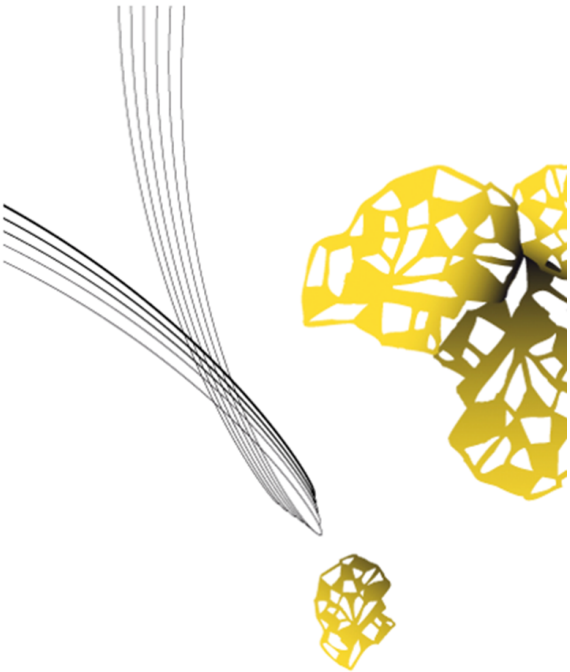


UNIVERSITY OF TWENTE.

Master Applied Mathematics

Multimodal map matching with smartphone data: a shortest path approach.

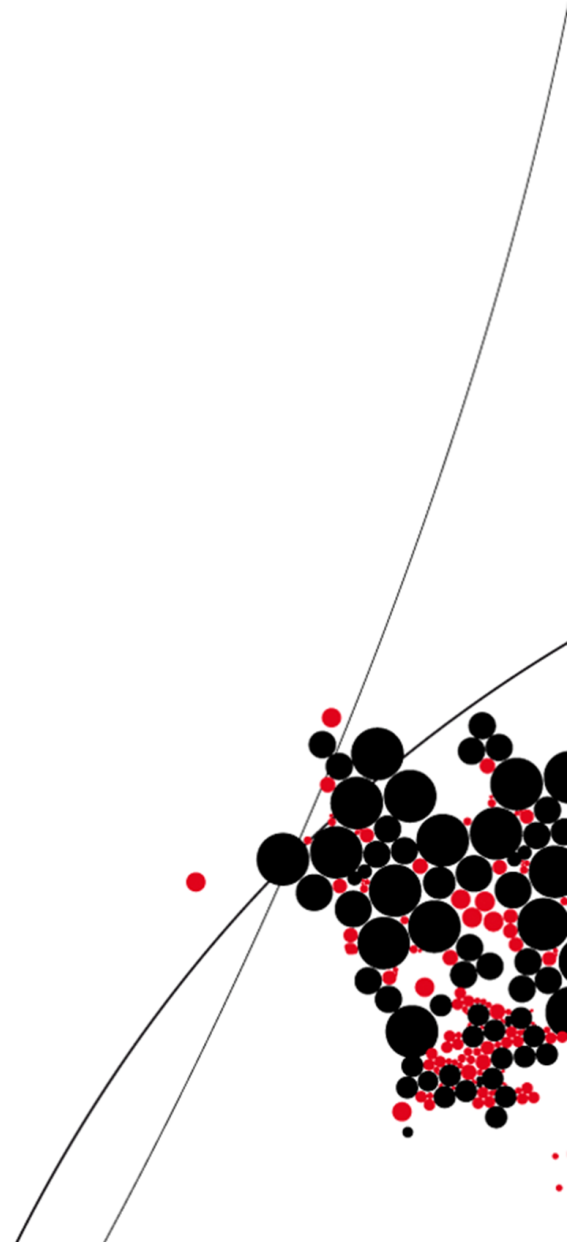
PUBLIC VERSION

J.M. Neeft

Final Project
May 2017

Supervisors:
Prof.dr. M. Uetz (UT)
Dr.ir. J.W. Koolwaaij (company)
Graduation Committee:
Prof.dr. M. Uetz (UT)
Dr.ir. J.W. Koolwaaij (company)
Dr. B. Manthey (UT)
Prof.dr. N.M. van Dijk (UT)

Chair Discrete Mathematics and
Mathematical Programming
Faculty of Electrical Engineering,
Mathematics and Computer Science



Disclaimer

Due to confidentiality, the full report has not been published. This is the non-confidential version. It is an attempt to publish the scientific relevant parts of the research, without publishing confidential information.

Roughly, the following information is included in the *confidential* version, but not in the *public* version:
- Preface
- Two problems in the Research Motivation (comparable with the two-step problem) and its case examples.
- Description of the benchmark algorithm
- Algorithmic details of the 4M algorithm
- Implementation of the 4M algorithm
- Practical issues

Abstract

Travel data has become too complex to be included in classic paper pencil questionnaires nowadays, as travellers combine different modes of transport for different purposes. Global Positioning System (GPS) technology in travel surveys has the potential to enhance the quality and scope of travel data collection. Embedded with various sensors, smart phones can be used for automatic trip and mode detection and have become a popular tool in studying mobility patterns and transportation network performance.

Saving smart phone battery consumption, the frequency of recording location measurements is limited. This study presents a method for solving the map matching problem for low frequency multimodal data. The map matching problem is to allocate a sequence of sensed location measurements onto a infrastructural map, determining the path that has been travelled. Many algorithms have been developed for high frequency data, however, the level of performance on smartphone data may be insufficient.

In this paper we propose an enhanced algorithmic shortest path approach to solve the map matching problem, called the MultiModal Map Matching (4M) algorithm. The main algorithmic contribution is the extension from a unimodal framework to a multimodal framework. The 4M algorithm exploits various type of data and combines both topological and probabilistic elements of existing algorithms. The developed 4M algorithm is tested using big sized real-world datasets and compared with a benchmark algorithm. Quality indicators are introduced to evaluate the performance of our algorithm. Results show an improvement in both quality and computational speed compared to the current state of the art.

Keywords: multimodal map matching, GPS, smartphone, low frequency, shortest path

Chapter 1

Introduction

The understanding of people's travel behaviour in many countries, including the Netherlands, has long been based on travel diary surveys [14][18]. Respondents were required to fill out many pages of travel diaries, covering only a few days. Quite often respondents missed short trips or reported them wrongly [8]. The demand for more precise and reliable spatial data can be derived from more detailed analysis and modelling. In addition to simple origin-destination coordinates, more complex data about the traversed path has been asked for. Travel data has become too complex to be included in classic paper pencil questionnaires nowadays, and most probably in the future too. Facing these problems researchers have been looking for alternative ways of collecting data.

Studies (e.g. [16], [18]) have shown that using, for example, Global Positioning System (GPS) technology in travel surveys has the potential to enhance the quality and scope of travel data collection. Embedded with various sensors, smart phones can be used for automatic trip and mode detection. Instead of relying merely on travel diaries and questionnaires, smart phones have become a popular tool in studying mobility patterns and transportation network performance.

Map matching

Core of the trip analysis is the map matching algorithm. The problem of a map matching is to locate a sequence of GPS coordinates onto a road map. In real-time applications, such as GPS navigation, map matching is an *online* problem. That is, location measurements have to mapped upon a road map as soon as they are available. The mapping can thus be based on past matches only. In contrast, post-processing applications use *offline* algorithms. These offline algorithms work with complete data only, i.e. all location measurements of a trip are known before processing the data. Offline algorithms can take the advantage of having information about 'future' measurements too.

online

offline

Quddus et al. [15] distinguishes between four algorithmic classes for solving the map matching problem:

- geometric
- topological

- probabilistic
- advanced

We note that multiple of these algorithmic ideas could be combined into one algorithm, thereby addressing multiple algorithmic questions. However, being an introduction into the topic, we discuss the algorithmic classes first in the same way as they appear in relevant literature.

geometric A geometric algorithm uses geometric information only, such as the shape of a road, to match a location measurement to a road segment. It does not consider the way which roads are connected to each other, i.e. the road is observed in isolation. Thereby, it does not guarentee to result in a meaningful path. Common geometric algorithms are search algorithms such as point-to-point matching, point-to-curve matching and curve-to-curve matching [21]. These approaches are easy to implement and fast, however, they are highly sensitive to outliers and low ranging in correct link identification [20].

topological Network topology refers to the spatial relation of objects in the network, such as, in our application, the connection between edges. A map matching algorithm that makes use of both the geometry of links and the topology is called a topological algorithm. An example of a topological algorithm is the algorithm by Marchal [10], which consists of two phases. First, the closest link is selected (geometric aspect). Second, the algorithm will continue mapping the points while taking the way in which edges are linked into consideration (topological aspect). The performance of topological algorithms is much better than geometrical algorithms, according to Velaga [19]. One of the primary features is that topological algorithms are only effective in high precision, high frequency measurement data [15].

probabilistic For a probabilistic algorithm, the definition of a confidence region around a location measurement is required. Zhao [23] suggests that this confidence region can be derived from the error variances of the location measurements. A recent implementation of a probabilistic algorithm is described by Chen and Bierlaire [5]. The result of their statistical method is a set of candidate paths, each with own probability of being the true one. Their algorithm works well in identifying the correct paths, but is computationally expensive.

advanced Lastly, there are all kind of advanced algorithms that use more advanced mathematical, statistical and artificial intelligence techniques such as Kalman filters, fuzzy logic and Bayesian inference. Performance evaluation and comparison has proved to be difficult, as there is no evident method of performance assessment.

Contribution

Most of the above algorithmic approaches for solving the map matching problem are designed for usage with high frequency positioning data (i.e. a sampling interval of 1s or less). Due to sensors low performance and pratical constraints (e.g. battery consumption), low frequency smartphone data is usally sparse and inaccurate [3]. In case of low frequency data, two important factors cannot be derived anymore: direct road connectivity and turn restrictions. Algorithms developed for high frequency data may therefore not be suitable to correctly match low frequency data [20]. Few researchers propose a shortest path approach when for some distance no data is known [20][22].

Most of previous research focussed on developing general algorithms for synthetic networks, assuming rational human behaviour and uni modal trips. However, human behaviour is sometimes erratic and travellers combine different modes of transport to reach their destination. Therefore, it is important to develop an algorithm that is capable of dealing with both human irrational behaviour and multi modality.

The methodology in this work follows Chen [5] in formulating a multimodal network. Then, partly based on the idea of the unimodal weighted shortest path approach of Quddus [20], a multimodal map matching algorithm using a shortest path approach is developed. Several characteristics of the just mentioned four different algorithmic classes are integrated in our approach, such as the connectivity of topological algorithms and the use of error regions in probabilistic algorithms.

Chapter 2

Problem description

This chapter starts by introducing the terminology that is used in this report and formulating the map matching problem in general terms. In the following sections, the research motivation and goals are discussed. The (rather extensive) section about the benchmark algorithm is omitted due to confidentiality.

2.1 Terminology

Throughout this report many abbreviations and definitions are used that might not be obvious. Definitions of most term are completely or partialy derived from [11], [13]. Also, terminology used by the company is adopted. In this section, the most important definitions are outlined. A list of all mathematical notation can be found in Appendix A.

