European Commission
Seventh Framework Programme

PROJECT REDUCTION (# 288254)

Deliverable 3.4

Report on Eco-Routing Computation Techniques

August 31, 2013

(www.reduction-project.eu)

Contents

List of Tables

List of Figures

Project acronym: Reduction
Project full title: Reducing Environmental Footprint based on Multi-Modal Fleet management Systems for Eco-Routing and Driver Behaviour Adaptation

Work Package: 3.4

Document title: Report on Eco-Routing Computation Techniques
Version: 1.0
Official delivery date: 31/08/2013
Actual publication date: 31/08/2013
Type of document: Deliverable Report
Nature: Public

Authors: Yu Ma (AU), Bin Yang (AU), Chenjuan Guo (AU), Christian S. Jensen (AU), Manohar Kaul (AU)

Approved by:

Version	Date	Sections Affected
0.1	01/07/2013	Initial version
0.9	30/07/2013	Reviewed by AAU
1.0	15/08/2013	Reviewed by UHI
1.1	28/01/2014	Comments from 2nd review meeting addressed
1.2	07/02/2014	Comments from UHI addressed
2.1	08/08/2014	Comments from remote interim review addressed

Executive Summary

This document reports key techniques developed for solving tasks in "T3.3 Advanced Eco-Routing Methods" of Project REDUCTION WP3 during the period of 01/09/2012 – 31/08/2013. The primary objectives of T3.3 are summarised as follows.

- To develop runtime efficient, high-performance data structures and data mining algorithms for computing eco-routes.
- To develop scalable eco-routes computation methods.
- To conduct extensive experimental studies to verify the eco-routing algorithms.

To support advanced eco-routing techniques, a major challenge lies in how to assign weights that accurately capture the environmental impact to road segments, so-called eco-weights. Once the eco-weights have been obtained, existing routing algorithms, such as Dijkstra's algorithm, can be applied to compute eco-routes in road networks. In T3.3, we mainly focus on the tasks of assigning eco-weights to road network, and rely on GPS records to conduct the tasks.

As summarised in the task objectives, GPS records can arrive at the central system in a high velocity stream, yielding a large volume of data. It is possible that some road segments are covered with sufficient amounts of GPS records, whereas other road segments have no or few GPS records. Meanwhile, eco-weights on road segments can also be diverse. For example, weights on road segments can be constant values, can change over time, or can be uncertain. To cover these possibilities of eco-weights on road segments, while still considering the scalability and efficiency, we propose the following techniques for T3.3.

1. Given a huge amount of historical GPS records, a method is proposed to assign time-dependent and uncertain eco-weights to road segments that are covered by sufficient GPS records. This kind of eco-weights captures the basic traffic behaviour, e.g., periodic traffic variations, on the road segments [1], as reported in Chapter 2.
2. As the real time GPS data streams in, a method for predicting near future eco-weights of road segments is proposed. This kind of eco-weights considers the most recent data and thus reflects the real time traffic of road networks. This method helps to improve the accuracy of eco-routing [2] and is presented in Chapter 3.
3. For road segments that are not covered by sufficient numbers of GPS records to derive eco-weights using the above two methods, a method is proposed to infer eco-weights on such kinds of road segments [3], as presented in Chapter 4.

The proposed methods are evaluated using a massive GPS data set collected from Denmark. The experimental results indicate that these methods are effective, efficient, and scalable up to country-level road networks.

The proposed methods have also been partially integrated into a prototype system[1] developed by AAU and AU. The prototype uses both GPS data and CANBus data.

In addition, the proposed methods have been partially assessed in the first BEK field trial in WP5. Further performance and scalability evaluation of our techniques is planned in the second BEK field trial in the third year, 01/09/2013 – 31/08/2014.

[1] `http://daisy.aau.dk/its`

Chapter 1

Introduction

1.1 Project Description

The reduction of CO2 emissions is a great challenge for the transport sector nowadays. Despite recent progress in vehicle manufacturing and fuel technology, still a significant fraction of CO2 emissions in EU cities is resulting from road transport. Therefore, additional innovative technologies are needed to address the challenge of reducing emissions. The REDUCTION project focuses on advanced ICT solutions for managing multi-modal fleets and reducing their environmental footprint. REDUCTION collects historic and real-time data about driving behaviour, routing information, and emissions measurements, that are processed by advanced predictive analytics to enable fleets enhance their current services as follows:

1. Optimising driving behaviour: supporting effective decision making for the enhancement of drivers education and the formation of effective policies about optimal traffic operations (speeding, braking, etc.), based on the analytical results of the data that associate driving-behaviour patterns with CO2 emissions;

2. Eco-routing: suggesting environmental-friendly routes and allowing multi-modal fleets to reduce their overall mileage automatically; and

3. Support for multi-modality: offering a transparent way to support multiple transport modes and enabling co-modality.

REDUCTION follows an interdisciplinary approach and brings together expertise from several communities. Its innovative, decentralised architecture allows scalability to large fleets by combining both V2V and V2I approaches. Its planned commercial exploitation, based on its proposed cutting edge technology, aims at providing a major breakthrough in the fast growing market of services for ”green” fleets in EU and worldwide, and present substantial impact to the challenging environmental goals of EU.

1.2 Objectives of Work Package 3 (WP3)

REDUCTION’s division of technical work is composed of V2I/V2V communication (Work Package 1), ecological routing (Work Package 3), ecological driving (Work

Package 2) and distributed data mining (Work Package 2). In addition, REDUCTION includes four field-studies (Work Package 5), where the state-of-art advances are applied to real-world scenarios.

The objective of WP3 is to design and develop a software prototype that can convert vehicle-related data, primarily GPS data, to metrics that capture environmental impact. The prototype must handle very large volumes of data from different types of vehicles[1] and must efficiently compute the multi-modal eco-routes in both real-time and off-line modes. In addition, the prototype must be able to report on the temporal evolution of eco-routes, e.g., due to a variety of changes in the transportation infrastructure and its use. The work package will

- define the interfaces for how vehicles communicate with the server side and with each other.
- develop and prototype techniques for computing eco-routes,
- develop and prototype techniques for the validation of eco-routes.
- design and prototype high-performance data structures and algorithms for the handling of very large volumes of streaming data from the vehicles.
- develop and prototype efficient, off-line data mining algorithms capable of monitoring and reporting on the temporal evolution of eco-routes.

WP3 is organised in four tasks, namely:

- **Task 3.1** – Requirement specification and Software architecture (AAU, BEK).
- **Task 3.2** – Basic eco-routing methods (AAU, AU).
- **Task 3.3** – Advanced eco-routing methods (AU).
- **Task 3.4** – Prototype Consolidation (AAU, AU).

Deliverable D3.2 described techniques developed in Task 3.2, including

- a method for lifting a 2D (i.e., latitude-longitude) road network to achieve a 3D (i.e., latitude-longitude-altitude) road network and hence providing grade information for all road segments, which can increase the accuracy of eco-weights assigned to the road segments;
- an evaluation of the utility of vehicular environmental impact models to assigning eco-weights to road segments; and
- a basic routing algorithm that recommends eco-friendly routes to users on a road network where eco-weights are annotated using the grade information, vehicular environmental impact models, and GPS trajectories.

Thus, the objectives of deliverable D3.2 have been achieved.

[1] The vehicle fleets considered in this report refer to passenger vehicle fleets rather than freight vehicles.

1.3 Objectives of Deliverable 3.4

This deliverable D3.4 describes advanced eco-routing techniques developed and implemented to solve task "T3.3 Advanced Eco-Routing Methods of WP3". We mainly put our effort into developing techniques for assigning various types of eco-weights that accurately capture environmental impact to road segments. Once the eco-weights, which may be time-dependent, are obtained on segments of a road network, traditional routing algorithms, e.g., Dijkstra's algorithm and A^* algorithm, can be used. When routing based on time-dependent eco-weights, algorithms that support both FIFO and non-FIFO graphs [4] can be applied, meaning that we do not make the assumption that the travel costs on road networks must satisfy the FIFO property[2].

In task T3.3, we cope with high velocity and large volume historical and real time GPS data. We do not make the assumption that the whole road network is covered with GPS data. Instead, we not only consider segments that are covered with sufficient GPS records, referred to as "hot" edges, but also address issues caused by segments with little or no GPS data, called "cold" edges. Further, we take various types of eco-weights into account. For example, eco-weights on a road segment can be constant or time-dependent, i.e., changing over time, or can be uncertain while following certain distributions under different traffic.

To address the aforementioned issues, we have conducted research on assigning eco-weights in task T3.3. The developed techniques are described as follows.

I. Assigning time-dependent uncertain eco-weights to hot edges. In Chapter 2, we present histogram-based techniques for assigning time-dependent uncertain eco-weights to hot edges in a road network using a collection of historical GPS data. Specifically, the major contributions are summarised as follows.

