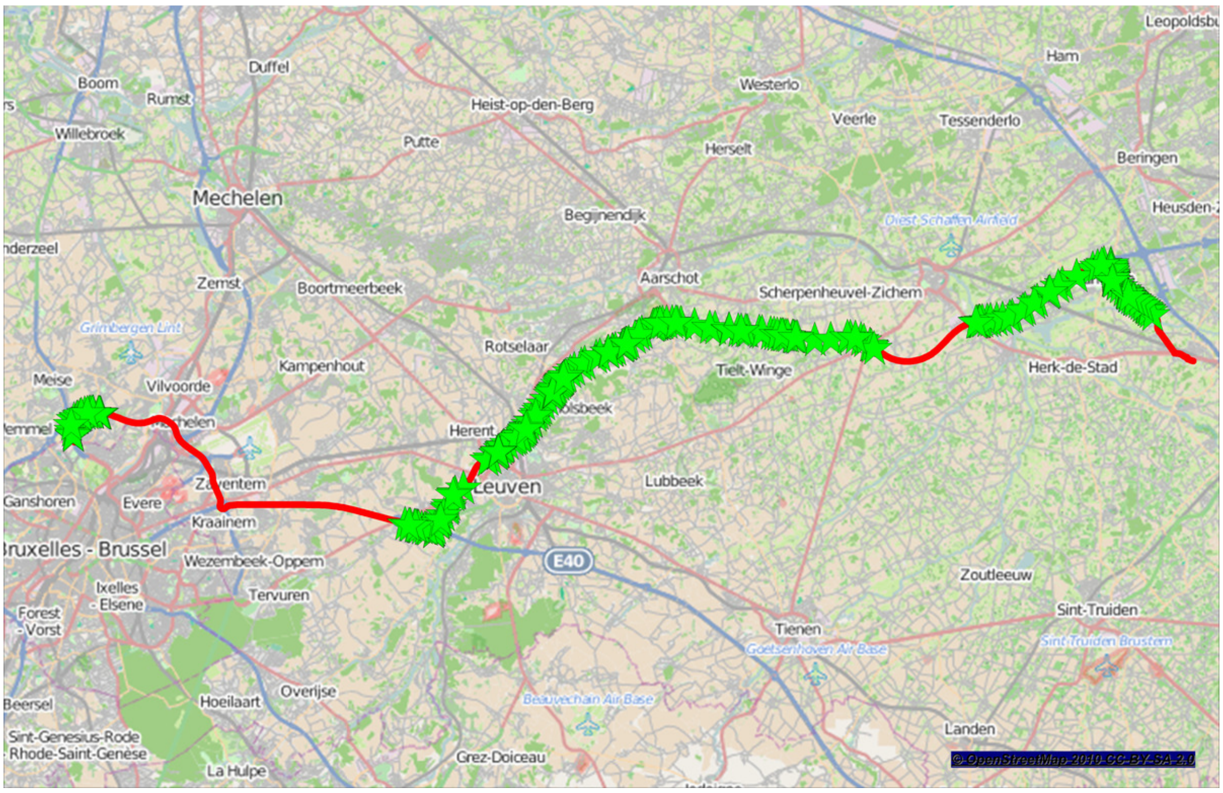


Figure 5: The largest part of the route (from Herk-de-Stad (right) to Kraainem (left)) makes use of the E314 and E40 highways. *SplitVertices* (represented by green star symbols) were determined using the raw (non-normalized) OSM network. The diagram shows the need for network normalization.

clearly the need for normalization. We add the process of network normalization after *map matching* and before *route splitting*.

5.3. Frequency distributions to feed simulators

We are interested in probability distributions for the following parameters:

1. The minimum number of BPC's for each route.

   Distributions of the minimum number of basic path components are essential to route generation. The distribution derived from the complete set of traces uncovers characteristics for the complete observed population. Similar distributions derived for specific individuals may reveal the existence of several behavioral categories when sufficient longitudinal data are available.

2. The use frequency of each *splitVertex* at the population level.

   This is a measure for the attractiveness of the vertex or its incident edges. This provides additional input to the route generator when applied to the region where the traces have been captured. It allows to automatically identify specific spots on the network that serve as *way points* for route generation methods. Analyzing the use frequency of road network nodes as *splitVertex* and the links they delimit, will elicit *network* characteristics (as opposed to the minimum size of a decomposition which is a *trip* characteristic). Frequently used *splitVertices* are interpreted as *route attractors*. The question is whether they can be associated with specific elements in the road network (highway entry/exit ramps, traffic lights, tunnels) or with specific (types of)

POI (point of interest) like a school, public transportation station, carpool parking etc. This analysis requires a much larger data set than the one that was available for research reported here.

3. The use frequency of each *splitVertex* at the individual level.

   This may uncover short intentional stops (bring/get, pick/drop activities) that are not discovered by the trip detector because of their short duration. Current trajectory annotation literature (e.g. Giannotti et al. (2007), Zheng et al. (2009), Kuijpers et al. (2009), Spinsanti et al. (2010), Alvares et al. (2007), Andrienko et al. (2011), Furletti et al. (2013)) focuses on stops found in GPS traces. From mobility science point of view, it is also relevant to annotate (i.e. to attach a meaning to) *splitVertexSuites*. Annotating *splitVertices* is expected to be more complex than annotating stop locations because of the uncertainty mentioned above.

6. Data preparation steps

In order to investigate Hypothesis 3.1, a large set of GPS trajectories has been analyzed.