Modality: Mode of transport. Among others, the following modalities are distinguished: foot, cycle, car, train, bus, metro, tram, boat, plane.

Trip: Journey in one direction from origin to destination using one or more transport modes.

Trip leg: Segment of a trip, which is separated by a transport mode change or a short intervening stop (e.g. coffee stop, public transport transfer, post delivery).

Tour: Total travel, including both directions of travel, such as home-work-home.

Multimodal: More than one modality.

Activity: Purpose of a trip. Working, shopping and leisure are common activities.

Location: Place on earth, represented by two coordinates. The first coordinate represents the latitude: the angle between the equatorial plane and the straight line passing through the point and the centre of earth. The second coordinate represents the longitude: the angle west or east of the prime meridian to a meridian passing through the point.

Vertex: Point in a graph.

Edge: Line segment.

Ground truth: Actual location(s), opposed to the estimated location(s).

Accuracy: Maximum distance between the estimated location and the ground truth.

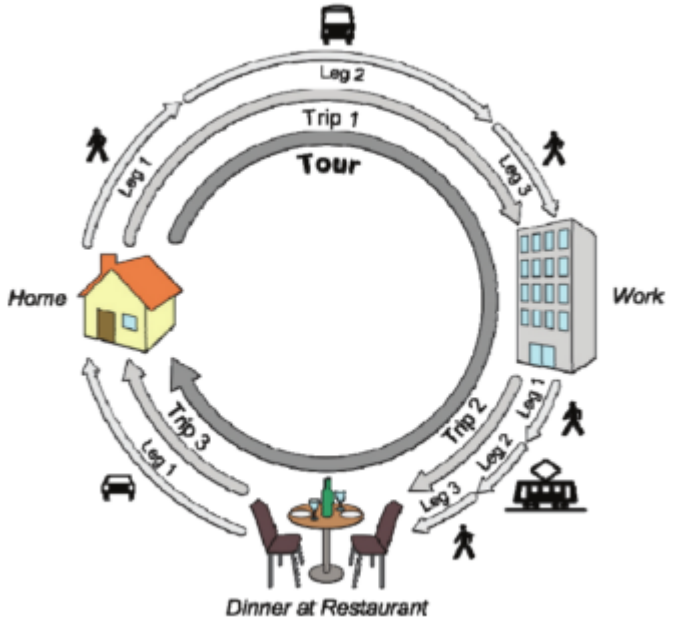


Figure 2.1: Represenstation of Trip leg, Trip and Tour [13]

2.2 Map Matching Problems

In this section two variants of the map matching problems are introduced: the *single map matching problem* (SMMP) and the *trip map matching problem* (TMMP).

vertex
edge
path
location

Consider a person or vehicle moving along a finite set of streets. The set of streets is not exactly known. Instead, it is represented by a directed graph G, which is composed of a finite set of egdes $E(G)$ and a finite set of vertices $V(G)$. We refer to a graph with vertex set $V = V(G)$ and edge set $E = E(G)$ as $G = (V, E)$. An edge $e \in E$ can be described by a finite sequence of n_e points $(e^0, e^1, ...e^{n_e})$, where each point is in $\mathbb{R}^2$. The first point e^0 and the last point e^{n_E} in the sequence are referred to as *vertices*, while all intermediate points are called *shape points*. Basically, a vertex is a point where the edge begins or terminates, or a point where it is possible to move from one edge to another (intersection). A vertex $v \in V$ is characterised by its latitude v^{lat} and longitude v^{lon}, determining the geographical location of the vertex. Each edge $e \in E$ is associated with a pair (v_i, v_j) of vertices and represents a road segment or rail track segment from vertex $v_i \in V$ to $v_j \in V$ $(v_i \neq v_j)$. An ordered list of connected vertices is called a path, denoted by $p = (v_0, v_1, ..., v_n)$. At a finite number T of points in time, we are provided with an estimate of the current location. At time t this estimate is denoted by $\bar{x}_t$, whereas the ground truth is x_t.

The goal in the SMMP is to determine the street that the person is on at time t. That is, match the estimated position $\bar{x}_t$ with an edge $e \in E$. The match is correct if e is the

ground truth edge e^{gt}, assuming that all ground truth edges are part of G.

ground thruth path

For the TMMP , we are give a set of location estimates, that correspond to a trip. All locations estimates, denoted by $\bar{X} = (\bar{x_1}, ..., \bar{x_T})$, have to be mapped at once. The goal is then to find the path p in G that equals the ground truth path, denoted by p^{gt}. For simplicity, we will refer to the TMMP as the *map matching problem (MMP)* throughout this report.

2.3 Research Motivation

two-step

Traditionally, transport mode inference and map matching in a multimodal setting consist of two steps [17]:

1. Split the data into multiple unimodal segments
2. Map matching and mode inference for each segment independently

However, the two step-approach poses a high risk of wrong results, for example due the fact that incorrect segmentation in the first phase may occur, which is not recoverable anymore [5]. We formulate the two-step problem as follows:

two step problem

Definition 2.3.1. *The two step problem is incorrect splitting in the first phase of a two step algorithm, which is not recoverable anymore.*

To further illustrate the two-step problem, a example will be given.

Example 2.3.1. ***Two-step problem***
This example demonstrates the risk of the two-step procedure as described at the beginning of this section. Consider a multi modal trip, consisting of a short walking trip lag and, subsequently, a longer train trip leg. This trip should be split at the train station. For this particular instance, it is stressed that the location measurements are highly accurate. Looking at the map, obviously, the map matching in purple is not equal to the ground truth. The trip is wrongly considered as unimodal by the algorithm, resulting in a train trip only. The walking trip leg is not represented in the result, because the algorithm fails to split the trip in two unimodal legs. Then, the algorithm tries to minimize the costs, utilizing railway only.

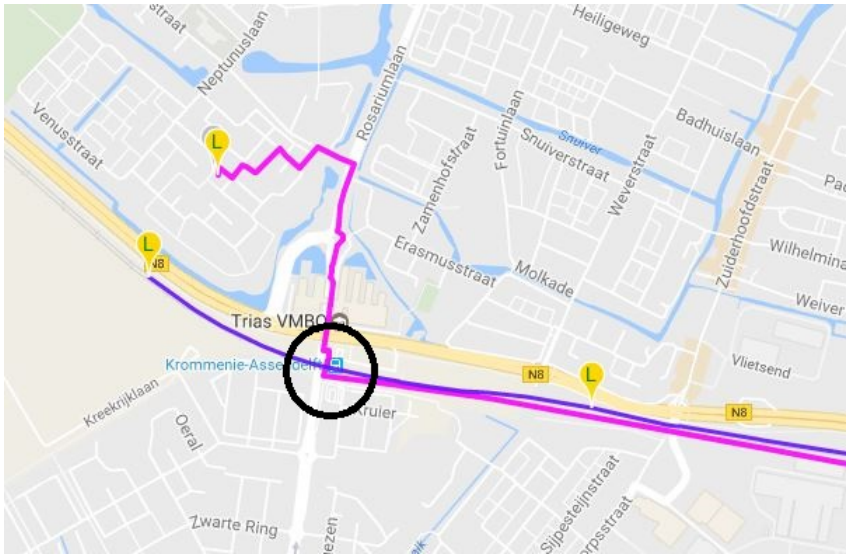


Figure 2.2: Illustration of Example 2.3.1, a segment of p in purple and $\bar{X}$ in pink

The general motivation for further investigation of the map matching algorithm is both computational speed and output quality. The ultimate goal is to find the ground truth p^{gt}, minimizing the number of location measurements needed and limiting computational time. There is a trade-off between number of measurements and quality of the map matching. Increasing the number of measurements will likely increase the quality of the mapped path, however, in the smart phone context, this consumes a lot of battery.

real life data

Little research has been done so far using real life data. Most developed algorithms have been tested upon artificially created data sets. However, these data sets do not deal with the irrational and unpredictable movements that human beings often make. Moreover, artificial data sets also abstract from the fact that sometimes GPS fails to measure the movements accurately.

2.4 Benchmark algorithm (BM)

As a benchmark map matching algorithm, denoted by BM, we take the algorithm that is currently in use by the company. This is a shortest path based algorithm. More details are omitted due to confidentiality.

2.5 Research Goal

Section 2.3 states that for finding the best map matching result, correctness and computational speed might be conflicting objectives. Keeping in mind data preprocessing and modality recognition are outside the scope of this research, the main goal of this research is as follows:

Improve the performance of the map matching algorithm used by the company by targeting both the quality of the map matching path and the computational speed of the algorithm.

We consider and divide improvements in two categories: speed improvements and quality improvements:

quality
- Quality improvements. We opt to improve the map matching result by both preprocessing the graph G in a preprocessing phase and improvements in the algorithmic design.

speed
- Speed improvements. These improvement try to find solutions faster, while preserving the quality of the solution.

Chapter 3

Multimodal algorithm

In our approach we focus on the multimodal nature of a trip as a point of departure. We want to overcome the risk of incorrect segmentation in the first step of the two-step approach, as incorrect segmentation is not recoverable anymore. In the multimodal approach, we no longer want to split the trip before running the map matching algorithm, but integrate the split decision in the map matching algorithm. This means that the trip leg identification is based upon the input graph, whereas in the benchmark approach the input graph is based upon the trip leg identification. The map matched path does not consist of disconnected trip legs anymore, but the result is one (multimodal) path consisting of one or several legs that connects the origin with the destination.

integrate split decision

3.1 Multimodal formulation

We assume that every vertex can be traversed by one specific modality only. Let v_i^m be the modality associated with vertex v_i. Let G^m be a unimodal transport graph that contains edges $(v_i, v_j) \in E(G^m)$ with $v_i^m = v_j^m \quad \forall(i,j)$. That is, all vertices can be traversed by the one modality only. A multimodal transport graph, however, is represented by a union of unimodal transport graphs with artificial edges connecting them.

multimodal path

A multimodal path p is a path in a multimodal graph. We will refer to multimodal path as 'path' throughout the rest of this report. This path may contain one mode or several modalities. A path without any mode information is referred to as 'physical path'. Figure 3.1 gives an example of a multimodal graph and a multimodal path. An artificial edge (v_i, v_j) connects vertex v_i with the vertex v_j where both have the same location $(v_i^{lat}, v_i^{lon}) = (v_j^{lat}, v_j^{lon})$, but different modality $v_i^m \neq v_j^m$. The artificial edge illustrates a change of transport mode and connects two vertices that have the same geographical location, but different modalities. In all drawings, all edges are represented as bidirectional, for the sake of drawing clarity. Moreover, for the sake of simplicity, each edge is a represented by straight line. Dashed lines represent artificial edges that connect two unimodal networks.

artificial edge

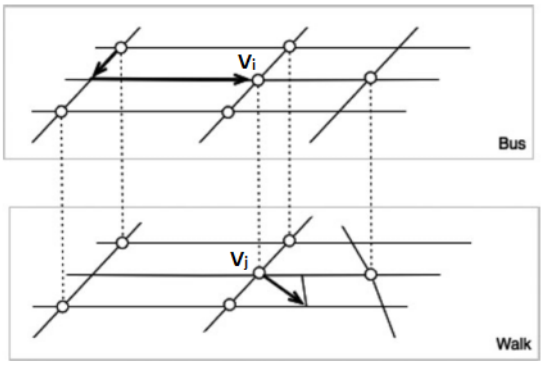


Figure 3.1: Multimodal graph and multimodal path

measurement set

Let us assume that we have access to all preprocessed measurements recorded during a trip. We store them in the set $\bar{Y} = (\bar{y}_1, ..., \bar{y}_T)$, where T is the total number of measurements. Each measurement $\bar{y}_i = (\bar{x}_i, \bar{v}_i, \bar{t}_i)$ constitutes of several components. $\bar{t}_i$ is the measurement time, in seconds, $\bar{x}_i = (\bar{x}_i^{lat}, \bar{x}_i^{lon})$ a pair of geographical coordinates; $\bar{s}_i$ is the measured speed in m/s. Moreover, we introduce

Note the difference between $\bar{X}$ and $\bar{Y}$. All components are stored in $\bar{Y}$, such as speed, time and location, $\bar{X}$ contains locations measurements only.