- A road network is proposed to capture environmental impact on road segments.
- A sequence of histograms is employed to represent the time-dependent uncertain eco-weight on each road segment[3]. Various compression techniques, including histogram merging and buckets reduction, are proposed to maintain compact histograms while achieving good modeling accuracy.
- A comprehensive empirical study is conducted using two years of GPS vehicle tracking data in order to gain insight into the effectiveness and efficiency of the proposed approach.

II. Predicting the near future eco-weights on hot edges using real time GPS data. In Chapter 3, we present a method for predicting eco-weights of hot edges in the near future. Using the proposed method, near future travel costs, e.g., travel time or greenhouse gas emissions, in a road network can be inferred using real time GPS data, enabling a variety of online routing services, e.g., eco-routing. The contributions are presented as follows.

- A general framework is proposed that is capable of modeling the traffic behaviour of a road network and thereby enables routing services that optimise different travel-related costs.

[2]Travel times on road networks are generally assumed to satisfy the FIFO property, but fuel consumption or GHG emissions on road networks may not satisfy the FIFO property

[3]Note that our aim is to estimate the travel cost distributions for routes, and a route can only consists of consecutive edges. Thus, we only take into account the dependencies between adjacent edges, but not the dependencies between other sets of edges. It is of interest to identify the dependencies between other sets of edges when modeling traffic in a large region, which is out of the scope of the deliverable.

- A spatio-temporal hidden Markov model is formalised to model the traffic behaviour of a road network.
- Learning algorithms are proposed to obtain individual state sets for all road segments and to determine the parameters needed to configure an STHMM.
- Comprehensive experiments are conducted to elicit the design properties of the proposed framework and algorithms.

III. Annotating cold edges with time-dependent eco-weights. In Chapter 4, techniques that take into account the structural similarities between cold and hot edges are proposed to annotate cold edges with time-dependent eco-weights. Techniques proposed in Chapter 2 are applied here to generate eco-weights that represent the CO_2 emissions of traversing a road segment. The contributions are presented as follows.

- We formalise a novel problem, road network weight annotation, which aims to assign time-dependent eco-weights to both cold and hot edges.
- A general framework for assigning time varying trip cost based weights to the edges of the road network is presented, along with supportive models, including a directed, weighted graph model capable of capturing time-varying edge weights and a trip cost model based on time varying edge weights.
- Two novel and judiciously designed objective functions are proposed to contend with the data sparsity. A weighted PageRank-based objective function aims to measure the variance of weights on road segments with similar traffic flows, and a second objective function aims to measure the weight difference on road segments that are directionally adjacent.
- Comprehensive empirical evaluations with real data sets are conducted to elicit pertinent design properties of the proposed framework.

The proposed techniques do not make any assumption on the supported types of vehicle fleets. In other words, the proposed techniques can be applied for both passenger vehicle fleets or freight vehicle fleets. The proposed techniques rely on the available data. When the data is collected from passenger vehicle fleets (or freight vehicle fleets), the obtained eco-weights can be used to provide eco-routes for passenger vehicle fleets (or freight vehicle fleets). In this sense, the proposed techniques are highly-flexible. Since only GPS data from passenger vehicles is available to us, the proposed techniques are only tested on the passenger vehicles in D3.4.

We use a large GPS data set collected at 1 Hz (i.e., one GPS record per second) in North Jutland, Denmark during week days from April 2007 to March 2008 to verify the proposed algorithms. The data is from an experiment where young drivers start out with a rebate on their car insurance and then are warned if they speed and are penalised financially if they continue to speed. The GPS data is map matched [5] to OpenStreetMap's road network for Denmark[4], where 34%, 29%, 15%, 9%, and 13% of the data occurs on tertiary, secondary, residential, motorway, and other roads according to OpenStreetMap road categories[5], respectively. Most of the data is from urban and suburban regions.

The objectives of this deliverable (D3.4) and the summary of the progress achieved along each objective are summarised in Table 1.1.

[4] `http://www.openstreetmap.org`
[5] `http://wiki.openstreetmap.org/wiki/Highway_tag_usage`.

Table 1.1: Summary of the Objectives of Deliverable 3.4

Objective Description	**Progress Summary**
To develop runtime efficient, high-performance eco-routes computing algorithms	Developed three efficient algorithms that are able to assign different types of eco-weights to a road network.
To develop scalable eco-routes computation methods	The developed three algorithms are scalable.
To conduct extensive empirical evaluation of algorithms	Conducted experiments on a 2-year GPS data collected in Denmark to verify that the proposed algorithms are efficient, effective, and scalable up to contra-level road network.

Chapter 2

Representing Time-Dependent Uncertain Eco-Weights For Road Networks Using Histograms

Global warming is mainly due to "greenhouse effect" [6] – the concentration of greenhouse gases (GHG) in the Earth's atmosphere prevents heat from escaping into the space. The GHG emissions are mainly generated by the burning of fossil fuels, and transportation is an important and increasing fossil fuels burning sector. Thus, reducing the GHG emissions from transportation is crucial in combating global warming.

Eco-routing is an easy-to-employ and effective approach to reduce GHG emissions from transportation. Given a source-destination pair, the eco-route is the most environmentally friendly route, i.e., the route that consumes the least fuel or produces the least GHG emissions is able to achieve approximately 8–20% reductions in GHG emissions or fuel consumption from road transportation [7].

Neither the shortest nor the fastest routes generally have the least environmental impact. Figure 2.1 shows an example of the shortest route, the fastest route, and the eco-route between source A and destination D.

Vehicle routing services rely on a weighted graph representation of the road network in which vehicles travel. The vertices and edges represent road intersections and road segments, respectively. The key to enabling effective eco-routing services is to assign eco-weights to edges, where the eco-weights accurately capture the environmental costs (i.e., GHG emissions or fuel consumption) of traversing the edges. Based on the obtained weighted graph, various existing routing algorithms [8][9] can be applied directly to enable eco-routing.

A single value as an edge weight typically can not fully capture the environmental cost of traversing the edge. For instance, while traversing an edge, aggressive drivers may generate more GHG emissions than average drivers. Thus, an uncertain eco-weight recording the distribution of all possible environmental costs of traversing the edge better captures the reality.

Further, the eco-weights should be time dependent, due to the temporal variations of traffic conditions. For instance, during peak hours, traversing an edge normally

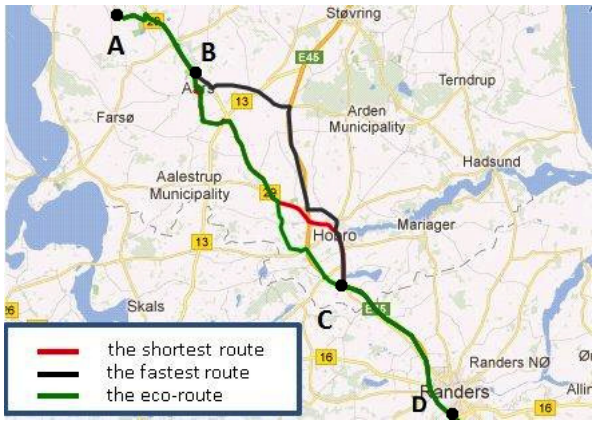
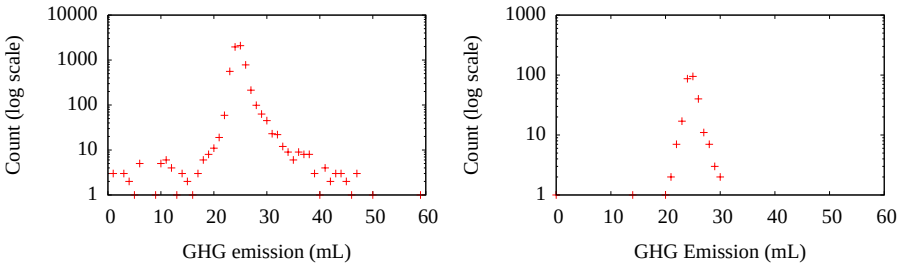


Figure 2.1: Eco-Route, Fastest Route, and Shortest Route

produces more GHG emissions than that of during off-peak hours. The distributions of GHG emissions on an edge in North Jutland, Denmark during peak hours and off-peak hours are shown in Figure 2.2. Since the edge is more likely to be congested during peak hours, some traversals produce between 20 mL and 50 mL GHG emissions. During off-peak hours, all the traversals produce less than 30 mL GHG emissions.



(a) Peak (8 a.m.–10 a.m.) (b) Off-Peak (8 p.m.–12 p.m.)