6.1. Belgian Person Traces

A set of 999 GPS traces recorded during the period 2006-2008 using a PDA have been analyzed. People took the PDA with them: the result is a set of *person* traces as opposed to *car* traces often used. Person traces contain more information but are more expensive to collect over a long period and are sensitive to omissions because people can forget to take the device with them. GPS recording frequency was 1[Hz].

6.1.1. Processing Method 1 : IMOB tools + OpenStreetMap Network

1. Trip detection was performed by finding recording gaps and by analyzing speed variations in sequences of GPS recordings. The detected trips can have several modes (e.g. car-train-walk) but mode detection was not performed (although walking and biking are quite easily identified).

2. Some trips have been detected to start/end at a petrol station located near a highway as a result of the threshold values used for *stop detection*. Those were not altered because refueling can be considered to be a shopping activity.

3. OpenStreetMap (http://www.osm.org/) was used to extract a road network for Flanders (Belgium). The network has 479920 links and 372608 nodes. Trips have been *map-matched* onto that network. The map matching step is crucial. Some map matchers try to fill (small) gaps in the recording by assuming that the traveler moved along a least cost path (according to some criterion). This shall not be done in this research because the hypothesis to be tested shall not be influenced by hypotheses used while map matching. Because of the high recording frequency, it can be expected that every road segment used by the traveler is selected by at least one GPS point. Traces of this kind are called *high density recordings.* Use of high density recording was essential to the reported research. Furthermore, map matching high density recordings can benefit from topological information available from the network. Making use of that information makes the matching process efficient. The map matcher for high density recording described by Knapen et al. (2015) was used.

   Trip detection and successive map matching resulted in 13098 cases.

6.1.2. Processing Method 2 : Fraunhofer tools + Navteq Network

The set of recordings used in section 6.1.1 has been map-patched to the Navteq network which was not normalized. This network consists of 903.217 links and 748.705 nodes for Belgium. The tool used to perform the map-matching was written by Fraunhofer IAIS. In general the process used within this map matching tool consists of two steps.

In the first step a data preparation is done. This includes the detection of outliers and standstills. GPS points falling in at least one of both categories were excluded from the map matching. In addition the time gaps M [min] between two consecutively logged GPS points, were calculated.

In the second step the map matching process is conducted. The goal is to connect each GPS point to exactly one Navteq street segment. Here the map matcher uses a combination of a geometrical and topological map matching. This means that both, the spatial distance of the GPS points to the Navteq street network, and the information of the topological connections in combination with driving restrictions, are used to assign a GPS point to a street segment.

In addition small gaps within the GPS traces were closed by a shortest path routing. Here we worked with three different time gaps (M = 1.0, 2.5 and 5.0[min]). When M grows, the number of detected trips decreases and their average size (distance, number of road network links used) grows. Furthermore, the probability that a reported trip is not a simple path but a walk, grows because small movements are combined to a single trip.

6.2. Italian Car Traces

A set of car trajectories recorded in the region of Milano, Italy was processed using the Fraunhofer IAIS map matcher and the Navteq network (Processing Method 2). The numbers of nodes and links for Italy are nearly ten times larger than the corresponding values for Belgium.

7. Analysis Results

The results for a given trip consist of zero or more *splitVertexSuite*s and the size of the minimum path decomposition (number of BPC). In all experiments described below, *distance along the road* (as opposed to travel time) was used as the generalized cost value to travel a road segment.

7.1. Examples of Discovered splitVertexSuites

Sample routes extracted from the recorded datasets are shown in following figures in order to grasp the idea of the observed *splitVertexSuites.* All diagrams have been generated using the Navteq network.

Figure 6 shows a trip of about 16[km] consisting of three basic components. The large *splitVertexSuite* near Sint-Truiden coincides with a segment of an arterial *express road* which runs parallel to the straight line north-west to it. Higher speed is allowed on the arterial road but the distance is longer. The large size of the *splitVertexSuite* again shows the need for network normalization.

Figure 7 shows part of a route starting at the right hand side, visiting the center of the city of Geel, then moving around the city in clockwise direction, heading to the north and finally arriving near the center of the city of Mol. The partial route shows 11 *splitVertexSuite*s and Figure 8 shows the lower-left part of the same route. The lower-right *splitVertexSuite* in Figure 8 suggests that a particular street in the city center was an intermediate destination.