A path $p = (p_1, ..., p_s)$ in G consists of s chronologically ordered trip legs p_i, $i = 1, ..., s$. A trip leg p_i has one unique modality p_i^m, which means that all vertices contained in the trip leg share the same modality. Let $G^m \subseteq G$ be the subgraph induced by vertices having modality m. A trip leg is a sequence of n connected vertices $(v_k, v_{k+1}, ..., v_{k+n-1}, v_{k+n})$ such that there is an edge e from v_k to v_{k+1} in G^m. Trip leg i and $i+1$, $1 \leq i \leq s-1$, are connected by an artificial edge, illustrating the mode change.

shortest path

The multimodal map matching problem can be modelled as a shortest path problem from origin vertex $o \in V$ to destination vertex $d \in V$. Let $c_{i,j}$ be the cost of using edge (i, j). Now, the shortest path is the path $p = (v_1, v_2, ..., v_n)$ (where $v_1 = o$ and $v_n = d$) that minimizes $\sum_{i=1}^{n-1} c_{i,j}$.

Note that this would in principle require us to know the modalities for the origin vertex o and the destination vertex d. Let us for this moment assume that we have the right classification for these vertices.

3.2 ILP-formulation

The MMP can be formulated as an Integer Linear Problem (ILP). This formulation is very compact and can therefore be helpful to get an overview of the problem at hand.

Let $z_{i,j}$ $(i \neq j)$ be a decision variable with $z_{i,j} = 1$ if edge (i, j) is selected and $z_{i,j} = 0$ otherwise. The cost function $c_{i,j}$ is non-negative for all i, j.

The ILP-formulation of the MMP is as follows:

$$\min \sum_{(i,j)\in E} c_{i,j} z_{i,j} \tag{3.1}$$

s.t.

$$\sum_{j\in V} z_{o,j} = 1 \tag{3.2}$$

$$\sum_{j\in V} z_{j,d} = 1 \tag{3.3}$$

$$\sum_{j\in V} z_{i,j} = \sum_{j\in V} z_{j,i} \quad \forall i \in V \tag{3.4}$$

$$z_{i,j} = \{0,1\} \quad \forall (i,j) \in E \tag{3.5}$$

where (3.1) is the objective function, minimizing the total cost (which will be defined later). (3.2) forces selecting one starting edge, constraint (3.3) ensures choosing one last edge and (3.4) is flow conservation. This formulation is a well-known flow-based formulation for the shortest path problem.

Note that the same formulation could be used in a two-step approach. Then, for each trip leg, a shortest path problem has to be solved. However, the novelty here lies in the fact that we use a multimodal network, which allows us to solve one shortest path problem only.

3.3 4M algorithm

Confidential

Chapter 4

Speedup techniques

Given a weighted directed graph and a source and target vertex, Dijkstra's algorithm finds the shortest paht from source to target . Its performance is dependent on both the number of nodes and arcs; the algorithm run in $O(nlogn + m)$[7] time. One way of improving the efficiency of Dijkstra's algorithm is to run a bidirectional search. Moreover, preprocessing the graph, such as exploiting hierarchy, can lead to time savings. In this section two speed improvement operations are described: a bidirectional variant of Dijkstra's algorithm and core node identification. These speedup techniques have been implemented and tested upon a case trip to regard if they could lead to an improvement in computational speed.

4.1 Bidirectional

The bidirectional variant of Dijkstra's algorithm runs two Dijkstra's algorithm simultaneously. The *forward* algorithm starts its search from the source, whereas the *backward* algorithm starts searching from the target (i.e. with t as source and s as target). This variant of Dijkstra's algorithms was first widely published by Dantzig [6]:

> "If the problem is to determine the shortest path from a given origin to a given terminal, the number of comparisons can often be reduced in practice by fanning out from both the origin and the terminal, as if they were two independent problems."

Algorithm 1 describes the bidirectional Dijkstra algorithm in pseudocode. The forward algorithm begins at s and the backward algorithm starts at t. At each iteration (line 3) both the forward and backward algorithm select one vertex, based on their distance labels $d_s(v)$ and $d_t(y)$. The shortest path length between s and t so far is maintained by $\mu, \mu = \infty$ initially. Whenever an arc (v, w) is scanned by the forward algorithm and w has already been selected by the backward algorithm, we know that we found a candidate shortest path from s to t via (v, w). The **relax**(v, w) operation refers to updating the distance label for vertex w. If this path is the shortest path found so far, μ is updated

accordingly (line 13). Similarly, the same procedure is run for the backward search (line 20). The algorithm terminates upon selecting a vertex that has already been selected by the algorithm in the reverse direction. We go into detail the details of the algorithms, including a proof and example, because earlier work (e.g.[2]) showed that finding the right stopping criterion is not straightforward.

Algorithm 1: bdDijkstra(G, c, s, t)

input : Undirected Graph $G = (V, E)$, nonnegative cost function $c : E \to \mathbb{R}$, sourcenode $S \in V$, targetnode $t \in V$
output: Path P, total cost C

1 *Initialise:* $d_s(s) = 0, d_s(v) = \infty$ for every $v \in V \setminus \{s\}$ and $d_t(t) = 0, d_t(v) = \infty$ for every $v \in V \setminus \{t\}$, $\mu = \infty$;
2 $F^* = V$ and $B^* = V$;
3 **while** $F^* \neq \emptyset$ *and* $B^* \neq \emptyset$ **do**
4 Choose a node $v \in F^*$ with $d(v)$ minimum;
5 Choose a node $y \in B^*$ with $d(y)$ minimum;
6 **if** $y \notin F^*$ *or* $v \notin B^*$ **then**
7 **break**; `// vertex already scanned`
8 **end**
9 Remove v from F^*;
10 **for** $(v, w) \in E$ **do**
11 **relax**(v, w);
12 **if** $w \notin B^*$ **then**
13 $\mu = max\Big(\mu, d_s(v) + c(v, w) + d_t(w)\Big)$; `// update shortest path cost`
14 **end**
15 **end**
16 Remove y from B^*;
17 **for** $(y, z) \in E$ **do**
18 **relax**(y, z);
19 **if** $v \notin F^*$ **then**
20 $\mu = max\Big(\mu, d_s(y) + c(y, z) + d_t(z)\Big)$; `// update shortest path cost`
21 **end**
22 **end**
23 **end**
24 **return** μ

To prove correctness of the bidirectional algorithm we need two lemma's on Dijkstra's algorithm. Let $\delta(v)$ be the exact distance label of vertex v. The lemma's are stated here without proof:

Lemma 4.1.1. *Whenever a vertex v is selected by Dijkstra's algorithm, we have $d_s(v) = \delta_s(v)$.*

Lemma 4.1.2. *Dijkstra's algorithms selects vertices in nondecreasing order of their distances from the starting (terminal) vertex s*

After noting that the algorithm does not necessarily terminates on a vertex which is on the shortest path, we now prove correctness.

Theorem 4.1.3. Algorithm 1 finds the length of the shortest path between source vertex s and target vertex t

Proof
Let v be the vertex on which the algorithm terminates (line 7 of Algorithm 1).Then, v has been selected by both the forward and backward search. Now, by Lemma 4.1.1, $d_s(v) = \delta_s(v)$ and $d_t(v) = \delta_t(v)$. Therefore, the length of the shortest path from s to t through v equals $d_s(v) + d_t(v)$.

We now show by contradiction that the length of any path from s to t containing vertices that have been unselected by the forward or backward search must be bigger than $d_s(v) + d_t(v)$. Assume that there is a path from s to t through an unselected vertex w that is shorter than any path using seen vertices only. Then, the path through w should be shorter than the path through v, so

$$\delta_s(w) + \delta_t(w) < \delta_s(v) + \delta_t(v)$$

From this, we conclude that either $\delta_s(w) < \delta_s(v)$ or $\delta_t(w) < \delta_t(v)$ (or both). However, as we assumed that w was not visited upon termination, this contradicts with the fact that Dijkstra's algorithm selects vertices in non-decreasing order of distance from the start vertex. Thus, the shortest path contains only edges that have been visited upon termination. □

Note that formulating the correct stopping criterion is crucial. Berge and Ghoulia-Houri [2] falsely stated that termination should be upon permanently labelling a node, for the first time, from both search directions. The example in Figure 4.1 shows that this can lead to a path that is not the shortest: after two iterations of the bidirectional algorithm, nodes 3 and 4 have their exact distance labels $d_s(3) = d_t(4) = 2$ and $d_t(3) = d_s(4) = 6$ and $\mu = 8$. However, s-3-4-t is not the shortest path from s to t. For correctness, the algorithm should, first properly described by Nicholson [12], continue until the intersection of seen vertices from both the forward and backward search is not empty anymore (which is not the case after 2 interations).

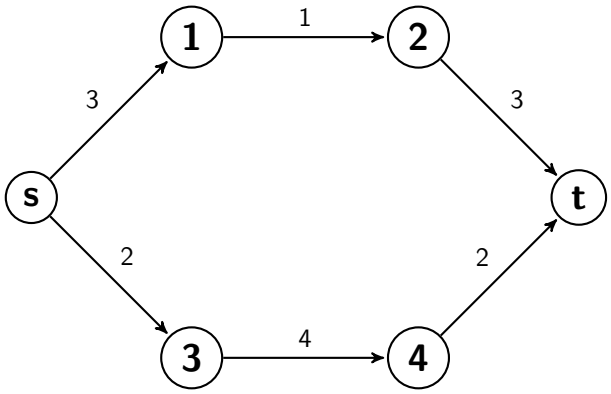


Figure 4.1: Example of trip which stresses the importance of a good stopping criterion for the bidirectional Dijkstra algorithm.