Figure 2.2: Distributions of GHG Emissions On An Edge

Vehicle tracking data, such as global positioning system (GPS) data, can be used for computing time-dependent, uncertain eco-weights of edges. Further, according to a recent benchmark on vehicular environmental impact models [7], environmental costs of traversing edges can be derived based on high-frequency GPS data (i.e., one GPS record per second) using pertinent vehicular environmental impact models. Thus, based on a collection of GPS tracking data, it is possible to assign time-dependent, uncertain eco-weights to edges in a road network.

We study how to obtain time-dependent, uncertain eco-weights for edges from a massive GPS tracking data set. In particular, advanced histogram based techniques are proposed to approximate GHG emissions distributions during different intervals. As we need to maintain histograms for a large number of edges in a road network, we propose approximation techniques to compress the histograms, thus obtaining space efficient and statistically accurate synopses for the distribution of GHG emissions for all the edges.

2.1 Related Work

2.1.1 Histograms

Histograms are used widely to capture the statistical distribution of data. In general, histograms partition the range of the original data into buckets and maintain statistics for each bucket. The criteria for choosing buckets vary across different types of histograms.

The most widely used histogram is the equi-width histogram [10], which groups contiguous ranges of values into buckets with uniform width. In the last few decades, many advanced histograms have been proposed. Such histograms let the underlying data distribution define the bucket boundaries. For instance, the V-optimal histogram [10] determines the buckets so that the variance of the overall frequency among buckets is minimised, while the MaxDiff histogram [10] aims to place values that differ substantially in the same bucket, which yields the overall best accuracy-time trade-off [10]. Due to the simplicity of equi-width histgram, we use these as the outset for the edge weights at the initial stage.

Histograms are used widely in database management systems. For instance, histograms [11, 12] are employed to efficiently estimate the sizes of query results and for extracting information from streaming and probabilistic data [13]. Similarly, we also use histograms to estimate the data distribution of our GPS streaming data.

A key objective of histograms is to capture the distribution of the underlying data in a space-efficient manner. To save space and obtain an improved trade-off between accuracy and space, several histogram compression techniques have been proposed. Wavelet histograms [14] transform an original histogram into a set of wavelet coefficients that yields customizable compression ratios. Regression is another popular technique that can be used to compress histograms [15]. Give a histogram H with n buckets, *regression* can produce an approximation H' with m buckets, $m < n$. Regression tends to minimise a certain error metric in comparison to the original values, such as sum of squared errors (*SSE*). For the purpose of saving storage space, we propose different histogram compression techniques to reduce the buckets in the histograms. Our experiments show that our histogram techniques can achieve considerable data approximation accuracy with limited storage usage.

2.1.2 Time-Dependent Routing

Much work has been conducted to enable time-dependent routing services. Different approaches based on Dijkstra's algorithm [8, 16, 9] solve the problem in different scenarios. However, we note that they all lack a detailed description of how to obtain the time-dependent weights, and some works randomly generate synthetic weights.

T-drive [17] is a recent effort to identify the fastest routes given departure times by learning from a set of taxi GPS records. T-drive assigns time-dependent, travel-time based weights to edges. T-drive clusters travel-time observations of an edge and uses the clusters as histogram buckets. The same buckets are used for the histograms on an edge during different intervals. In contrast, our proposal is more flexible and is able to assign different buckets (based on distinct distributions) to the histograms on an edge during different intervals. For instance, more buckets are used for representing more complicated distributions, e.g., the GHG emissions distributions during peak hours. Since T-drive is considered as the most relevant work and as the state-of-the-art technique for estimating travel times, we experimentally compare our proposal with an

approach based on T-drive and find that we can obtain a better trade-off between storage space and accuracy. In particular, we can obtain more accurate weights at lower storage cost. In addition, our approach covers both travel-time weights and eco-weights, and T-drive is only evaluated based on travel-time weights [17].

2.2 Problem setting and definition

2.2.1 Time Dependent Histograms

Given a multiset of cost values C, the range of the cost values $Range(C)$ is the set of non-duplicated values that occur in C. The *data distribution* of the cost values in C, denoted as $DD(C)$, is a set of $(value, probability)$ pairs, where *value* indicates a value in $Range(C)$, and *probability* is the number of occurrences of the value in C divided by the total number of values in C. An example is shown as follows, where multiset C contains GHG emission values observed from a road segment.

$$\begin{aligned} C =& \{5,8,10,20,15,10,20,20,34,28\}; \\ Range(C) =& \{5,8,10,15,20,28,34\}; \\ DD(C) =& \{(5,0.1),(8,0.1),(10,0.2), \\ & (15,0.1),(20,0.3),(28,0.1),(34,0.1)\}. \end{aligned}$$

Given a multiset of cost values C, a *histogram* is an approximation of C's data distribution $DD(C)$ [18]. In particular, a histogram $H = \langle (b_1, p_1), \ldots, (b_n, p_n) \rangle$ is a vector of $(bucket, probability)$ pairs, where a *bucket* $b_i = [f_i, l_i)$ indicates a sub-range of the cost values, where $f_i, l_i \in Range(C)$ indicate the starting and ending values of the sub-range. The buckets are disjoint, i.e., $b_i \cap b_j = \emptyset$ if $i \neq j$; and all elements in $Range(C)$ belong to the union of the all buckets, i.e., $b_1 \cup \ldots \cup b_n$. The width of a bucket is defined as $|b_i| = l_i - f_i$. If every bucket in a histogram has the same width, i.e., $|b_1| = \ldots = |b_n|$, the histogram is called *equi-width histogram*. A *probability* p_i records the percentage of the cost values that are in the sub-range indicated by b_i. The sum of all probabilities is 1, i.e., $\sum_{i=1}^{n} p_i = 1$.

As a running example, assuming that an equi-width histogram h with width 10 is built on the multiset C, we have:

$$H = \langle ([5,15),0.4),([15,25),0.4),([25,35),0.2) \rangle.$$

Given a time interval of interest TI, a *Time Dependent Histogram* is a vector of $(period, histogram)$ pairs, where the time interval TI is partitioned into several *period*s. Specifically, in a time dependent histogram $tdh = \langle (T_1, H_1), \ldots, (T_m, H_m) \rangle$, *period* T_i indicates a particular period in TI, and *histogram* H_i is the histogram of the cost values observed in period T_i. The periods covers the time interval of interest, i.e., $T_1 \cup \ldots \cup T_m = TI$.

Two equal-width histograms H_1 and H_2, are *isomorphic* if they have the same number of buckets and they represent the same data range.

The corresponding probabilities of the buckets may still be different. Figure 2.3 shows two isomorphic histograms that describe the GHG emissions of an edge during periods [8 a.m.,9 a.m.) and [9 a.m.,10 a.m.), respectively.

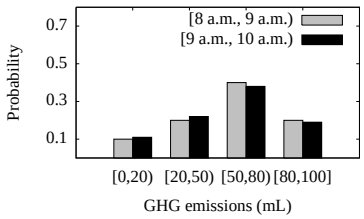


Figure 2.3: Isomorphic Histograms

2.2.2 Road Networks and Trajectories

An *Eco Road Network* (*ERN*) is a weighted, directed graph $G = (V, E, F)$, where V and E are vertex and edge sets. An vertex $v_i \in V$ represents a road intersection or the end of a road, and an edge $e_k = (v_i, v_j) \in E$ models a directed road segment that enables travel from vertex v_i to vertex v_j. Function F:$E \rightarrow TDH$ assigns a time dependent uncertain eco-weight which is represented as a time dependent histogram to each edge in E. Specifically, the time dependent histogram on edge e_i, i.e., $F(e_i)$, models the dynamic GHG emissions based travel costs on edge e_i.

A *Trajectory* $tr = \langle p_1, p_2, \ldots, p_x \rangle$ is a sequence of GPS records. Each GPS record p_i specifies the location (typically with latitude and longitude coordinates) and velocity of a vehicle at a particular time stamp $p_i.t$. Further, the GPS records in a trajectory are ordered based on the time stamps of the GPS records, i.e., $p_i.t < p_{i+1}.t$ for $i \in [1, x)$. Given the road network where the trajectories occurred, a GPS record in a trajectory can be mapped to a specific location on an edge in the road network using a some map matching algorithm [19].

2.2.3 Problem Definition and Framework Overview

Given a set of map matched trajectories *TR* in a road network $G' = (V, E, null)$, the paper studies how to obtain the corresponding Eco Road Network $G = (V, E, F)$. Specifically, the key task is to determine function $G.F$ that assigns time dependent histograms to edges based on trajectory set *TR*.

An overview of the framework that determines the function $G.F$ is shown in Figure 2.4. The pre-processing module transforms the map matched trajectories into a set of *traversal records* of the form $r = (e, t_s, tt, ge, tr_j)$. Such a traversal record r indicates that edge e is traversed by trajectory tr_j starting at time t_s. The travel time and the GHG emissions of the traversal are tt and ge, respectively. Note that the travel times can be derived directly from the GPS records by simply getting the difference between the times of the fist and last GPS records. Various vehicular environmental impact models [7] can be applied to compute the GHG emissions from the GPS records. After pre-processing, each edge e_i is associated with a set of traversal records $R_i = \{r | r.e = e_i\}$.