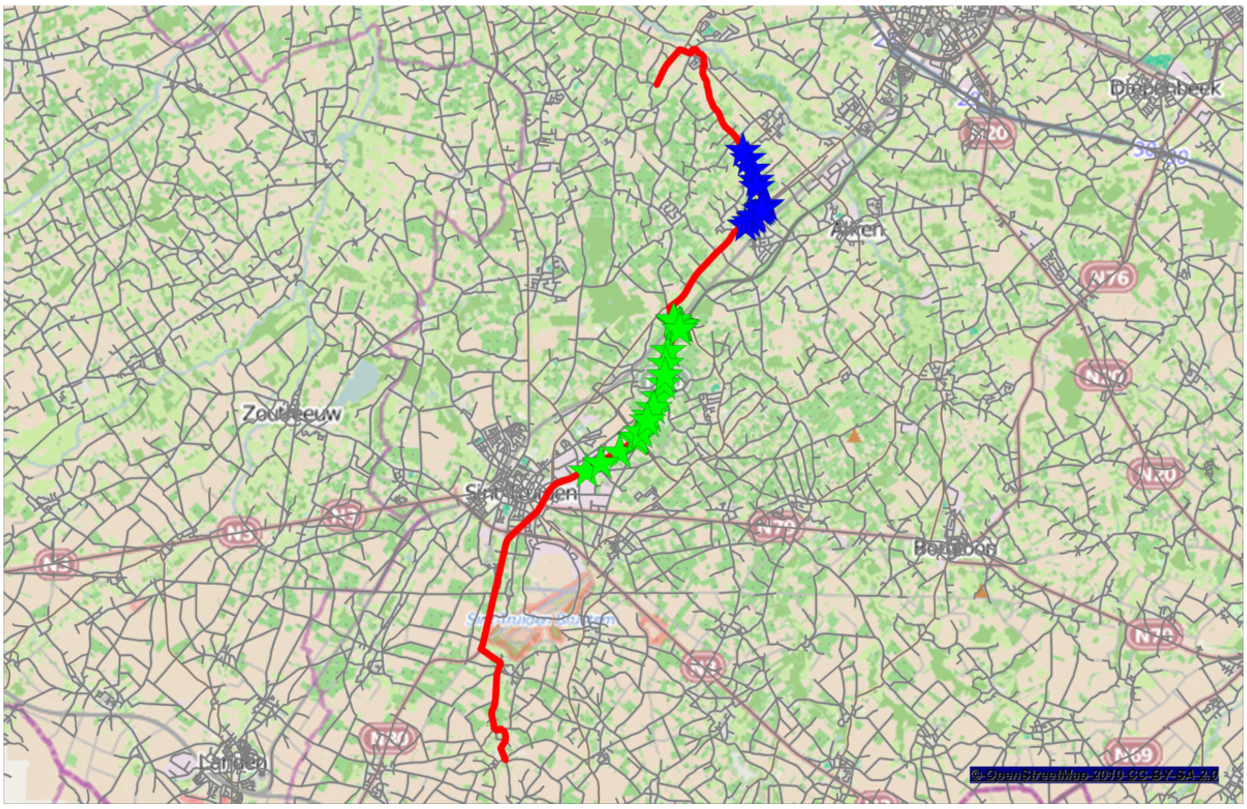


Figure 6: Route consisting of three basic components and showing the need for network normalization. The rectangle measures approximately 40[km] by 25[km].

The lower-left *splitVertexSuite* suggests the intentional use of the ring way and/or a specific junction.

Figure 9 shows a route of about 4 kilometers having 7 *splitVertexSuites* first visiting something special at the first *splitVertexSuite* and then avoiding the narrow streets in a residential area up to the 4-th *splitVertexSuite* which represents a location equipped with traffic lights. The arterial road is used up to the 5-th *splitVertexSuite* which also contains a junction equipped with traffic lights. The trip ends near the parking of a shopping center. The 7-th *splitVertexSuite* is the upper-right one in Figure 10. It is an artifact caused by the fact that the street labeled *Van Groesbeekstraat* (south of the *splitVertexSuite*) constructed in 2012-2013 did not exist at the time of trajectory recording (between 2006 and 2008).

7.2. Distributions for the Size of the Splits at the Population Level

For the interpretation of the results, it is important to remind the definitions for *stop* and *trip*. A trip is a movement between chronologically consecutive stops. A stop is location where the traveler intends to execute an activity. Hence, a *stop* corresponds to a *standstill on purpose*. Not every standstill constitutes a stop (e.g. waiting for a traffic light is a standstill but not a stop). Furthermore, a *splitVertex* acts as an intermediate destination when the traveler mentally constructs a route from least cost components.

Figure 11 shows the absolute and relative distributions for the number of *basicComponents* per trip computed for the respective complete sets of trips. Results are shown for the cases identified in section 6 based on the map matching method and the capturing region.

1. The relative frequency distributions suggest that Hypothesis 3.1 holds. The distributions depend on the methods used for trip detection and map matching. The distri-