4.2 Core node identification

As have been noted before, the performance of the shortest path algorithm is dependent on both the number of edges and nodes of the input graph. A reduction of the size of the input graph $\bar{G}$ could therefore lead to a reduction in computation time. To reduce the graph size, we suggest to exclude edges and nodes from $\bar{G}$ that are said to be part of a *dead end*. These edges and nodes are located such that once included in a path from source to target, there is no opportunity to get to the destination except traversing the same edges in the opposite direction. We will identify dead ends by using the notion of the *k-core*.

First, as an example, consider Figure 4.2. The input graph $\bar{G}$ for the shortest path algorithm that is printed contains six nodes. The shortest path from s to t is obviously via $(s, 1)$ and $(1, t)$, having a cost of 3. Edges $(s, 3)$ and $(s, 4)$ are said to be part of a dead end, because they have no change of being in the shortest path. However, they will be visited by Dijkstra's algorithm in the first iteration, thereby taking unnecessarily computational time. In this section, we will come up with a technique to delete the edges $(s, 3)$ and $(s, 4)$ and nodes 3 and 4.

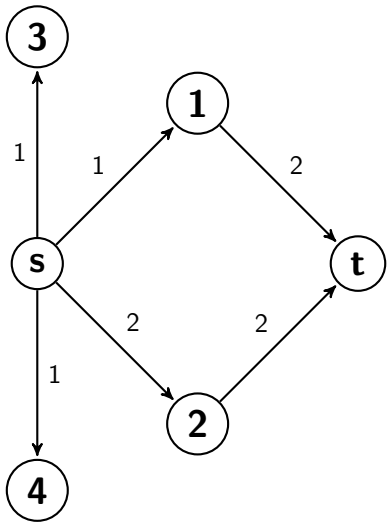


Figure 4.2: $\bar{G}$ with two *dead ends*: $(s, 3)$ and $(s, 4)$

To find the *dead ends* of a graph, let us define the concept of a *k-core*. Let H be a subgraph
degree of $\bar{G}$ and $deg_H(v)$ be the *degree* of vertex v in graph H. The degree of a vertex is the number of edges attached to it.

k-core **Definition 4.2.1.** *H is a **k-core** of $\bar{G}$ iff the minimum degree of all edges in H, denoted by $\Delta(H)$, is at least k, and H is a maximal subgraph of $\bar{G}$.*

The core-decomposition algorithm presented here is based on the following property [1]: Delete all vertices, and lines incident to them, of degree less than k, from a given graph. The remaining graph is the k-core.

Based on this property, the algorithms to identify the k-core is as follows:

Algorithm 2: kCore($\bar{G}, k$)

input : Undirected Graph $\bar{G} = (\bar{V}, \bar{E})$, represented by a list of neighbours
output: Subgraph $H, H \subseteq \bar{G}$

```
1  H = Ḡ;
2  Compute deg(v) for every v ∈ V̄;
3  Order V̄ in non-decreasing degree order;
4  for v ∈ V̄ do
5  |  core(v) := deg(v);
6  |  if core(v) ≥ k then
7  |  |  break
8  |  end
9  |  for u ∈ Neighbour(v) do
10 |  |  deg(u) := deg(u) − 1;
11 |  |  reorder V accordingly
12 |  end
13 end
14 Remove all u and (u, v) with core(u) = 1 from H ;        // remove dead-ends
15 return H
```

We refer to [1] for a full description of the k-core algorithm that runs in $O(m)$.

The k-core algorithm is applied to remove *dead ends* from $\bar{G}$ using the concept of the 2-core:

dead ends

Definition 4.2.2. *The dead ends of a graph $\bar{G}$ are those vertices and edges that are not in the 2-core.*

By deleting all dead ends from the input graph we expect to be able to speed up Dijkstra algorithm.

Chapter 5

Experimental Design & Datasets

In this chapter we describe how we tested the proposed algorithms. In the first section, we describe how the 4M algorithm has been implemented. Next, the constructed datasets, consisting of personal data and anonymous data, can be found in the second section. Then, the way in which the experiments are set up is explained is discussed. Lastly, different quality indicators are reported to evaluate the quality of a path.

5.1 Experimental Environment

For execution of the experiments, we have extended and altered the already existing analysis environment in Python and C#, designed by the company. All new solution methods have been incorporated directly into this environment, such that these methods could be tested on real instances from the trip database and, moreover, the results could be compared with results from previous methods. We first describe how we implemented the 4M algorithm.

Phases of 4M

The 4M algorithm distinguish four phases:
1. Data preprocessing phase
2. Graph construction phase
3. Matching phase
4. Evalution phase

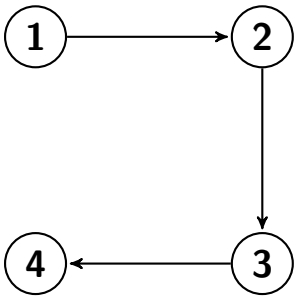


Figure 5.1: Phases of 4M (schematic)

Data preprocessing phase

In this phase outliers are detected, inaccurate data is deleted and locations of missing measurements are estimated.

Graph construction phase

The input is as follows:

- The initial graph G is taken from *OSM* and contains all vertices and edges located in the Netherlands. Trips that have location measurements outside the Netherlands are mapped onto the Planet database, consisting of all vertices and edges worldwide.
- The distance parameter r_i determines the size of the ball around a location measurement, from which the nodes and edges of G are taken, let this be $\bar{G}$. To take into account the inaccurate nature of location measurements and the low density of measurements, the distance parameter r_i is dependent on both the distance between the consecutive measurements and the accuracy of the measurement:

$$r_i = \min\left(1000, \max\left(\frac{||\bar{x}_i - \bar{x}_{i-1}||}{2}, acc(\bar{x}_i), 50\right)\right) \tag{5.1}$$

  The distance parameter determines the size of the subgraph $\bar{G}$. The accuracy of $\bar{x}_i$ is denoted by $acc(\bar{x}_i)$. Again, the values in equation (5.1) are set by experience.

Matching phase

In the matching phase the shortest path algorithm is executed on the subgraph $\bar{G}$ constructed in the graph construction phase. The pgRouting library (http://pgrouting.org) extends the PostgreSQL geospatial database to provide geospatial routing functionality. The pgRouting library contains well-known implementation of routing algorithms, such as Dijkstra's shortest path algorithm and A* shortest path. The Bidirectional algorithmic variants included in the pgRouting library proved to be faulty and slow, therefore, they have not been applied. Instead, the bidirectional variant of Dijkstra's algorithm has been implemented using the Networkx package (http://networkx.github.io) from the Python library.

Quality

The quality of the map matched path is evaluated using quality indicators, described in Section 5.4.

5.2 Datasets

This section discusses the data that is used in the coming experiments. The collected data can be classified into two categories: personal data and anonymous data. Personal data consists of all measurements collected by the researcher personally, whereas the anonymous data is provided by other users that collect data with one of the mobile phone apps of the company.

5.2.1 Personal Data

All personal data has been collected between October 7th 2016 and March 22nd 2017 by the researcher personally. For this purpose the a suitable App for Android devices has been used. All trips that have been made within the time span, except occasional failures due to insufficient battery capacity, have been recorded and uploaded to the company's database. The resulting collection of trips is unique in its size, time span and detail in this field of research. We base our experiments on this data. The data is graphically represented by a heatmap in Figure 5.2.

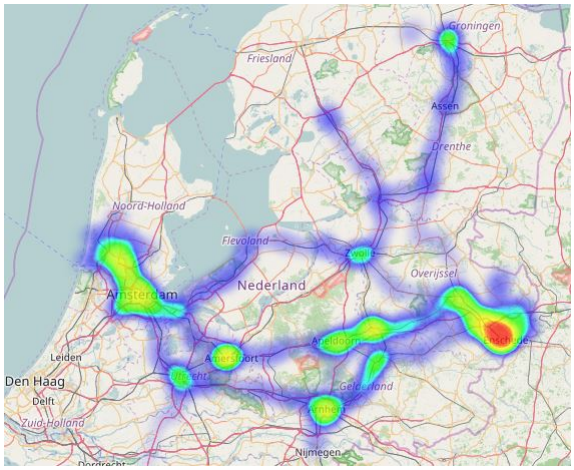


Figure 5.2: Heatmap of all personal trips

A summary of the statistics is presented in Table 5.1. The table includes the total number of trips and trip legs recorded and both the average length and duration of a trip leg. Standard errors are deported in brackets.

During the recorded period, 790 trips have been made, consisting of 1016 trip legs. Out of these 790 trips, 140 can be classified as multimodal. The main characteristic of the travel pattern is the share of different modalities, which is presented in Table 5.2. The majority of the trip legs is made by bike (61%) followed by foot (17.2%), public transport (15%)

and car (6%). Looking at the distance share by mode, we note that public transport travel is dominating, with more than 66% of the travelled kilometres. The mode share has been calculated for different ranges of travel, namely short distance (up to 1500m), medium distance (from 1500m to 10k) and long distance (above 10k). Looking at the split of mode share by range of travel shows the complementary of modes between travel distances, with walking and biking dominating the short distances and public transport and car travel dominating the longer distances.

The statistics for the share of different modalities in the personal data set differs highly from those collected by the *Kennisinstituut voor Mobiliteit*, both regarding the distance travelled en the number of times travelled. The national Duth 2016 study [4] reports that 73 % of all travelled kilometres in the Netherlands is by car, 12 % by public transport, 9 % by bike and 3% by foot. Of the number of trips made, 46 % is by car, 28 % by bike, 18 % by foot and just 4 % by public transport.

Sensed data: 2016-10-07 till 2017-03-22	
Number of trips	790
- unimodal	650
- multimodal	140
Number of trip legs	1016
Average length	8.50 km [2.38]
Average trip duration	18m53 [50m15]

Table 5.1: Data description (standard error in [.])

	Foot	Bike	Car	Public	Other
Leg	17.2	61.2	6	15	0.5
-short	26.5	68.3	3.7	0.7	0.7
-middle	6.5	70.1	4.8	18.6	0
-long	0	13.6	17.8	67.8	0.9
Total distance	1.9	15.7	15.5	66.3	0.6
-short	22	73.1	3.9	0.8	0.1
-middle	3.5	63.5	5.4	27.6	0
-long	0	5.9	14.9	78.1	1.2

Table 5.2: Mode share of travelled distance and number of trips leg, splitted by range of distance

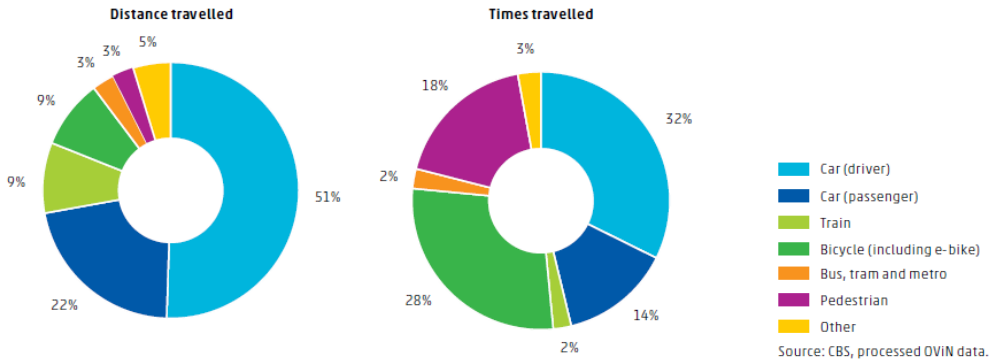


Figure 5.3: Mobility of Dutch people by means of transport, 2016 [4]

Figure 5.4 shows a boxplot for the length of a trip leg in kilometres for the four main transport types. Each box contains a straight line at the median and the lower and upper quartile of the trip leg length. The whiskers show the variability outside the quartiles. Points past the end of the whiskers are outliers (these are mostly just correctly measured trips that highly deviate with respect to distance from the average).