In the ERN building module, initial time dependent histograms are built for edges based on the traversal records associated with them. Maintaining the time dependent histograms of all edges in a large road network may require very large storage space. To reduce the storage overhead, approximation and compression techniques are employed to reduce both the number of $(period, histogram)$ pairs in the time dependent histograms and the number of the buckets in individual histograms. Specifically, *histogram merging* and *bucket reduction* are conducted in order to obtain a compact representation of an ERN. Finally, the Eco Road Network G is returned, where the function

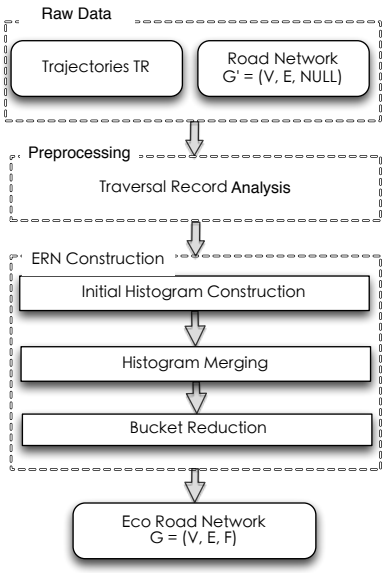


Figure 2.4: Framework Overview

$G.F$ assigns compact, time-dependent, uncertain eco-weights to edges.

2.3 ERN Building

In this section, we propose methods to generate an *Eco Road Network* from our raw GPS observations.

2.3.1 Initial Time Dependent Histograms

An initial time dependent histogram is built for every edge $e_i \in E$ based on the traversal records associated with e_i. Given a time interval of interest TI, e.g., a day or a week, and the finest temporal granularity α, e.g., 15 minutes or one hour, TI is split into $\lceil \frac{TI}{\alpha} \rceil$ periods, where the j-th period $T_j = [(j-1) \cdot \alpha, j \cdot \alpha)$. For each period T_j, an equi-width histogram H_j is built based on the traversal records that occurred in the period, i.e., $R_i^{(j)} = \{r | r.e = e_i \wedge r.t_s \in T_j\}$.

The histograms for all periods share the same range $[l,u]$, where l and u are the lowest and highest GHG emissions (or travel times) observed in R_i. Further, the same number of buckets is chosen for all histograms, where the number of buckets N_{bucket} is an adjustable parameter. Thus, $\lceil \frac{TI}{\alpha} \rceil$ isomorphic histograms are obtained for each edge. Assuming α is set to one hour, Figure 2.3 shows two isomorphic histograms during periods [8 a.m., 9 a.m.) and [9 a.m., 10a.m.) for an edge in North Jutland, Denmark. The similarity of the two histograms motivates us to compress them into one histogram without losing too much information. We proceed to show how to compress the histograms with acceptable accuracy using histogram merging and bucket reduction.

2.3.2 Histogram Merging

If two temporally adjacent histograms H_i and H_{i+1} represent similar data distributions, it is potentially attractive to merge the two histograms into one histogram $\overline{H}$ that represents the data distribution for a longer period $\overline{T} = T_i \cup T_{i+1}$.

Given two distributions, several techniques exist to measure their similarity, such as cosine similarity, the K-S test and the chi-square test. The simplicity and efficiency of computing cosine similarity makes it appropriate for use when evaluating the similarity between two histograms.

To facilitate the usage of cosine similarity, we treat a histogram as a vector. Given a histogram $H = \langle (b_1, p_1), \ldots, (b_n, p_n) \rangle$, its vector is $V(H) = \langle p_1, \ldots, p_n \rangle$. Since all the initial histograms are isomorphic and equi-width, they have the same number of buckets, and each bucket corresponds to the same sub-range. Thus, all the vectors are isomorphic, meaning that they have the same number of dimensions, and each dimension represents the same entity, i.e., the probability in a particular sub-range. The *histogram similarity* between two histograms is defined as the cosine similarity between their vectors, as shown in Equation 2.1.

$$sim(H_i, H_j) = \frac{V(H_i) \odot V(H_i)}{\| V(H_i) \| \cdot \| V(H_j) \|}, \tag{2.1}$$

where $\odot$ indicates dot product between two vectors, and $\cdot$ indicates the product between two reals.

To merge two adjacent isomorphic histograms H_i and H_{i+1},into a new histogram $\overline{H}$, the probability value for the k-th bucket in $\overline{H}$ is computed based on Equation 2.2.

$$\overline{H}.p_k = \frac{H_i.p_k \cdot H_i.c + H_{i+1}.p_k \cdot H_{i+1}.c}{H_i.c + H_{i+1}.c}, \quad \forall\, k \in [1, n], \tag{2.2}$$

where $H_i.c$ is the total number of cost values that are used to derive H_i, which is equivalent to the number of traversal records in the i-th period. When merging two isomorphic histogram, the obtained $\overline{H}$ is isomorphic with H_i and H_{i+1}.

The data probability of the k-th bucket in $\overline{H}$ is not just the average of $H_i.p_k$ and $H_{i+1}.p_k$. As the number of traversal records in the k-th buckets of histograms H_i and H_{i+1} may be different, we use the weighted average to construct the data probability of the k-th bucket in $\overline{H}$, as shown in Equation 2.2. We also maintain $\overline{H}.c = H_i.c + H_{i+1}.c$. so that we can use the count of traversal records in $\overline{H}$ for further merging step.

Given an initial time-dependent histogram *tdh* for an edge, Algorithm 1 returns a corresponding merged time-dependent histogram $\overline{tdh}$. Histogram merging is conducted iteratively (lines 2–10). In each iteration, an adjacent pair of histograms with the highest histogram similarity is identified (line 3). If the similarity exceeds a merging threshold T_{merge}, the two histograms are merged into a new one according to Equation 2.2, and the new period of the merged histogram is the union of the two corresponding periods (lines 4–6). The iteration continues until the highest histogram similarity is lower than the merging threshold T_{merge} (lines 7–9). The merging threshold is user-specified and adaptive. We study the effect of different merging thresholds in Section 2.5.

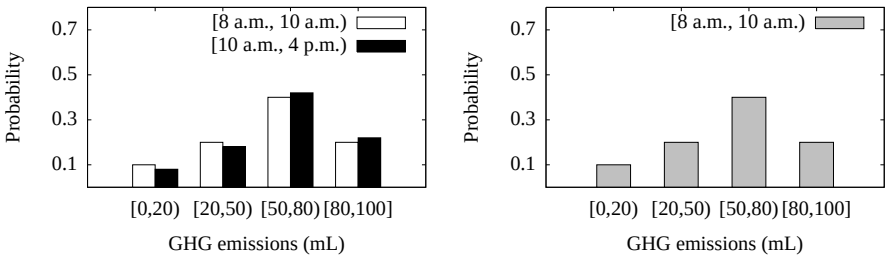
An example of histogram merging is shown in Figure 2.5. Two histograms representing the GHG emissions during two adjacent periods are shown in Figure 2.5(a). They are merged into a single histogram as shown in Figure 2.5(b).

Algorithm 1: HistogramMerge

Input:
Time-dependent histogram of an edge e: $tdh = \langle (T_1, H_1), \ldots, (T_n, H_n) \rangle$;
Merge threshold: T_{merge};

Output:
Merged time-dependent histogram of edge e: $\overline{tdh} = \langle (\overline{T}_1, \overline{H}_1), \ldots, (\overline{T}_m, \overline{H}_m) \rangle$;

1: $\overline{tdh} \leftarrow tdh$;
2: **while** TRUE **do**
3: Find the adjacent histogram pair (H_i, H_{i+1}) with the highest histogram similarity according to Equation 2.1;
4: **if** $\text{sim}(H_i, H_{i+1}) \leq T_{merge}$ **then**
5: Generate a new histogram $\overline{H}$ according to Equation 2.2, and a new period $\overline{T} = T_i \cup T_{i+1}$;
6: Replace (T_i, H_i) and (T_{i+1}, H_{i+1}) in $\overline{tdh}$ by $(\overline{T}, \overline{H})$;
7: **else**
8: BREAK;
9: **end if**
10: **end while**
11: return $\overline{tdh}$;



(a) Before Merging (b) After Merging

Figure 2.5: Histogram Merging

2.3.3 Bucket Reduction

Histogram merging reduces the numbers of histograms. Bucket reduction is able to further reduce the size of individual histograms, which is orthogonal to histogram merging. Three options are explored for bucket reduction, namely *regression*, *advanced regression*, and *wavelets*.