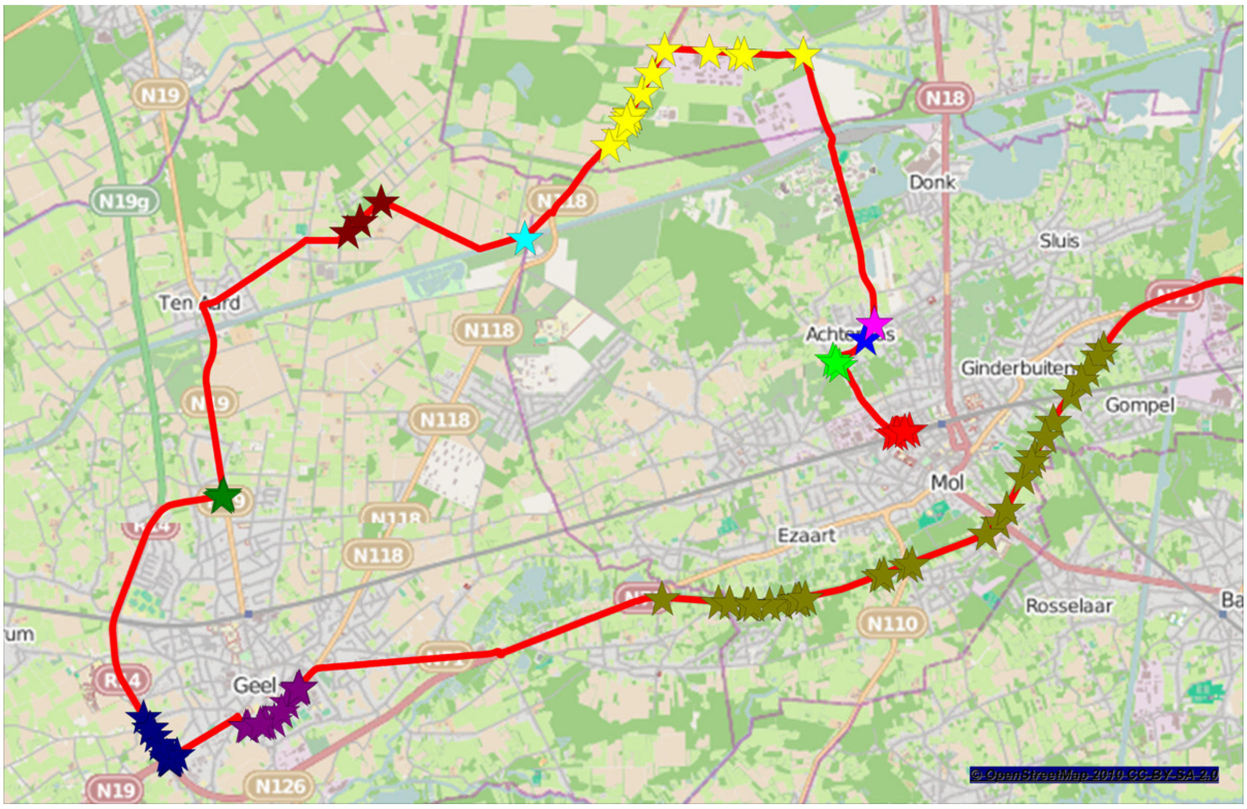


Figure 7: Part of a larger route showing a large number of *splitVertexSuites*.

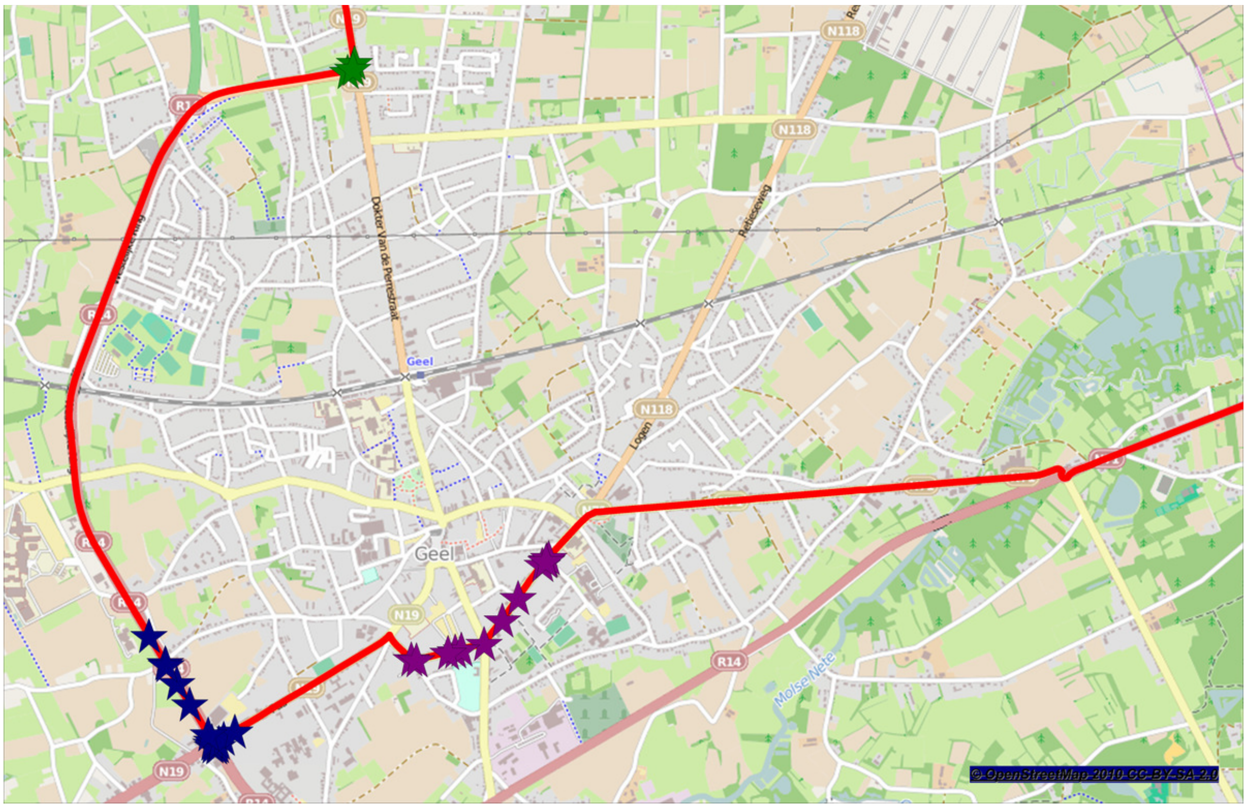


Figure 8: Detail view of Figure 7 showing splitVertexSuites in and near the city center.