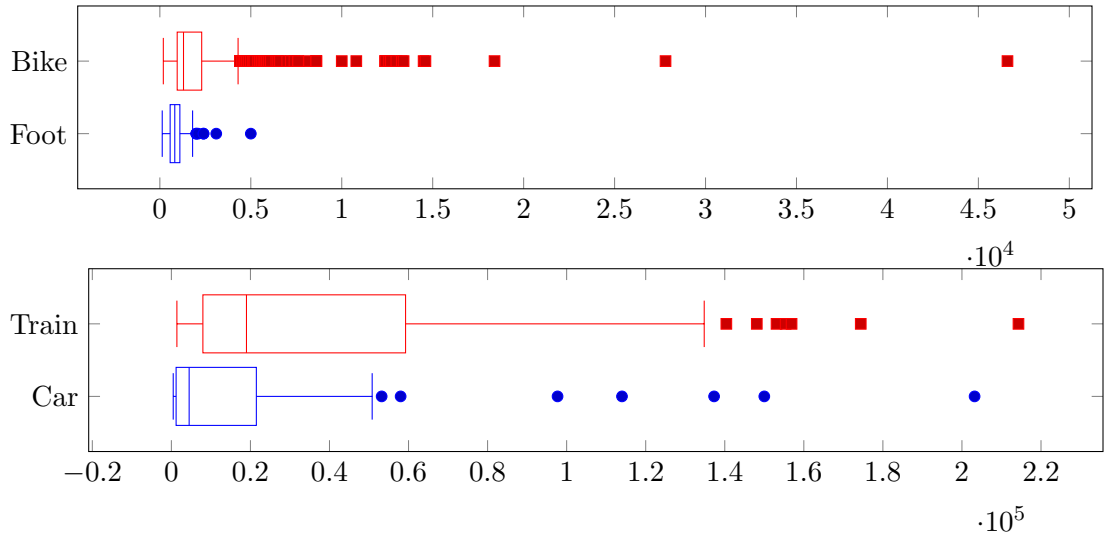


Figure 5.4: Boxplot of the length of personal trips in kilometres for different modalities

5.2.2 Anonymous data

The anonymous dataset consists of trips that have collected by smartphone users in the city of Enschede and around. The set contains trips that have been made between Thursday March 23th 2017 18:00 and Monday March 27th 2017 09:00. It covers both a weekday (Friday) and weekenddays (Saturday and Sunday). This means that both home-to-work trip, mostly made on workdays, and recreational trips, mostly made on weekenddays, are included. Thereby, we ensure that a big variation in travel behaviour is encountered while

analysing the dataset.

The dataset consists of 3312 trips, of which 692 are multimodal. The full data description can be found in Table 5.3.

Data Enschede: 2017-03-23 18:00 till 2017-03-27 09:00	
Number of trips	3312
- unimodal	2618
- multimodal	692
Number of trip legs	4225
Average length	10.6 km [25.78]
Average trip duration	18m64 [26m59]

Table 5.3: Data description Enschede dataset (standard error in [.])

Regarding the share of different modalities, the anonymous data set is similar to to the national Dutch statistics. The percentage of trip legs made by different means of transport in the anonymous data set does not differ much from the national numbers. A comparison between the anonymous data set and the national statistics is made in Figure 5.5.

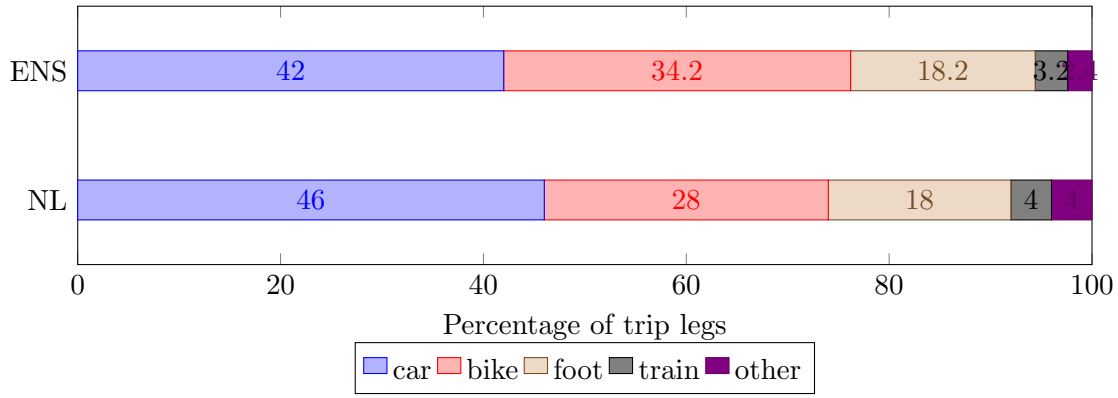


Figure 5.5: Percentage of trip legs by modality for anonymous dataset

5.3 Setup of Experiments

Four different experiments have been designed. These experiments are to evaluate the performance of the solution methods.

Case experiment

The first experiment considers the two step problem. To examine if this problem can be solved by applying the 4M algorithm, we execute our approach on the example from 2.3.1. In this experiment we can evaluate the effect of our solution approach on a individual trip by comparing it with the benchmark.

Validation experiment

Next to individual, pre-selected, trips, we are also interested in improving the results on an aggregated scale. This second validating experiment examines the effect of executing the 4M algorithm on a set of trips. This set of trips is a subset of the personal tripset, as described in Section 5.2.1. The subset consists of 56 trips out of 790 trips, which are selected such that the set contains a big variety of trips, both in length, modal choice and accuracy. The main advantage of this set, compared to non-personal data, is that the true path can be reconstructed. This gives the opportunity to quantify the quality of the result, by comparing the resulting path with the true path. A subset is taken because of the time-consuming process of reconstructing the true path (which was limited in this project).

We will first execute both the benchmark approach and the 4M solution approach on the selected set, after which we can compare the results on both quality indicators and computational speed. Then, we try to find indicators that correlate with the quality of the mapped paths, such that we can use these indicators to quantify the quality of mapped paths for other sets (of which we do not know the ground truth).

Figure 5.6: Map of trips included in the dataset for the validation experiment

Anonymous data experiment

In this experiment, the 4M algorithm has been used for solving the map matching problem for trips of the anonymous data set from Section 5.2.2. This data set contains only trips of which the ground truth is not known. Therefore, the quality indicators, that have been validated in the validation experiment, are used to evaluate the outcome. Unfortunately, due to practical issues, it was not possible to compare the benchmark and 4M algorithms by analysing the same dataset with both approaches.

Speedup experiment

This last experiment is to show whether a speed up can be achieved if a bidirectional variant of Dijkstra's algorithm and/or core node identification to remove dead-end structures would be implemented. We were able to test both methods on one instance and report the results in the next chapter. These results are indicative and to draw any solid conclusions,

more testing is needed.

5.4 Quality Indicators

As the ground truth (the path traversed) is mostly not known, there is no opportunity to compare the map matched path with the traversed path. To determine the quality of the map matching, we have to introduce quality measures that have the ability to evaluate how good the map matching result is. We distinguish between three quality indicators:

- Mapping percentage
- Average cost
- Hausdorff distance

Mapping percentage
First, the mapping percentage is the proportion of trips for which the map matching algorithm has found a result that is not rejected. Reasons for not being able to find a path are a disconnected graph, many missing measurements and large measurements errors, among others.

Average cost per meter
The average costs for path p, $\bar{c_p}$, is defined by simply dividing the total costs (**??**) of a path by the length of this path, l_p:

$$\bar{c_p} = \frac{\sum_{(i,j)\in p} c_{i,j}}{l_p} \tag{5.2}$$

Hausdorff distance
The Hausdorff distance is a widely used distance measure for shapes in general. It measures how far two subsets of a metric space are from each other. Informally, the Hausdorff distance is defined as the **greatest** of all distances from a point in one set to the **closest** point in the other set. Let A and B be subsets of M, with (M, d) a metric space. The *directed* Hausdorff distance $h(A, B)$ is the lowest upper bound (supremum) over all points in A of the distances to B:

$$h(A, B) = \sup_{a\in A} \inf_{b\in B} d(a, b) \tag{5.3}$$

The Hausdorff distance $H(A, B)$ is the maximum of the two values obtained by considering each of the two sets as the first set in (5.3) , i.e. the maximum of $h(A, B)$ and $h(B, A)$:

$$H(A, B) = \max\{\sup_{a\in A} \inf_{b\in B} d(a, b), \sup_{b\in B} \inf_{a\in A} d(a, b)\} \tag{5.4}$$

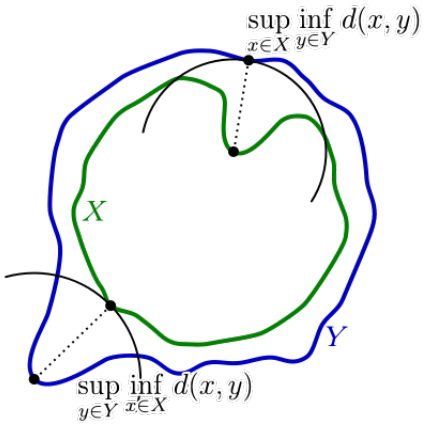


Figure 5.7: Directed Hausdorff distances as components of the Hausdorff distance between sets X (green line) and Y (blue line) [9]

Typically, in the case of map matching using a shortest path approach, the sets A and B are of different size, since not all points from A have a one-to-one-corresponding point in B. The Hausdorff distance can be suitable in this case, however a major drawback is that this measure does not take continuity into account. In the calculation of the Hausdorff distance, every point is assigned to the closest point in the other set and then maximizes over all distances. The Hausdorff distance can therefore be small for two sets that are are quite distant, see Figure 5.8.

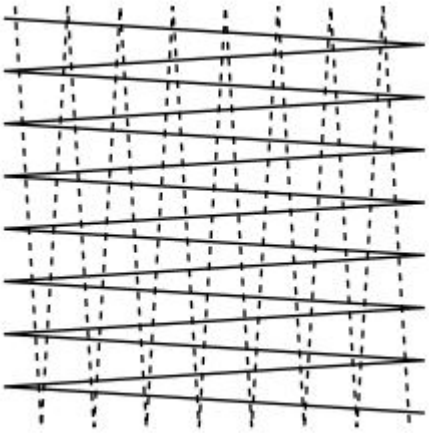


Figure 5.8: Hausdorff distance can be small for sets that are not similair

Moreover, the Hausdorff distance is very sensitive to noise. One single outlier can already determine the value for the Hausdorff distance.