2.3.3.1 Regression

Given an original histogram H, regression transforms it into a new histogram $\widehat{H}$ that approximates the data distribution represented by H using fewer buckets. The new histogram $\widehat{H}$ is not necessarily equi-width, meaning that different buckets may have different widths. Histogram regression is conducted by merging two adjacent buckets. The range of the new bucket is the union of the original two buckets, and the probability of the buckets is the sum of the probabilities of the two original buckets. For example, given a histogram $H = \langle (b_1, p_1), \ldots, (b_i, p_i), (b_{i+1}, p_{i+1}), \ldots, (b_n, p_n) \rangle$, after merging buckets b_i and b_{i+1}, the new histogram is $\widehat{H} = \langle (b_1, p_1), \ldots, (b_{i-1}, p_{i-1}), (b_x, p_x), (b_{i+2}, p_{i+2}), \ldots, (b_n, p_n) \rangle$, where $b_x = b_i \cup b_{i+1}$ and $p_x = p_i + p_{i+1}$.

The sum of squared error (*SSE*) is employed to measure the discrepancy between the original histogram H and the histogram after bucket reduction $\widehat{H}$. Since the error is only introduced by the buckets we merge and we merge only one pair of adjacent buckets, for example, merging buckets b_i and b_{i+1} into a single bucket b_x, thus we only need to calculate the error introduced by this merge operation, which is defined in Equation 2.3.

$$\begin{aligned} SSE(H, \widehat{H}) = & \left(\frac{|H.b_i|}{|H.b_i| + |H.b_j|} \widehat{H}.p_x - H.p_i\right)^2 \\ & + \left(\frac{|H.b_j|}{|H.b_i| + |H.b_j|} \widehat{H}.p_x - H.p_j\right)^2 \end{aligned} \tag{2.3}$$

where $|H.b_i|$ is the range of ith bucket in histogram H, and $H.p_i$ is the probability of the ith bucket in histogram H. A smaller *SSE* indicates $\widehat{H}$ achieves a smaller accuracy loss compared to the original histogram H.

Given a merged time-dependent histogram $\overline{tdh}$, Algorithm 2 reduces the buckets in each histogram in $\overline{tdh}$ according to a user specified *SSE* value as the reduction threshold T_{red}.

2.3.3.2 Advanced Regression

We consider a scenario where a storage budget (i.e., number of buckets for an edge) is given, and where we need to decide how to use the buckets in different histograms in representing the GHG emissions for different periods so that we maximise the overall accuracy. For example, we may use more (or fewer) buckets for the histogram representing the peak hours (or off-peak hours), instead of distributing the buckets evenly.

Given a merged time-dependent histogram $\overline{tdh}$ of an edge, and a reduction threshold T_{red} indicating the total number of buckets available for the edge, Algorithm 3 describes how to obtain the final time-dependent histogram within the storage budget, while maintaining good accuracy. Note that for different edges, the reduction threshold T_{red}, i.e., the bucket budget, may be different. A simple heuristic is to assign higher bucket quotas to the edges that have more merged histograms in the time dependent histogram *tdh* after histogram merging phase. As an edge with more histograms of one

Algorithm 2: RegressionBasedBucketReduction

Input:
Merged time-dependent histogram of edge e: $\overline{tdh} = \langle(\overline{T}_1,\overline{H}_1),\ldots,(\overline{T}_m,\overline{H}_m)\rangle$;
Bucket Reduction Threshold: T_{red};

Output:
Final time-dependent histogram of edge e: $\widehat{tdh} = \langle(\overline{T}_1,\widehat{H}_1),\ldots,(\overline{T}_m,\widehat{H}_m)\rangle$;

1: $\widehat{tdh} \leftarrow \overline{tdh}$;
2: **for** each histogram $\overline{H}_i$ in $\overline{tdh}$ **do**
3: **while** TRUE **do**
4: Find an adjacent bucket pair $(\overline{H}_i.b_j,\overline{H}_i.b_{j+1})$ that leads to the lowest *SSE* loss according to Equation 2.3;
5: Generate candidate histogram H_i' by merging b_j and b_{j+1} in $\overline{H}_i$;
6: **if** $SSE(\overline{H}_i,H_i') < T_{red}$ **then**
7: Replace the corresponding two buckets in $\overline{H}_i$ by the merged bucket;
8: **else**
9: BREAK;
10: **end if**
11: **end while**
12: **end for**
13: return $\widehat{tdh}$;

edge requires more buckets to retain its distribution information, more buckets should be assigned to the edge.

So given a bucket budget for each edge, Algorithm 3 determines how to distribute the buckets among all the histograms of a single edge.

Advanced regression also works iteratively. For each iteration, it scans all the adjacent buckets paris and finds then pair of adjacent buckets that achieve the least *SSE*. We then merge these buckets (lines 2–11). Note that to identify two buckets that need to be merged, every histogram in the given $\overline{tdh}$ has to be checked. This process terminates when the total number of buckets for the edge is below the given reduction threshold T_{red}.

2.3.3.3 Wavelets

Wavelet are able to transform a histogram into a sequence of coefficients (i.e., real values) that represent the original histogram. After applying wavelet decomposition on an equi-with histogram, a compact data synopsis that comprises a small set of wavelet coefficients is obtained. A histogram can be resumed based on the wavelet coefficients. The more coefficients we keep (equivalently, the more space we use), the closer the reconstructed histogram to the original histogram. The details of wavelet decomposition are omitted due to the space limitation, but can be found elsewhere [20].

Here, a reduction threshold T_{red} is used that specifies the maximal number of coefficients. Given a merged time-dependent histogram, wavelet decomposition is applied to each histogram, and each histogram is transformed into a sequence of T_{red} coefficients.

Algorithm 3: AdvancedRegressionBasedBucketReduction

Input:
Merged time-dependent histogram of edge e: $\overline{tdh} = \langle(\overline{T}_1, \overline{H}_1), \ldots, (\overline{T}_m, \overline{H}_m)\rangle$;
Bucket Reduction Threshold: T_{red};

Output:
Final time-dependent histogram of edge e: $\widehat{tdh} = \langle(\overline{T}_1, \widehat{H}_1), \ldots, (\overline{T}_m, \widehat{H}_m)\rangle$;

1: $\widehat{tdh} \leftarrow \overline{tdh}$
2: **while** Total buckets in $\widehat{tdh}$ exceeds T_{red} **do**
3: $minSSE \leftarrow \infty$;
4: **for** each histogram $\overline{H}_i$ **do**
5: **for** each adjacent buckets $\overline{H}_i.b_j$ and $\overline{H}_i.b_{j+1}$ in $\overline{H}_i$ **do**
6: Generate candidate histogram H_i' by merging b_j and b_{j+1} in $\overline{H}_i$;
7: **if** $SSE(\overline{H}_i, H_i') < minSSE$ **then**
8: Record the pair buckets into *MinPairBuckets*;
9: $minSSE \leftarrow SSE(\overline{H}_i, H_i')$;
10: **end if**
11: **end for**
12: **end for**
13: Merge the pair of buckets with minimum *SSE* (recorded in *MinPairBuckets*) in all histograms in the edge, and update $\widehat{tdh}$;
14: **end while**
15: return $\widehat{tdh}$;

2.3.4 Dynamic Maintenance

After building an ERN based on historical GPS trajectories, it is of interest to maintain the ERN as real time GPS records arrive in order to capture the real-time traffic status and to improve estimation accuracy. Updating the time-dependent histograms obtained from historical trajectories each time a GPS record arrives incurs high cost and is also not necessary. Previous work [21] applies, an error tree to keep track of the K most important coefficients for a wavelet histogram, and the coefficients are updated periodically after receiving enough incoming updates. This method can be applied in our setting if wavelet histograms are used for bucket reduction.

For non-wavelet histograms, we apply a sliding window (e.g., the most recent 15 minutes) based approach. Given an edge, we keep updating a real-time histogram $\widetilde{H}$ using the GPS data that occurs on the edge in the sliding window and keep track of the dissimilarity between $\widetilde{H}$ and the historical H for the edge. As the cardinality of $\widetilde{H}$ and H may be different, we first transform $\widetilde{H}$ and H to a pair of isomorphic histograms $\widetilde{H'}$ and H' to compute their similarity. If $\widetilde{H'}$ and H' are similar, H can be used for estimating GHG emissions directly. A dissimilarity may indicate an event such as an accident or road construction has occured on the edge, which make the traffic condition deviate from the normal case. Then, real-time histogram $\widetilde{H}$ is applied for estimating GHG emissions. As a histogram normally requires about 0.1KB in our system, real-time histograms for all the edges fit in memory.

We rebuild the time-dependent histograms periodically using a log-based method. We first save all incoming GPS records into a log for each edge. If the log size (i.e., the number of GPS records) exceeds a threshold *MaxLogSize*, the edge's time-dependent histogram is rebuilt. Note that the frequency of rebuilding the time-dependent histograms varies across edges. In conclusion, we avoid rebuilding the ERN each time we receive GPS data, and also capture the recent traffic in the ERN.