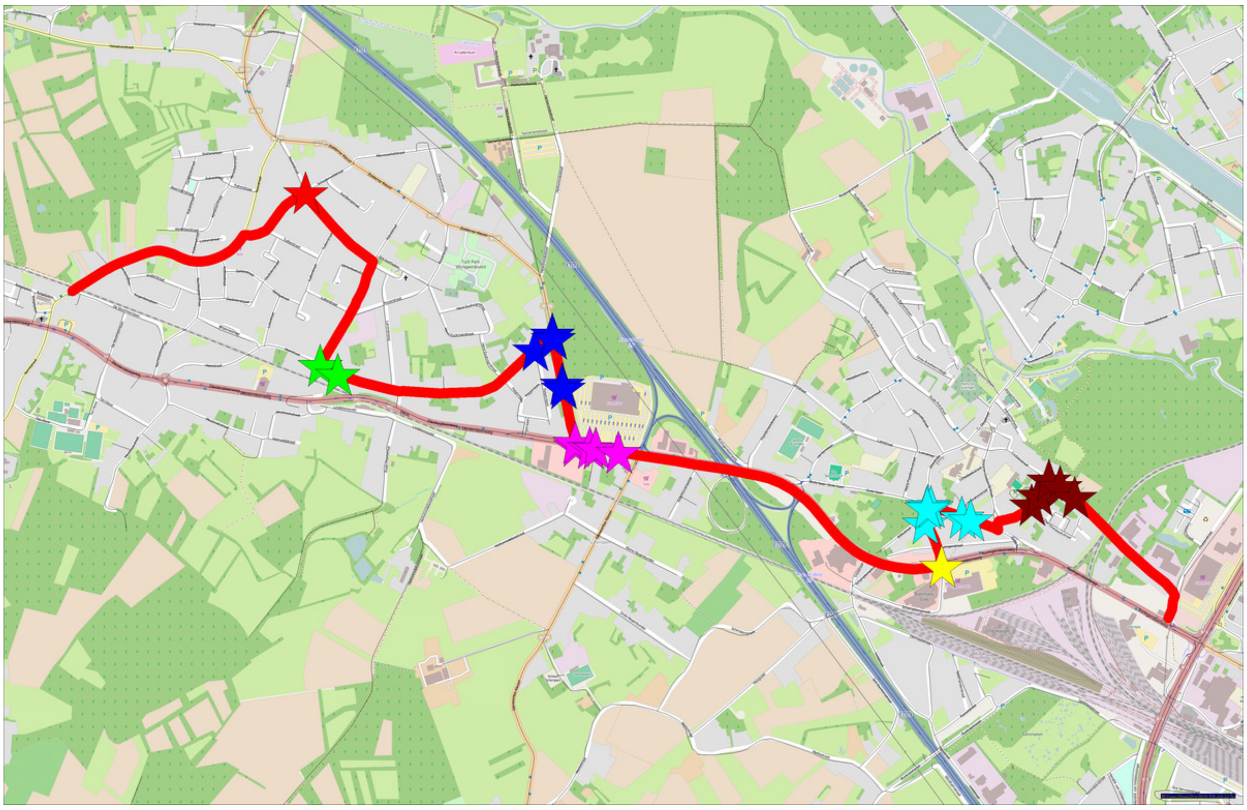
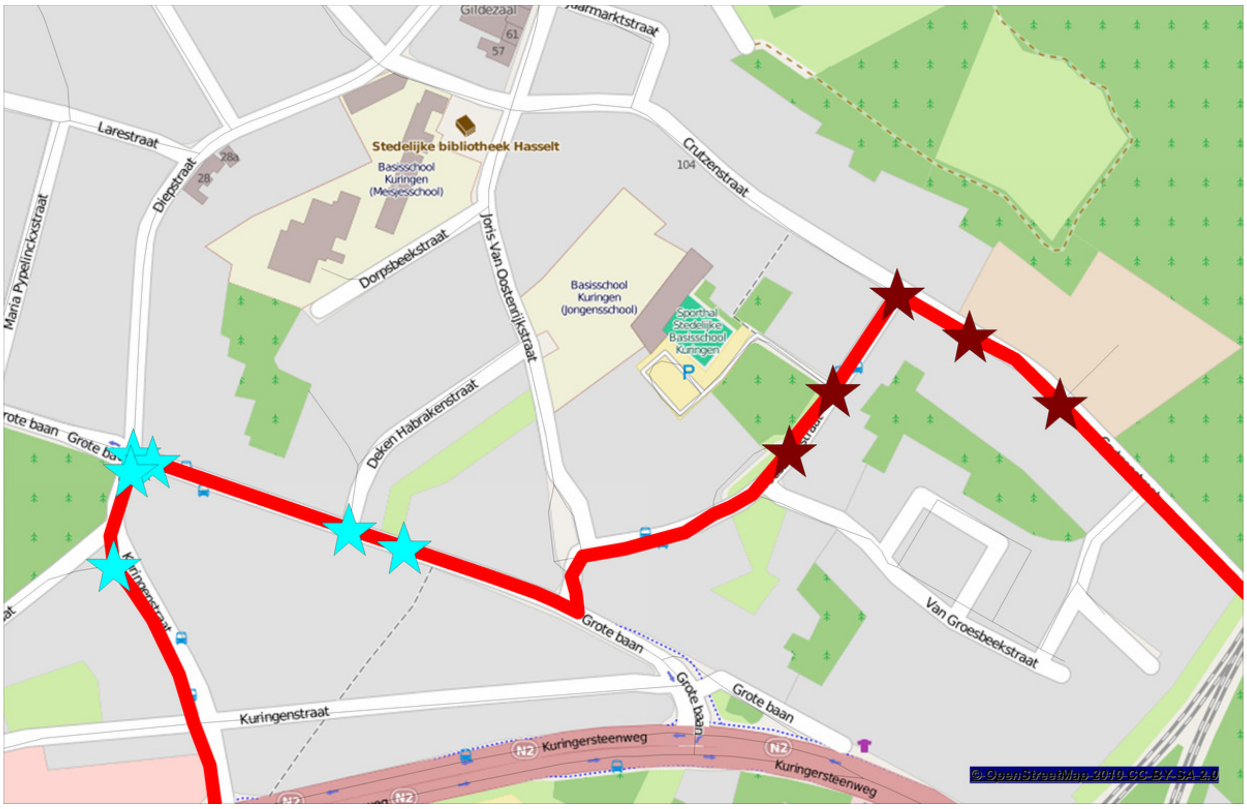