We conclude that the Hausdorff distance tells us something about how close two sets are. Therefore, it can be a useful criterion, however, it has drawbacks as indicated above and, moreover, does not tell anything about the *average* closeness.

Chapter 6

Results

This section reports and discusses the results of the experiments as explained in Section 5.3. Consecutively, the case, validation, anonymous data and speedup experiments are addressed.

6.1 Case experiment

In the design of the solution approach, it has been tried to solve the two step problem from Section 2.3. The result is discussed by comparing the result from the benchmark algorithm with the results from the 4M algorithm and by looking at the ground truth. We stress that the constructed case is fully based upon real life data and that other cases can lead to other results. The case is developed to report the impact of 4M on a individual trip.

The case trip being analysed is the trip from example 2.3.1. This case trip focusses on the multimodal aspect of 4M, which is one of the main contributions of this thesis.

The two step map matching approach may fail in the case of a multimodal trip, because wrong segmentation in the first step is not recoverable anymore. By implementing the 4M algorithm, splitting the trip in unimodal segments in a first step is not needed anymore. Figure 6.1 shows the improvements that have been realised by implementing 4M. The path resulting by using 4M, consists of two trip legs: a train trip leg and a foot trip leg. These two trips leg have been correctly identified now, which is an improvement compared to the failure of the benchmark algorithm in this case.

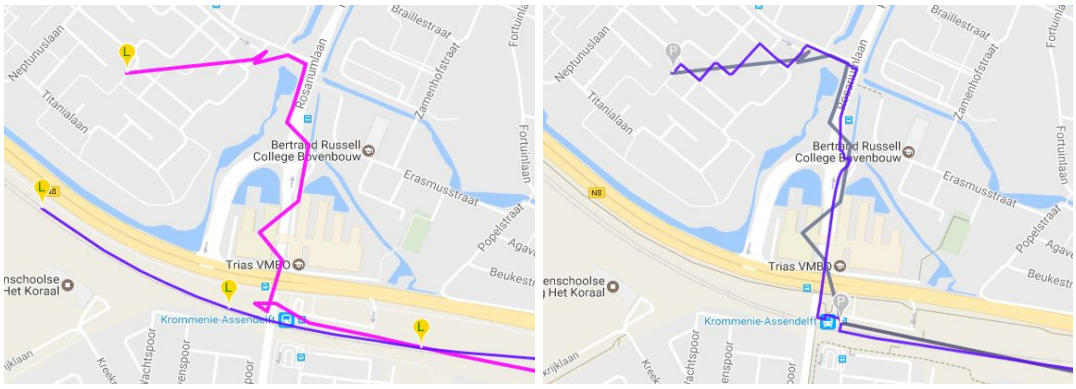


Figure 6.1: Re-analysis of Example 2.3.1: map matching results of BM (left) and 4M(right)

Measure	BM	4M
Path quality	90.6	100
Average cost per meter	24.9	3
Hausdorff distance	427	69
Modality	Train	Foot & Train
Processing time	3s	3s

Table 6.1: Node selection problem: results BM vs 4M

6.2 Validation experiment

In this validation experiment, the effect of executing 4M on a set of trips of which the ground truth is known is examined. The main advantage of this set is that a clear quality measure is available, the ground truth. Moreover, the ground truth can be used to validate the suggested quality indicators. The results are summarized in Table 6.2. In the next three subsections, we will discuss the ground truth, the computational speed and the quality indicators.

Measure	BM	4M
Path quality	90.72	94.89
Average cost per meter	13.45	10.66
Hausdorff distance	197.19	173.98
Processing time	10.6s	9.2s

Table 6.2: Overview results validation experiment with BM and 4M

6.2.1 Ground truth

First, Figure 6.2 shows the path quality of the validation set when analysed with BM and 4M. Note that many trips could not be mapped by the BM algorithm. Using 4M, much better results are obtained. The following mapping percentages have been accomplished:

- BM: 71.2 %
- 4M: 100 %

The average distance correctness improves from 90.72 to 94.89 percent. Most striking, the proportion of trips that have a path quality of above 95% increases from 50.85% to 72.88%.

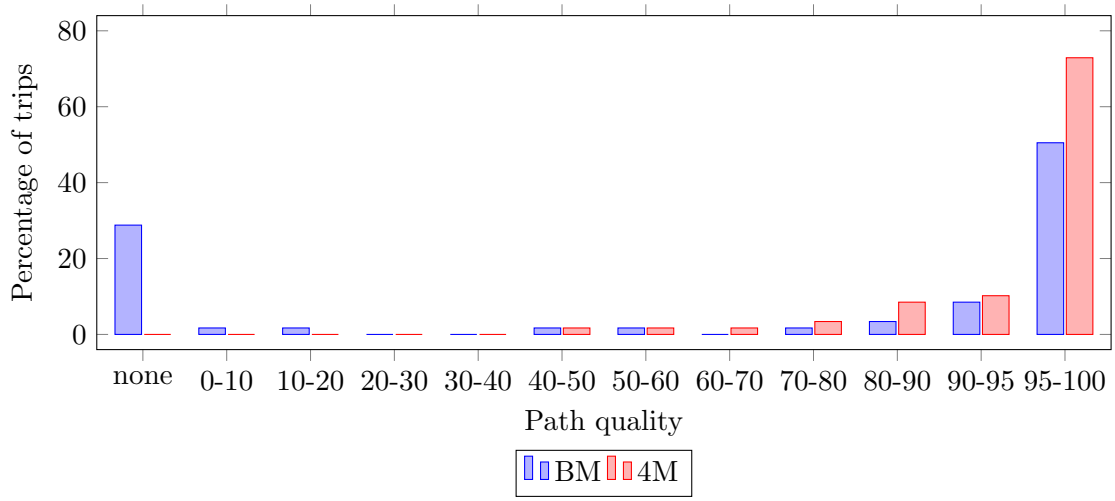


Figure 6.2: Path quality of validation trips

6.2.2 Computational Speed

Considering the computational speed, 4M performs slightly better than BM, see Figure 6.3. Analysing the validation set using 4M leads to more trips having a processing time of 0 to 3 seconds and less trips having a processing time of more than 60 seconds. The total time saving is 57 seconds, on a total of around 600 seconds; a 10% speedup. The average processing time decreases from 10.6 seconds to 9.2 seconds.

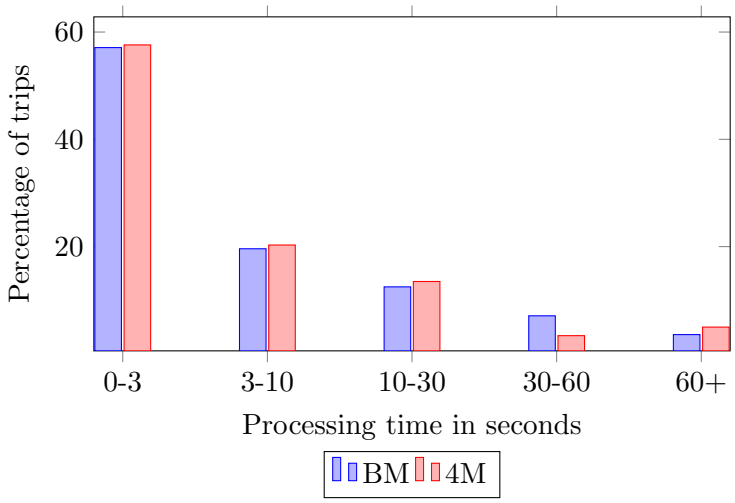


Figure 6.3: Computational speed of analysis of trips from validation set

6.2.3 Quality indicators

The fact that the actual traversed paths are known, gives the opportunity to compare the real paths with the mapped paths. However, in daily practise, it is not known by the analyst what the correct paths are. The only information that is available are the sensed locations and the mapped path. Based on this information, one must decide upon the quality of the resulting path. To do this, without knowing the correct path, we suggested *quality indicators* in Section 5.4. In this section we evaluate the extent to which two of the suggested quality indicators, the average cost per meter and the Hausdorff distance, are useful in determining the quality of a map matched path.

In Figure 6.4 the average cost per meter has been plotted against the quality, i.e. the distance correctness, for each trip. The blue vertical line represents the average of the average cost per meter. For visualising purposes, the trips are splitted in two categories: trips having a quality above 95% (open circles) and trips having quality below 95% (red triangles). From this, we observe that for this data set the red triangles are located mostly above the average of the average cost per meter (the blue vertical line). Most open circles have a average cost per meter lower below the average. This indicates that if a trip has on average cost per meters that are low, then the change is bigger that this trip has a high quality.

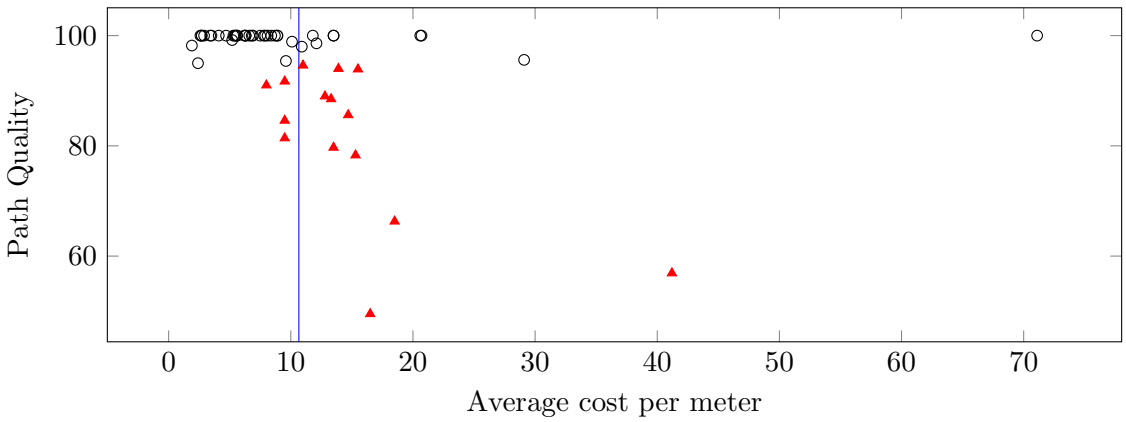


Figure 6.4: Relation between average cost per meter and quality. White circles denote trips with quality above 95%, red tringles below 95%

To investigate the mutual relation between quality and average cost per meter more rigorously, we use the statistical concept correlation. The most commonly used correlation coefficient is the Pearson correlation coefficient. However, this coefficient can be used only if the data is approximately normally distributed. Clearly, this is not the case here. Therefore, we use the Spearman rank correlation, which is less sensitive to strong outliers than the Pearson correlation. The Spearman rank correlation uses a monotonic function to describe the relationship between two variables. The main idea is to turn the measurement variables into ranked variables and use the Spearman rank correlation on the two sets of ranks. Ranked variables are those for which individuals observations are ordered from small to large and ranked accordingly. The Spearman rank correlation is known as being non-parametric in two ways: the result is a perfect correlation if the two variables are related by any monotonic function (instead of any linear function for the Pearson correlation) and, second, no knowledge about the joint distribution of the two variables is needed.