2.4 GHG Emissions Estimation

After histogram merging and bucket reduction, we obtain an ERM where each edge is associated with a compact time-dependent histogram. In this section, we study how to use the obtained ERN to estimate the GHG emissions of trajectories.

Rather than estimating a single value for a trajectory, e.g., the expected GHG emissions, we estimate the distributions of GHG emissions using histograms. This yields much more detailed information than a single value, which are quite useful for many applications, e.g., stochastic route planing [22, 23] and probabilistic threshold-based routing [24, 25]. For instance, when a logistics company plans trips, it is useful analysis is to check whether using a certain route (based on historical trajectories) generates more than 500 ml GHG emissions with probability at least 80%. This kind of analysis can only be supported if the distributions of GHG emissions are known, and it is impossible if only expected GHG emissions are known.

The distribution of the GHG emissions of a trajectory is estimated based on the ERN. In particular, the distribution of the GHG emissions of a trajectory, also represented by a histogram, is achieved by aggregating pertinent (w.r.t. the traversal time) histograms of the edges in the trajectory. In particular, according to the starting traversal time t of an edge e in the trajectory, where e has time dependent histogram $tdh = \langle (T_1, H_1), \ldots, (T_m, H_m) \rangle$, histogram H_i is selected for the histogram aggregation if $t \in T_i$.

2.4.1 Dependence Between Adjacent Edges

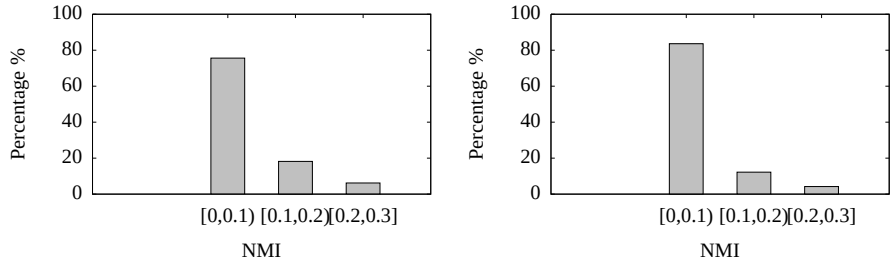
When aggregating the histograms of the edges in a trajectory, we need to consider the dependencies of the GHG emission distributions of adjacent edges. Most existing work [24, 22, 23, 26] assumes that the travel costs (e.g., travel times) on adjacent edges are independent. To evaluate this assumption, we conduct an empirical study on a collection of frequently traversed adjacent edge pairs. Specifically, we identify 82 edge pairs that each is traversed by at least 1,000 trajectories.

The GHG emissions distributions of two adjacent edges are modeled as two random variables X and Y, and normalised mutual information $\mathbf{NMI}(X,Y)$ is applied to quantify the dependency between X and Y, as shown in Equation 2.4.

$$I(X,Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)log(\frac{p(x,y)}{p(x)p(y)})$$
$$\mathbf{NMI}(X,Y) = 2\cdot\frac{I(X,Y)}{H(X)+H(Y)}, \tag{2.4}$$

where $H(\cdot)$ denotes entropy.

Figure 2.6(a) shows the percentage of edge pairs w.r.t. different ranges of **NMI** values. It suggests that the majority of adjacent edges tend to be independent; however, some adjacent edges do have non-negligible dependencies, as indicated by the last two bars in Figure 2.6(a).



(a) NMI (b) NMI Considering Time Periods

Figure 2.6: GHG Emissions Dependency

Next, we investigate whether the dependence varies if travesals occur at different times. Assume that clear peak and off-peak hours exist for adjacent edges e_1 and e_2. To further investigate the dependency among two edges, for each edge pair, we partition the time domain according to the intervals obtained from the edges' corresponding time dependent histograms. For each partition, we compute **NMI** only using the trajectories that occurred in the partition. The results, shown in Figure 2.6(b), suggest that the GHG emissions dependencies between adjacent edges are reduced when we take into account the traversal times. However, some adjacent edges still have relatively high dependencies.

Based on the above findings, the independency assumption, which is used extensively in the literature, does not always hold. To better model the dependencies, we introduce a *virtual edge* for each pair of dependent consecutive edge (i.e., whose **NMI** exceeds a threshold). As for a normal edge, a time dependent histogram can be obtained that represents the distribution of GHG emissions for a virtual edge. Given a route R, Algorithm 4 describes how to utilise virtual edges when estimating GHG emissions for the route.

Algorithm 4: SelectEdgesForEstimation

Input:
A route R that contains a sequence of N consecutive edges $e_1, \ldots, e_N$;
Output:
A set of edges selected for GHG emissions estimation, S;

$i \leftarrow 1$;
while $i \leq N$ **do**
  e_j is next edge of e_i in R;
  if e_i is not dependent on the next edge e_n or $i = N$ **then**
    insert e_i into S;
    $i \leftarrow i + 1$;
  else
    if $j = N$ **then**
      insert the virtual edge of e_i and e_j: $e_{v.ij}$ into S;
      $i \leftarrow i + 2$;
    else
      e_k is next edge of e_j in R;
      if NMI(e_i, e_j) > NMI(e_j, e_k) **then**
        insert the virtual edge of e_i and e_j: $e_{v.ij}$ and e_k into S;
      else
        insert e_i and the virtual edge of e_j and e_k: $e_{v.jk}$ into S;
      end if;
      $i \leftarrow i + 3$;
    end if;
  end if;
end while;
Return S;

Consider the adjacent edges e_1, e_2, and e_3 in the road network show in Figure 2.7. Here, e_1 and e_2 are dependent, and e_2 and e_3 are also dependent. When we estimate GHG emissions for a route that contains e_1, e_2, and e_3, we compare the **NMI** of e_1 and e_2 and the **NMI** of e_2 and e_3. The pair with the higher **NMI** is used as a virtual edge, and the remaining edge is used as a single edge. So if the **NMI** of e_1 and e_2 is the highest, we use the virtual edge e' generated from e_1 and e_2. By modeling adjacent edges with high dependency as virtual edges, we expect to get better accuracy when we estimate GHG emissions for routes.

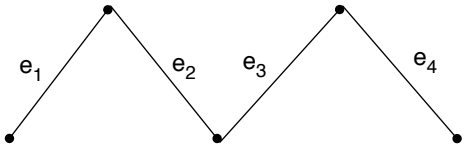


Figure 2.7: Road Network Example

We do not consider dependence between edges that are not adjacent because (1) it is not easy to identify such dependence using GPS data; (2) if really needed, such dependence can be computed based on the current dependence modeling of adjacent edges.

2.4.2 Histogram Aggregation

We estimate the distribution of GHG emissions along a route by aggregating the histograms of the edges (or virtual edges) covered by the route. If two adjacent edges in the route are dependent, i.e., there exists a virtual edge that is composed of the two adjacent edges, we use the virtual edge instead of the original two edges when aggregating histograms. Thus, all the histograms to be aggregated are independent. While aggregating the histograms, we start from the edge (or virtual edge) on one end of the route and proceed with the aggregate computation until we reach the other end of the route. The resulting histogram is the estimated histogram of GHG emissions for the whole route. Suppose we have two histograms H_1 and H_2 that represent the GHG emission distribution on two consecutive edges e_1 and e_2. We then apply *Histogram Aggregate* that yields a histogram H' that represents the aggregated GHG emissions for traversing both e_1 and e_2.

We use discrete convolution techniques to aggregate H_1 and H_2. We first transform each histogram into a list L $(value, prob)$ pairs in which the values are integers. We can get several value points from each bucket that reflect the data distribution derived from the original histogram H. The values here are in the range between the minimum and maximum data values of H. And the resolution of the resulting values can be customised by changing the histogram aggregate parameter, T_{agg}, that defines the granularity with which we extract data points from buckets. For example, given a bucket with data range [10, 16), if T_{agg} = 1, the result value list is $\langle 10, 11, 12, 13, 14, 15\rangle$, and if T_{agg} is 2, the result value list is $\langle 10, 12, 14\rangle$. After we get the value-prob pair lists L_1 and L_2, the aggregated value-prob list L' is shown in Equation 2.5.

$$\{\langle p_i.v + p_j.v, p_i.p \cdot p_j.p\rangle | p_i \in L_1 \wedge p_j \in L_2\} \tag{2.5}$$

Algorithm 5 shows how to aggregate two histograms into a single histogram, and T_{agg} is the customizable parameter that defines the resolution that we extract data points from the data range. We refer to this method as **point-wise aggregation**.