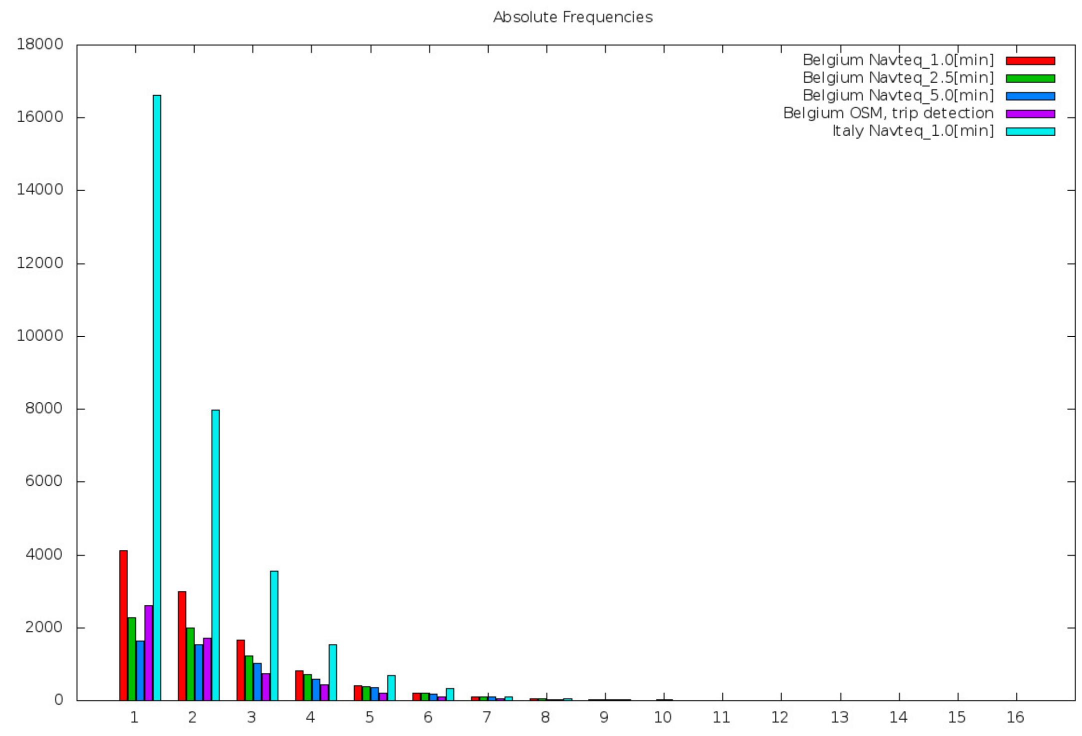
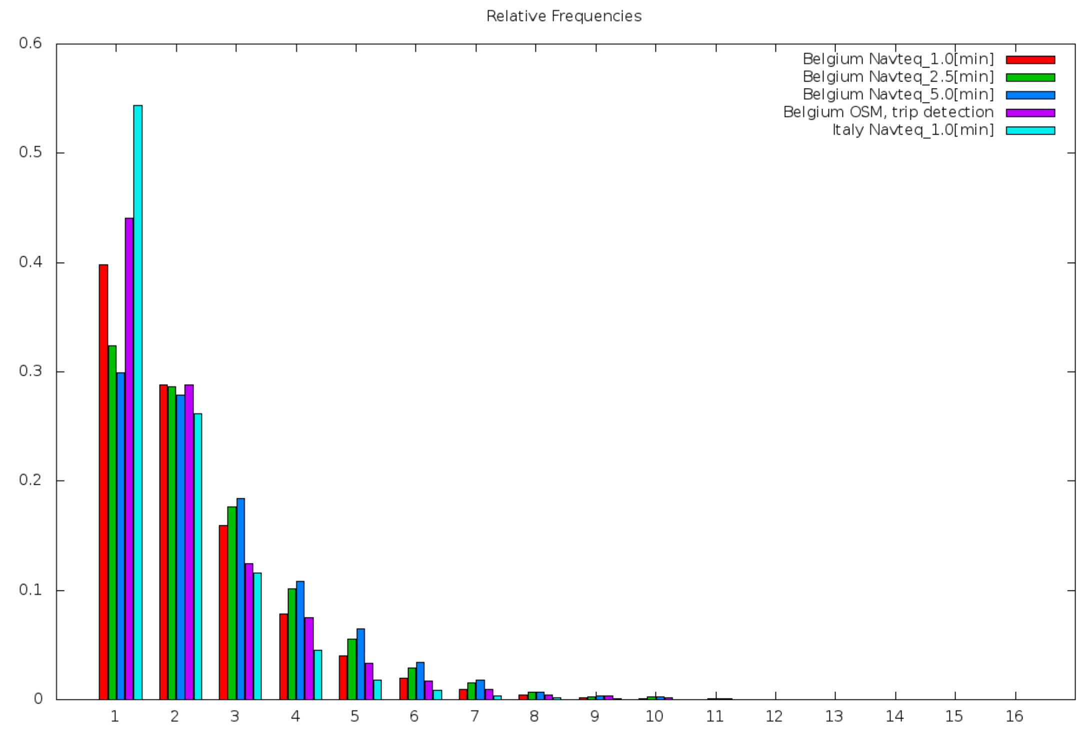