The Spearman rank correlation is defined as follows:

$$r_s = \frac{1 - \frac{6}{n(n^2-1)} \sum_i D_i^2 - \frac{1}{2}(T_x + T_y)}{\sqrt{(1 - T_x)(1 - T_y)}} \tag{6.1}$$

with $T = \frac{1}{n(n^2-1)} \sum_k t_k(t_k^2 - 1)$ and t_k equals the number of observations in the sample that have the same rank.

We obtain the following result for the Spearman rank correlation between average cost and quality of the validation set:

$$r_s = -0.51$$

The correlation coefficient r_s is always between -1 and 1. The value of -0.51 indicates that there exists a moderate negative relationship between the average cost and the quality. For every increase in the average cost of 1 unit, the quality decreases by 0.51 %.

In Figure 6.4 the Hausdorff distance has been plotted against the path quality for each trip. For visualising purposes, the trips are splitted in two categories again: trips having a quality above 95% (open circles) and trips having quality below 95% (red triangles). Looking at this figure, there appears to be no clear relationship between the Hausdorff distance and path quality.

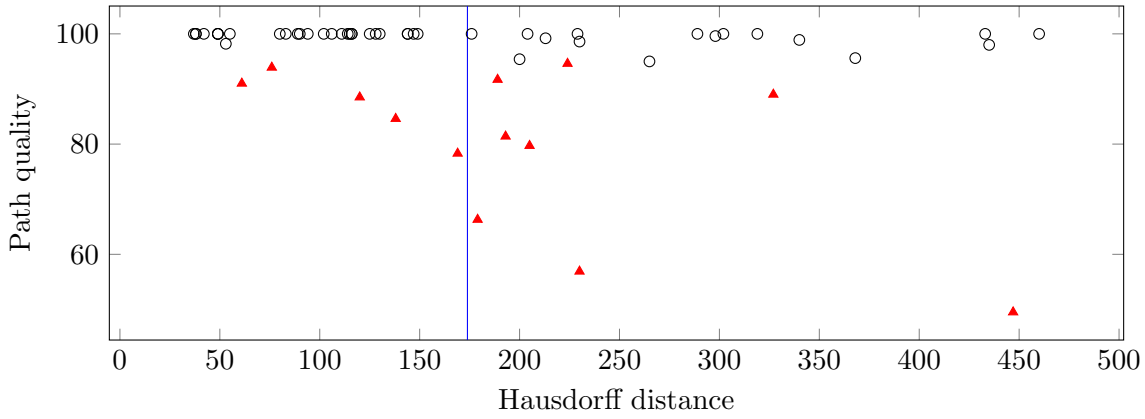


Figure 6.5: Relation between the Hausdorff distance and quality. White circles denote trips with quality above 95%, red tringles below 95%

The Spearman rank correlation coefficient has been calculated to be:

$$r_s = -0.38$$

The value of -0.38 indicates a weak negative relationship between the Hausdorff distance and the quality. Thereby, combined with the earlier mentoined drawbacks, the Hausdorff doest not seem to be a good quality indicator.

6.3 Anonymous data experiment

The scores for the quality indicators after running the anonymous data experiment as described in section 5.3 are reported in Table 6.3 below.

Measure	4M
Region	Enschede
Trips	3312
Triplegs	4225
Mapping percentage	94.6 %
Average cost per meter	13.01
Hausdorff distance	159.6

Table 6.3: Quality indicator results for anonymous dataset

We note the following:

1. Using 4M we obtain a **mapping percentage** of 94.6%
2. The **average cost per meter** is 13.01.
3. The **Hausdorff distance** is on average 159.6.

Figure 6.6 shows the mapping percentage on the left side, splitted by modality. Foot trips shows to be the hardest, having the lowest mapping percentage. Probably, this is best explained by the fact that these trips are often short and do contain inaccurate measurements. For such trips it is better to reject or skip the map matching, because a good quality can never be obtained. Train trips also score below average, which we could clarify by the sparse density of railway. The sparseness could lead to a disconnected graph $\bar{G}$ if the distance parameter is chosen too small.

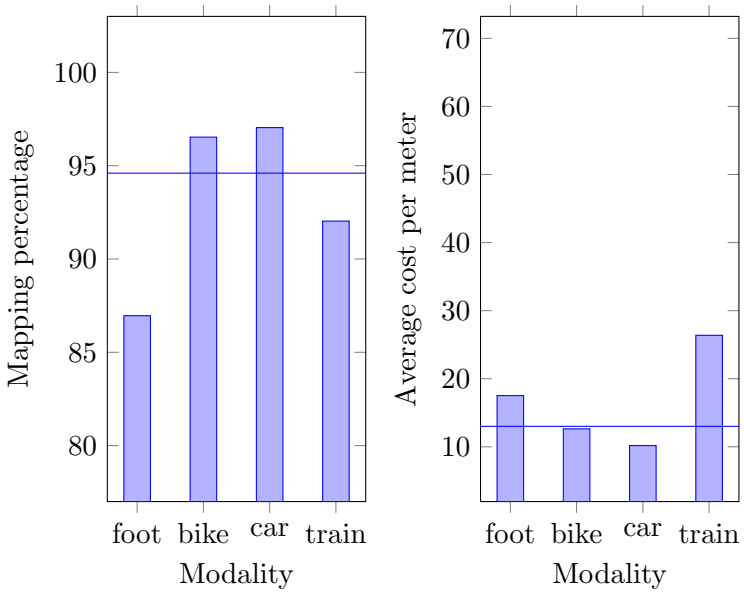


Figure 6.6: Mapping percentage (left) and average cost per meter (right), per modality. Horizontal lines represent the aggregated value

On the right hand side of Figure 6.6, the average costs per meter are presented, splitted by modality. The average cost is 13 per meter. This means that the average distance between p and the measurement line set $e(\bar{X})$ is approximately 13 meter. Mainly, train trips have a much higher average cost per meter. We explain this by the sparseness of railway edges in the initial graph G (and also in $\bar{G}$). Therefore, there are little railway edges available to choose from, which automatically has a big impact on the average cost per meter in case of inaccurate measurements.

In Figure 6.10, a boxplot shows some statistics of the processing times for trip from the anonymous dataset that have been analysed with the 4M algorithm. A boxplot is a standardizes way of of displaying the distribution of data. The box is sized from the

first to the third quartile, the median has been marked by a vertical line. The whiskers stretch from 1.5 of the inter quartile range (IQR) of the lower quartile to 1.5 IQR of the upper quartile. The average is represented by a dot. From the fact that the average is much higher than the median and the distance from the first quartile to the median is much smaller than the distance from the median to the third quartile , we conclude the processing time has a skew distribution. Large outliers do exist.

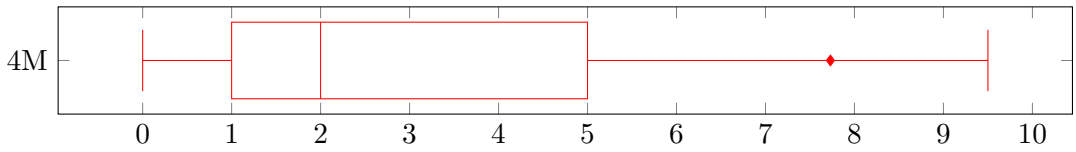


Figure 6.7: Boxplot of processing times for region Enschede with 4M.

Figure 6.8 displays the average processing time per modality. As expected, train trips have the longest processing time, because train trips need on average a lot of preprocessing and have a long distance on average. Also, modalities having a low mapping percentage, like train here, mostly have a higher processing time, because more time is spend on trying to find a mapping from origin to destination (Dijkstra's algorithm visits many vertices). Moreover, for train trips, some computational time is spend on splitting shared modality vertices (see Appendix B1).

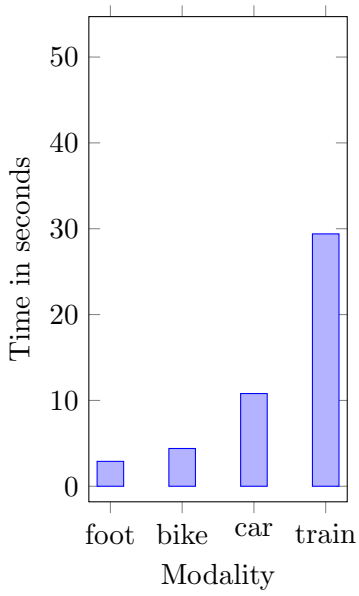


Figure 6.8: Average processing times, per modality

6.4 Speedup experiment

This last experiment is to show whether a speed up can be achieved if we implement a bidirectional version of Dijkstra's algorithm, core node identification to remove dead-end structures, or both. One case trip has been analysed to show the potential computational benefits of both speedup techniques. The sensing of this trip can be found in Figure 6.9.

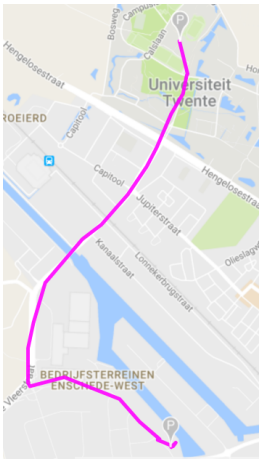
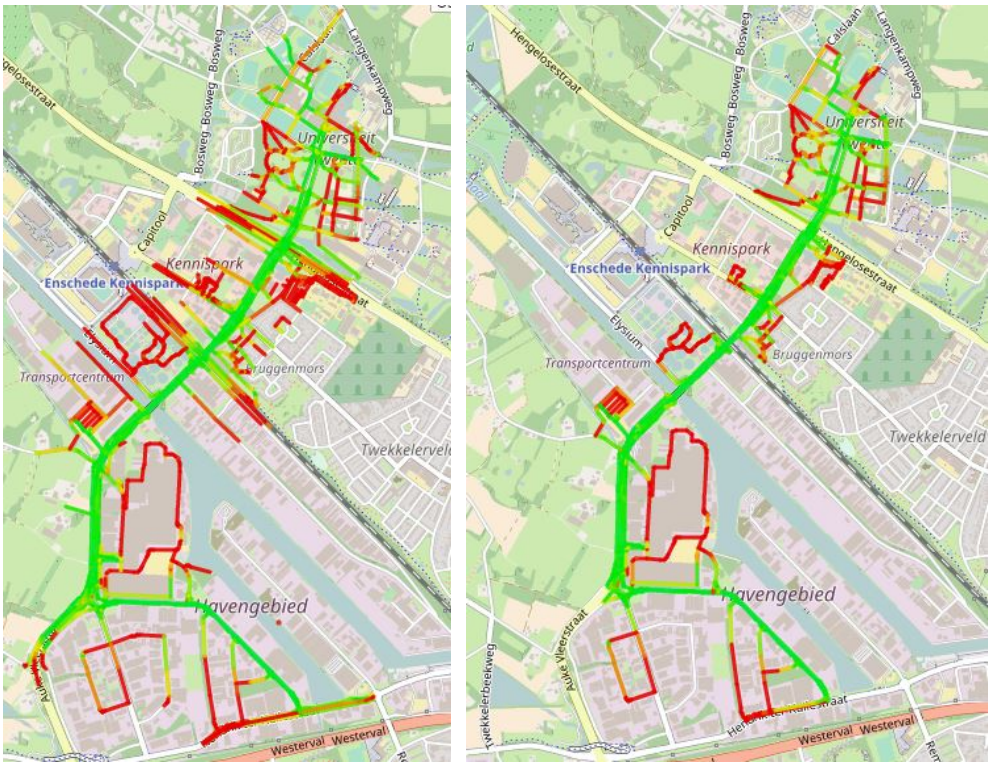


Figure 6.9: Sensing of case trip for speedup experiment

The graph that represents the trip under consideration is given in Figure 6.10(a). Hereby, the colors indicate the cost of each edge. The colors range from light green for the cheapest edges to dark red for the most expensive edges. Clearly, the edges that are close to the location measurements have low cost, i.e. they are green.