Algorithm 5: HistogramAggregate

Input:
Time-dependent histograms H_1 and H_2 of two consecutive edges e_1 and e_2;
Aggregate resolution: T_{agg};

Output:
Aggregated time-dependent histogram H' of e_1 and e_2;

1: Based on T_{agg}, transform H_1 and H_2 to lists of $\langle value, prob \rangle$ pairs with required resolution, namely L_1 and L_2;
2: **for all** p_i in L_1 **do**
3: **for all** p_j in L_2 **do**
4: $p' \leftarrow (p_i.val + p_j.val,\ p_i.prob \cdot p_j.prob)$;
5: add p' to candidate list L';
6: **end for**;
7: **end for**;
8: Rearrange the value list L';
9: Generate the result histogram H' that approximates the data distribution derived from L';
10: Return H';

The performance of Algorithm 5 deteriorates drastically as the number of edges in a trajectory grows, which makes it unattractive to use for longer trajectories. By making minor changes to Algorithm 5, we obtain an alternative methods to aggregate histograms with better performance and desirable accuracy. Instead of extracting multiple points from a buckets, **median aggregation** uses the median point of each bucket in the histogram when we transform a histogram into a $\langle value, prob \rangle$ pair list. Thus the following steps to generate aggregated histogram are identical with **point-wise aggregation**.

Histograms can be considered as the estimated disitributions of random variables. And The data in each buckets is considered as uniformly distributed. Given two histograms H_1 and H_2, we can use bucket-wise convolution to aggregate them. We first transform H_1 and H_2 to two histograms with equal bucket size, denoted as H_1' and H_2'. The convolution of H_1' and H_2' is also a histogram, denoted as H_{con}, as shown in Equation 2.6.

$$H_{con}[i].p = \sum H_1'[j].p \cdot H_2'[k].p, \tag{2.6}$$

where $H[i]$ is the ith bucket in histogram H and $Range(H_{con}[i]) = Range(H_1[j]) + Range(H_2[k])$. Again, as there might be overlap between buckets in H_{con}, we rearrange the buckets in H_{con} and generate an equi-width histogram as the final result. Algorithm 6 shows how to perform **bucket-wise aggregation**. After all the iterations, we rearrange buckets in H_{con} in (Line 7). We combine two buckets with the same data range and split a data range if it contains the range of another bucket; thus, we obtain buckets without overlap and duplicates. We compare these three aggregate methods empirically in Section 2.5.

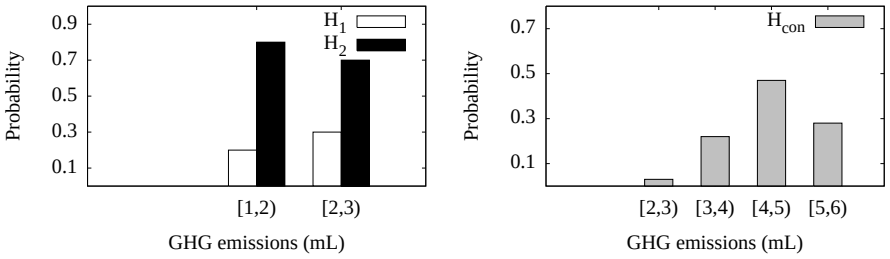
To illustrate, consider histograms H_1 and H_2 in Figure 2.8(a). The size of buckets in H_1 and H_2 is 1. H_1 and H_2 can reprensented as $\langle([1,2),0.2),([2,3),0.8)\rangle$ and $\langle([1,2),0.3),([2,3),0.7)\rangle$. Then according to Equation 2.6, the result histogram H' can be represented as $\langle([2,4),0.06),([3,5),0.14),([3,5),0.24),([4,6),0.56)\rangle$. The buckets in H' overlap, so we rearrange them and generate the equi-width historgam shown in

Algorithm 6: HistogramAggregateII

Input:
Time-dependent histograms H_1 and H_2 of two consecutive edges e_1 and e_2;
Output:
Aggregated time-dependent histogram H_{con} of e_1 and e_2;

```
1: for all buckets b_i in H_1 do
2:   for all buckets b_j in H_2 do
3:     b' ← [[b_i.l + b_j.l, b_i.l + b_j.u), b_i.p · b_j.p]
4:     add b' to H_con
5:   end for;
6: end for;
7: Rearrange the generated buckets;
8: Return H_con;
```

Figure 2.8(b) as the final result. The final result can be reprensented as $\langle([2,3),0.03),([3,4),0.22),([4,5),0.47),([5,6),0.28)\rangle$.



(a) Original Histograms (b) Convoluted Histogram

Figure 2.8: Histogram Convolution

2.5 Emprical Study

We conduct a range of experiments to gain insight into the accuracy, efficiency, and storage properties of the proposed approaches.

2.5.1 Experimental Settings

We use a large GPS tracking data set containing more than 200 million GPS records. The data is collected from 150 vehicles traveling in Denmark, during January 2007 to December 2008. The sampling frequency is 1 HZ. We use the road network of Denmark from OpenStreetMap[1]. The road network contains around 414K vertices and 818K edges. We apply an existing map matching tool [19] to match the GPS records to the road network, from which we get a set of trajectories TR. VT-micro [27, 28], a robust model that can evaluate vehicular environmental impact according to a recent benchmark [7], is applied to compute the GHG emissions of the trajectories. The time

[1] http://www.openstreetmap.org/

interval of interest TI is set to a day [0:00, 24:00), and the finest temporal granularity α is set to 1 hour. The bucket widths are set to 10 ml for GHG emissions.

We vary temporal granularity α, merge threshold T_{merge}, reduction threshold T_{red}, and aggregation threshold T_{agg}. The unit of α is hour. Our experiments mainly focus on the values that are shown in bold.

Table 2.1: Parameter Settings

Parameters	Values
T_{merge}	0.9, 0.91, 0.92, 0.93, 0.94, **0.95**, 0.96, 0.97, **0.98**, 0.99
T_{red} for regression	0.1, 0.11, 0.12, 0.13, 0.14, **0.15**, 0.16, 0.17, **0.18**, 0.19
T_{red} for Wavelet	16, 20, 24, **28**, 32, 36
T_{red} for Advanced Regression	35, 40, 45, **50**, 55, 60
T_{agg}	**1**, **2**, 3, **4**, 5
α	0.5, **1**, 2

An edge with few records during a year is either unlikely to have much traffic and traffic variation, or it is not covered by the vehicles in the GPS data set. To avoid abnormalities caused by such edges, we only consider the subset of the road network that is composed of edges that have at least 500 cost records, denoted as E. The resulting, smaller road network contains $|E| = 1,916$ edges.

For the edges with infrequent or no GPS records, we apply existing techniques [3] to obtain baseline GHG emissions estimates for predefined peak and offpeak periods, respectively.

In the experiments, we consider four kinds of histograms: (1) Initial time-dependent equi-width histograms; (2) merged time-dependent histograms obtained after histogram merging phase; (3) compact time-dependent histograms obtained after bucket reduction phase; and (4) T-drive based histograms. We include T-Drive in the experiments because it also uses histogram-based techniques and is considered as the state-of-the-art technique for estimating travel costs.

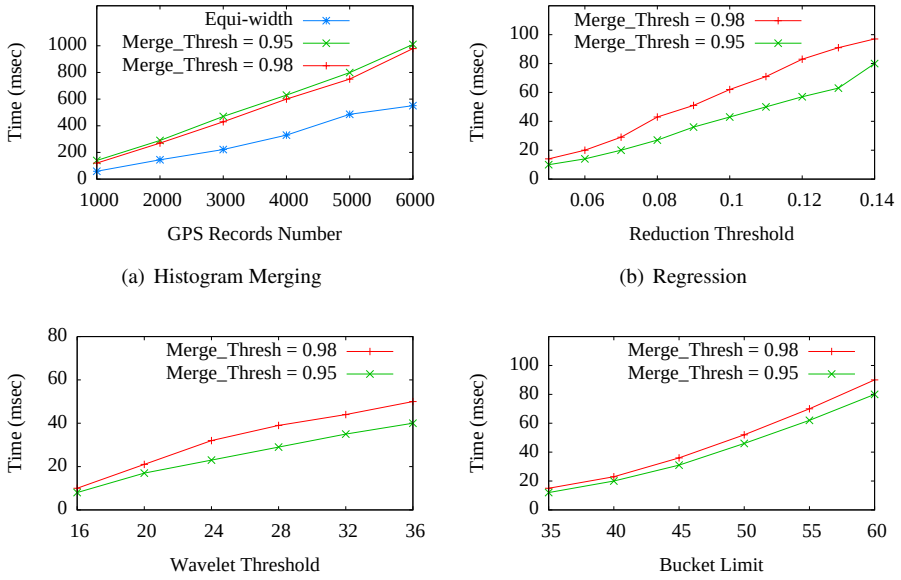
2.5.2 Efficiency

We report the run-times for histogram merging, bucket reduction, and histogram aggregation.