Figure 9: Route showing *splitVertexSuites* that correspond to traffic lights.


Gildezaal
Larestraat
Diepstraat
Maria Pypelinckxstraat
Stedelijke bibliotheek Hasselt
Basisschool Kuringen (Meisjesschool)
Dorpsbeekstraat
Joris Van Oostenrijkstraat
Crutzenstraat
Basisschool Kuringen (Jongensschool)
Sporthal Stedelijke Basisschool Kuringen
Deken Habrakenstraat
Grote baan
Kuringenstraat
Van Groesbeekstraat
Kuringersteenweg
N2
© OpenStreetMap 2010 CC-BY-SA 2.0
Figure 10: Detail of Figure 9 (rightmost part of the route).



Absolute frequency distribution for the number of *basicPathComponents*.



Relative frequency distribution for the number of *basicPathComponents*.

Figure 11: Frequency distributions (top:absolute, bottom:relative) for the number of *basicComponents* per trip. The number of trips in each set depends on the map matcher used.

butions labeled *Belgium Navteq_x.y[min]*, differ only in the value for delay threshold parameter used in stop-detection. The smaller the threshold value, the more stops are detected and hence the smaller the size of the trips (expressed as the number of links they contain). For a given sequence of GPS recordings, the more subsequences are flagged as stops, the lower the number of detected *basicPathComponent*s; this occurs because some briefly visited locations will be flagged as a *stop* when using a small delay threshold parameter value whereas they are detected to be a *splitVertexSuite* in the opposite case. This is reflected in the relative frequency distribution diagram. The probability (relative frequency) to find routes having 1 *basicPathComponent* decreases with increasing value of the delay parameter (1.0 , 2.5 , 5.0). For the case of 2 *basicPathComponents* the phenomenon is largely attenuated. Starting at 3 *basicPathComponents* per trip, the effect is reversed as expected.

2. For Belgium, the OSM and Navteq cases seem to slightly differ. This can have been caused by the use of different map matching tools, based on different methods and concepts. The Fraunhofer IAIS map matcher closes small gaps by assuming that the traveler used the shortest path between successive recordings. The IMOB map matcher does not use this procedure. This conclusion is not final because the cases differ both in the network and the map matcher used.

3. The difference between the Belgian and Italian cases, however, is much larger. The relative frequency for trips consisting of a single *basicPathComponent* is much larger than for the trajectories registered in Belgium. For routes having more than one component, the relative frequency is lower than in any Belgian case. Detail analysis is required to find out whether this phenomenon occurs because in the Italian data set, the pre- and post-car-trip components are missing (Italian traces are car traces).

7.3. Algorithm Execution Run Times

Table 1 summarizes characteristics of the runs. The results for the computed cases differ as a result of:

1. the difference in map matching methods and parameters used and

2. the difference between the networks (it is suspected but not yet verified) that the number of links between two junctions that are each others neighbors in the road network, is larger for the Navteq network than for the OSM network (due to the presence of trivial nodes mentioned in section 5.1).

8. Conclusions and future research

A given path in a graph can be split into BPC (Basic Path Components) that are either least-cost paths or non-least-cost edges. We developed an algorithm that computes efficiently the size of a minimum splitting of a given path. The algorithm's correctness is proved. The analysis results in a set of *splitVertexSuites* for a given path from which minimum path splittings can be generated, and an upper bound for the number of possible minimum path splittings is obtained.

The new algorithm is used to verify the hypothesis that for utilitarian trips, people tend to compose their route from a small number of least cost paths. The feasibility to determine the size of a minimum path decomposition is shown. This size is a measure for the structural complexity of the path.

Region	Belgium	Belgium	Belgium	Belgium	Italy
Case	OSM	Navteq 1.0[min]	Navteq 2.5[min]	Navteq 5.0[min]	Navteq 1.0[min]
Runtime[sec]	5191	17058	25822	36424	498850
Machine	calc2	calc2	lucp2364	linux1	calc4
OS	Linux Debian wheezy	Linux Debian wheezy	Linux Debian wheezy	Linux Debian wheezy	Windows Server 2008
CPU Memory	Xeon 3[GB]	Xeon 3[GB]	i5 4[GB]	Core2 2[GB]	Xeon X5670 48[GB]
ClockFreq	2.8[GHz]	2.8[GHz]	2.4[Ghz]	2.4[GHz]	2.93[GHz]
Cores used	7	7	3	2	20
Trips scanned	6632	12429	9408	8020	34308
Trips dropped	694	2066	2426	2508	3427
Net number of trips	5938	10363	6982	5512	30881
Number of basic components	12687	22921	17429	14412	56346
Number of basic components per trip	2.14	2.21	2.50	2.61	1.82
Average number of nodes per trip	21.68	82.47	75.43	62.20	55.37
Number of least cost path calculations	128736	854637	526652	342846	1772569
Least cost calculations per second	24.8	50.10	20.40	9.41	3.55
Number of trips per second	1.14	0.61	0.27	0.15	0.06

Table 1: Run characteristics overview.