After executing the 2-core algorithm from Section 4.2 and thereby removing the dead ends, a subgraph of $\bar{G}$ is obtained. Let us call this subgraph H, which is printed in Figure 6.10(b). The size of $\bar{G}$ is **1679** edges, whereas H has **1174** edges, a reduction in size of about 30%.



(a) Graph $\bar{G}$ (b) Subgraph H of $\bar{G}$, dead ends deleted

Figure 6.10: OSM representation of trip speedup experiment: colors range from low costs per meter (green) to high costs per meter (red)

The goal of this experiment is to see if there is a speed up opportunity. We already found a significant graph size reduction by applying core node reduction, which could lead to a lower computational time for a shortest path algorithm. Therefore, we executed Dijkstra's algorithm on both $\bar{G}$ and H. To minimize the effect of an unstable cpu and memory storage, both algorithms have been executed 1000 times in sequence and average processing durations of the are reported in Table 7.1. A speed up of 24 milliseconds can be reported, relatively 16 %. Note that the processing duration does not include the preprocessing time for the removal of dead ends.

Also, the effect of executing a bidirectional variant of Dijkstra's algorithm is researched. Both $\bar{G}$ and H have been used as input graph. Again, the algorithms have been executed 1000 times. Compared with the unidirectional algorithm, which has an average computation time of 154ms (input $\bar{G}$) or 130ms (input H), the bidirectional variant speeds up to respectively 125ms and 102 ms. These are 19 % and 21 % improvements. In total, by applying both a bidirectional algorithm and removing dead ends, in this example, a speed up of 34 % has been achieved.

We stress that all four combinations of input graph and algorithm output the correct path.

	Dijkstra	Bidirectional Dijkstra
$\bar{G}$	154	125
$H \subseteq \bar{G}$	130	102

Table 6.4: Average processing duration (in ms), excluding preprocessing, of different algorithmic combinations for the speedup experiment

Chapter 7

Conclusion and discussion

In this chapter we first draw our main conclusions. Then, we discuss our findings by evaluating the approach and experimental setup, revealing limitations of this research. Based on this discussion, some recommendations for future research on the topic of multimodal map matching for smartphone data are given.

7.1 Conclusion

The goal of this research is to improve the performance of the map matching algorithm used by the company by targeting both the quality and the computation speed of the algorithm. To obtain this, we develop a multimodal map matching algorithm using a shortest path approach.

The main contribution is that the 4M algorithm finds a multimodal map matching without subdividing a trip in one of more unimodal segments beforehand, in order to avoid that this segmentation has an influence on the solution. Results show **significant improvements in the quality, simultaneously not being computational more expensive then the benchmark**. Quality indicators, such as the average cost per meter and the Hausdorff distance, are suggested to evaluate the quality of the result. However, they show only weak to moderate correlation with the path quality in our validation experiment.

A potential speed up of the shortest path algorithm can be obtained by implementing core node identification to delete dead-end structures and using a bidirectional variant of Dijkstra's algorithm.

Concluding, the newly developed multimodal map matching algorithm is able to find a path which resembles the ground truth and achieves improvements of quality and computational speed compared to the benchmark.

7.2 Discussion: Limitations and Recommendations

In this work we have been developing a multimodal map matching algorithm for smartphone data, named the 4M algorithm. Based on locations measurements collected by smartphone users, our algorithms tries to find a path in a multimodal network that equals the ground truth. We examined the effect of our algorithm on both case examples and data sets and compared those with the benchmark BM algorithm. The results are promising, indicating improvements of both quality and computational speed. However, some cautiousness when interpreting the results is needed. In this section we discuss some limitations and give recommendations for further research.

Numerical comparison First, the results obtained in this research are difficult to compare with numerical results from previous research. This is mainly due to the constructed datasets, which consists of real-life multimodal trips in our case. In contrast, previous research on map matching algorithms mostly dealt with synthetic data, just case trips and/or uni modal trips. Mainly, one should interpret our findings by comparing the results with the benchmark algorithm and valuing the relative improvement.

Parameter settings Parameter settings has not been extensively tested in this research, but they have been taken from experience. For example, the distance parameter r determines the size of $\bar{G}$ and has thereby some influence on computational speed. Testing different settings could enhance the performance of our algorithm.

More testing Unfortunately, we have not been able to compare the benchmark algorithm and the 4M algorithm on the same big size dataset consisting of trips of which the ground truth is not known. More systematic testing is needed to confirm our results and eventually find more directions of improvement. Moreover, the speed up techniques should be tested on multiple instances and integrated into the current software used by the company. However, the datasets are bigger than in earlier results.

Modality determination The reader should bear in mind that this research was focussed on the map matching algorithm solely. Modality determination is beyond the scope of this research. However, it would be of great interest to further examine the integration of the modality determination into the map matching algorithm in a multimodal framework. Variables such as speed and acceleration, which depend on the modality highly, could further enhance the map matching algorithm.

Quality indicators Lastly, a major problem in this field of research is the way in which the quality of map matching results are evaluated, mainly of those trips of which the ground truth is not known. Further research might explore quality indicators that have higher correlation with the quality of the mapped paths. A possible direction of thought could be an indicator that combines the average distance error with the maximum distance error or an indicator depending on the distribution of the error.

Bibliography

[1] Batagelj, V. (2011). An O(m) Algorithm for Cores Decomposition of Networks. *Advances in Data Analysis and Classification*, 5(2):129–145.

[2] Berge, C. and Ghouilla-Houri, A. (1965). *Programming, games and transportation networks*. Methuen, London.

[3] Bierlaire, M., Chen, J., and Newman, J. (2013). A probabilistic map matching method for smartphone GPS data. *Transportation Research Part C: Emerging Technologies*, 26:78–98.

[4] CBS (2016). Transport en mobiliteit 2016. Technical report, Centraal Bureau voor de Statistiek, Den Haag.

[5] Chen, J. and Bierlaire, M. (2015). Probabilistic Multimodal Map Matching With Rich Smartphone Data. *Journal of Intelligent Transportation Systems*, 19(2):134–148.

[6] Dantzig, B. (1963). *Linear Programming and Extensions*. Princeton University Press, Princeton.

[7] Fredman, M. and Tarjan, R. (1987). Fibonacci Heaps And Their Uses In Improved Network Optimization Algorithms. *Journal of the Association for Computing Machinery*, 34(3):596–615.

[8] Gong, H., Chen, C., Bialostozky, E., and Lawson, C. T. (2012). A GPS/GIS method for travel mode detection in New York City. *Computers, Environment and Urban Systems*, 36(2):131–139.

[9] Hausdorff_Distance (2013). In Wikipedia. Retreived April 20, 2017 from https://en.wikipedia.org/wiki/Hausddistance.

[10] Marchal, F., Hackney, J. K., and Axhausen, K. W. (2004). Efficient Map-Matching of large GPS data sets - Tests on a speed monitoring experiment in Zurich. *Arbeitsbericht Verkehrs- und Raumplanung*, (July):1–13.

[11] McGuckin, N. and Nakamoto, Y. (2004). Trips, Chains and Tours - Using an operational definition. In *NHTS Conference*, Washington DC.

[12] Nicholson, T. (1966). Finding the shortest route between two points in a network. *The computer journal*, 9(3):275–280.

[13] Nitsche, P., Widhalm, P., Breuss, S., and Maurer, P. (2012). A strategy on how to utilize smartphones for automatically reconstructing trips in travel surveys. *Procedia - Social and Behavioral Sciences*, 48:1033–1046.

[14] Ortúzar, J., Armoogum, J., Madre, J., and Potier, F. (2011). Continuous mobility surveys: the state of practice. *Transport Reviews*, 31(3):293–312.

[15] Quddus, M. A., Ochieng, W. Y., and Noland, R. B. (2007). Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328.

[16] Reddy, S., Mun, M., Burke, J., Estrin, D., Hansen, M., and Srivastava, M. (2010). Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks*, 6(2):1–27.

[17] Schuessler, N. and Axhausen, K. (2009). Processing Raw Data from Global Positioning Systems Without Additional Information. *Transportation Research Record: Journal of the Transportation Research Board*, 2105(8):28–36.

[18] Stopher, P. R., Greaves, S., Stopher, P. R., and Greaves, S. (2007). Household travel surveys: Where are we going? *Transportation Research Part A: Policy and Practice*, 41(5):367–381.

[19] Velaga, N. R., Quddus, M. A., and Bristow, A. L. (2009). Developing an enhanced weight-based topological map-matching algorithm for intelligent transport systems. *Transportation Research Part C: Emerging Technologies*, 17(6):672–683.

[20] Washington, S. (2015). Shortest path and vehicle trajectory aided map-matching for low frequency GPS data. *Transportation Research Part C: Emerging Technologies*, 55:328–339.

[21] White, C. E., Bernstein, D., and Kornhauser, A. L. (2000). Some map matching algorithms for personal navigation assistants. *Transportation Research Part C*, 8(1):91–108.

[22] Yin, H. and Wolfson, O. (2004). A Weight-based Map Matching Method in Moving Objects Databases 1. *In Proc. 16th SSDBM conf.*, pages 437–438.

[23] Zhao, Y. (1997). *Vehicle location and navigation systems*. Artech House Publishers, Norwood.

Appendix A: Notation

Symbol	Description
x_t	ground truth location at time t
$\bar{x}_t$	location measurement at time t
v_i^{lat}	latitude of vertex i
v_i^{lon}	longitude of vertex i
$c_{i,j}$	cost of edge (v_i, v_j)
v_i^m	modality for vertex i
G^m	Unimodal graph G for modality m
s	number of trip legs
p	map matched path: $(p_1, p_2, ..., p_s)$
p^{gt}	ground truth path
$Q(p)$	quality of path p
T	number of measurements
$\bar{Y}$	Set of all measurements: $\{\bar{y}_0, \bar{y}_1, \bar{y}_T\}$
$\bar{X}$	Set of location measurements: $\{\bar{x}_0, \bar{x}_1, \bar{x}_T\}$
R	Set of distance parameters: $\{r_0, r_1, ..., r_T\}$

Table 7.1: Symbols