The resulting distributions for the number of BPC establish a criterion to assess the plausibility of a proposed route while generating choice sets in simulators. This criterion can be deployed (i) in the generation stage when using branch-and-bound techniques and (ii) in the assessment stage to filter inappropriate candidates when using other techniques. As such they provide a foundation to generate realistic routes in transportation simulation models.
The following questions need to be addressed in future research:

1. Determine an exact computation of the number of possible ways to partition a path into a minimum number of BPC.
2. Investigate more rigorously how the distribution for the minimum number of BPC in a path depends on the level of detail (coarseness) of the network and the map matcher used.
3. Design an efficient algorithm that generates routes from a given origin to a given destination for which the travel distance and the number of basic path components are values sampled from distributions determined from recorded trajectories.
4. Generate routes using a method such as the one described in Frejinger et al. (2009) in order to compare the resulting *splitVertexSuite* size distribution with the one extracted from GPS traces.

Acknowledgment

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 270833.

Octo Telematics, Italy (www.octotelematics.com) provided the set of trajectories for the Milano region.

References

Alvares, L.O., Bogorny, V., Kuijpers, B., de Macedo, J.A.F., Moelans, B., Vaisman, A., 2007. A model for enriching trajectories with semantic geographical information, in: Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems, ACM, New York, NY, USA. pp. 22:1–22:8.

Andrienko, G., Andrienko, N., Hurter, C., Rinzivillo, S., Wrobel, S., 2011. From Movement Tracks through Events to Places: Extracting and Characterizing Significant Places from Mobility Data, in: IEEE Conference on Visual Analytics Science and Technology, IEEE, Providence, Rhode Island, USA.

Balmer, M., Rieser, M., Meister, K., Charypar, D., Lefebvre, N., Nagel, K., 2009. MATSim-T: Architecture and Simulation Times., in: Multi-Agent Systems for Traffic and Transportation Engineering., Igi global edition. pp. 57–78.

Bekhor, S., Ben-Akiva, M., Ramming, M., 2006. Evaluation of choice set generation algorithms for route choice models. Annals of Operations Research 144, 235–247. 10.1007/s10479-006-0009-8.

Bovy, P.H.L., 2009. On Modelling Route Choice Sets in Transportation Networks: A Synthesis. Transport Reviews 29, 43–68,.

Dijkstra, E., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271.

Frejinger, E., Bierlaire, M., Ben-Akiva, M., 2009. Sampling of alternatives for route choice modeling. Transportation Research Part B: Methodological 43, 984 – 994.

Furletti, B., Cintia, P., Renso, C., Spinsanti, L., 2013. Inferring human activities from GPS tracks, in: UrbComp 13 Proceedings of the second ACM SIGKDD International Workshop on Urban Computing, ACM, Chicago.

Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D., 2007. Trajectory pattern mining, in: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA. pp. 330–339.

Kaplan, S., Prato, C.G., 2010. Joint modeling of constrained path enumeration and path choice behavior: a semi-compensatory approach, in: Proceedings of the European Transport Conference. Association for European Transport.

Knapen, L., Bellemans, T., Janssens, D., Wets, G., 2015. Efficient Offline Map Matching of GPS Recordings Using Global Trace Information. Unpublished results .

Kuijpers, B., Moelans, B., Othman, W., Vaisman, A., 2009. Analyzing Trajectories Using Uncertainty and Background Information, in: Mamoulis, N., Seidl, T., Pedersen, T., Torp, K., Assent, I. (Eds.), Advances in Spatial and Temporal Databases. Springer Berlin / Heidelberg. volume 5644 of *Lecture Notes in Computer Science*, pp. 135–152. 10.1007/978-3-642-02982-0_11.

Pillat, J., Mandir, E., Friedrich, M., 2011. Dynamic Choice Set Generation Based on Global Positioning System Trajectories and Stated Preference Data. Transportation Research Record 2231, 18–26.

Prato, C.G., 2009. Route choice modeling: past, present and future research directions. Journal of Choice Modelling 2, 65 – 100.

Prato, C.G., 2012. Meta-analysis of choice set generation effects on route choice model estimates and predictions. Transport 27, 286–298.

Prato, C.G., Bekhor, S., 2006. Applying Branch-and-Bound Technique to Route Choice Set Generation. Transportation Research Record , 19–28.

Prato, C.G., Bekhor, S., 2007. Modeling Route Choice Behavior: How Relevant Is the Composition of Choice Set? TRB Research Record 2003, 64–73.

Schüssler, N., Balmer, M., Axhausen, K.W., 2010. Route Choice Sets for Very High-Resolution Data, in: TRB 2010 Annual Meeting, TRB (Transportation Research Board), Washington, DC, USA. p. 16.

Spinsanti, L., Celli, F., Renso, C., 2010. Where you stop is who you are: understanding people's activities by places visited, in: Proceedings of the 5th BMI, Workshop on Behaviour Monitoring and Interpretation 2010, EU-FET Coordination Action MODAP, Karlsruhe. pp. 38–52.

Zheng, V.W., Zheng, Y., Yang, Q., 2009. Joint Learning User's Activities and Profiles from GPS Data, in: Proceedings of the 2009 International Workshop on Location Based Social Networks, ACM, Seattle WA.

Zijpp, N.J.v.d., Catalano, S.F., 2005. Path enumeration by finding the constrained K-shortest paths. Transportation Research Part B: Methodological 39, 545 – 563.