Figure 2.9(a) shows the run-time to build equi-width histograms and to merge the histograms for one edge. The run-time increases as the the number of GPS records increases. Thus, if an edge is associated with more records, it takes longer time to be build initial histograms. Figures 2.9(b)–2.9(d) show the time cost of different bucket reduction methods for varying reduction when the number of associtated GPS records is 6,000. The wavelet method takes less time to achieve the desired reduction than the other two regression-based methods.

We further study the time cost to aggregate histograms with different methods while the number of edges in a trajectory varies. Given value lists with M and N elements, the time cost to aggregate them is O(M·N). Figure 2.10(a) shows the overall performance of our histogram aggregation methods. We set the aggregation threshold to 1, 2, and 4 for the first histogram aggregation family. Figure 2.10(b) shows the effect of using virtual edgea. If **NMI** of two adjacent edges exceeds 0.2, we build histograms for

corresponding virtual edges in advance,we only generate virtual edges for adjacent edge pairs. We can see that when T_{agg} = 4 for the point-wise aggregation, using virtual edges is able to yield up to a 20% speedup for a trajectory with 10 edges.



(a) Histogram Merging (b) Regression

(c) Wavelets (d) Advanced Regression

Figure 2.9: Run-Time Efficiency

2.5.3 Accuracy

Since a histogram approximates original data distribution, we need to study the approximation accuracy. Specifically, we measure the distance between the original data distribution and different histogram representations, including initial equi-width histograms, histograms after merging, and histograms after bucket reduction.

Suppose the original data distribution in a period is $dd = (v_1, p_1)$, $\ldots, (v_n, p_n)$, where v_i and p_i indicate a value and its probability. The accuracy of a histogram in the period is defined by Equation 2.7.

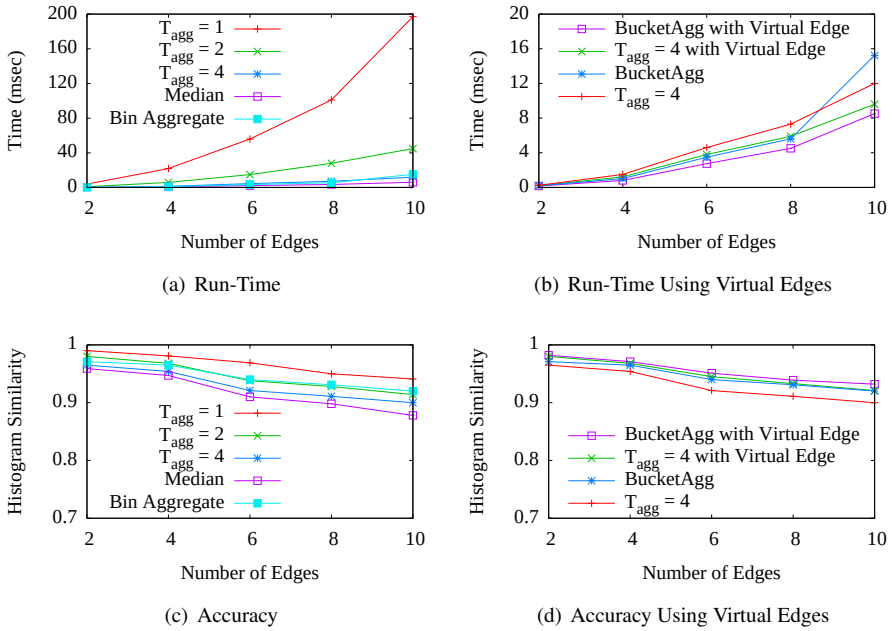
$$Acc(H, dd) = \frac{1}{n} \sum_{i=1}^{n} \frac{|p_i - H.p_k|}{max(p_i, c)}, \tag{2.7}$$

where $H.p_k$ is the probability of value v_i in the histogram, i.e., $v_i \in H.b_k$. A constant c is used to avoid fluctuations introduced by small probabilities. This accuracy metric indicates the relative accumulative deviation from the original distribution.

Figure 2.11(a) shows average Acc values of the initial equi-width histograms, and the histograms after merging with varying merge thresholds. The initial equi-width histograms achieve quite good accuracy and achieve better accuracy as the merge threshold increases.

We also evaluate the accuracy of our three bucket reduction techniques. We choose two sets of merged histograms with merge thresholds 0.95 and 0.98, respectively; and

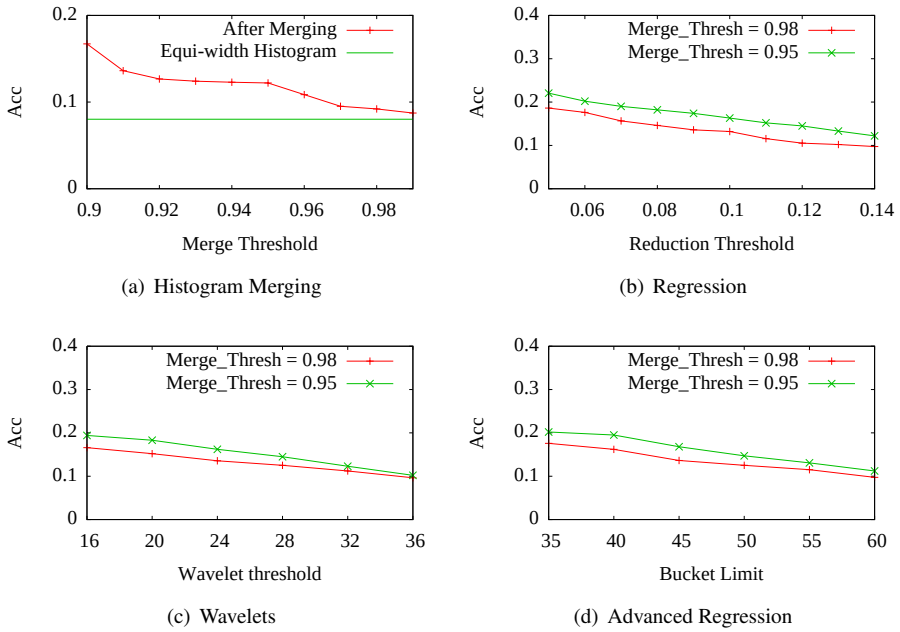
we further conduct bucket reduction on them. Figures 2.11(b)– 2.11(d) show the accuracy using different bucket reduction methods with varying reduction thresholds. All



(a) Run-Time (b) Run-Time Using Virtual Edges

(c) Accuracy (d) Accuracy Using Virtual Edges

Figure 2.10: Histogram Aggregate Study



(a) Histogram Merging (b) Regression

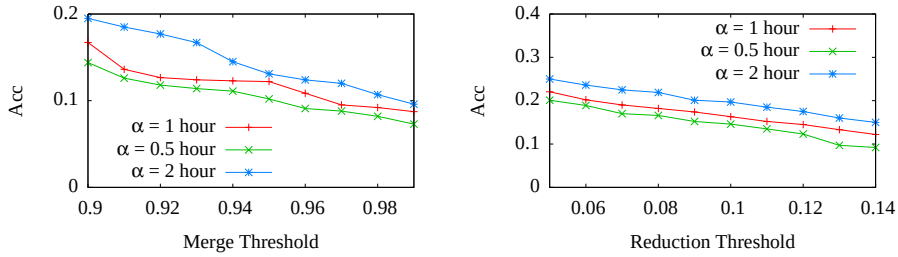
(c) Wavelets (d) Advanced Regression

Figure 2.11: Accuracy Study

three bucket reduction methods suggest the same trend: with higher reduction thresholds, the corresponding histograms achieve better accuracy.

Next, we consider the accuracy of our histogram aggregation techniques. We choose a set of 1,012 routes that are traversed frequently to evaluate the accuracy of the GHG emissions estimates. On average, these routes cover 16 edges. We first use 18 months of data to build an ERN and generate the GHG emissions estimation histograms for all the routes. In our setting, there are at most 24 histograms for a route. Then we use 6 months of GPS data to generate GHG emissions histograms for each route in every time interval with no data compression. We then evaluate the similarity of the histograms derived in the two steps. In Figure 2.10(c), we show the accuracy of three aggregation algorithms when varying β. The figure also shows that with the same setting, we can achieve better accuracy with less computation time when we aggregate histograms for a sequence of edges.

As we set 1 hour as the default time interval when we start building the time dependent histograms, we also study how *Acc* changes while choosing temporal granularity α. Figure 2.12 shows the accuracy we can achieve when we use different temporal granularity. We set merge threshold T_{merge} to 0.95 for all the histogram reduction experiments. The results show that we can get better accuracy if we choose a finer temporal granularity if we have enough GPS records for the time intervals.



(a) Histogram Merging (b) Regression

(c) Wavelets (d) Advanced Regression

Figure 2.12: Accuracy Study II

2.5.4 Storage

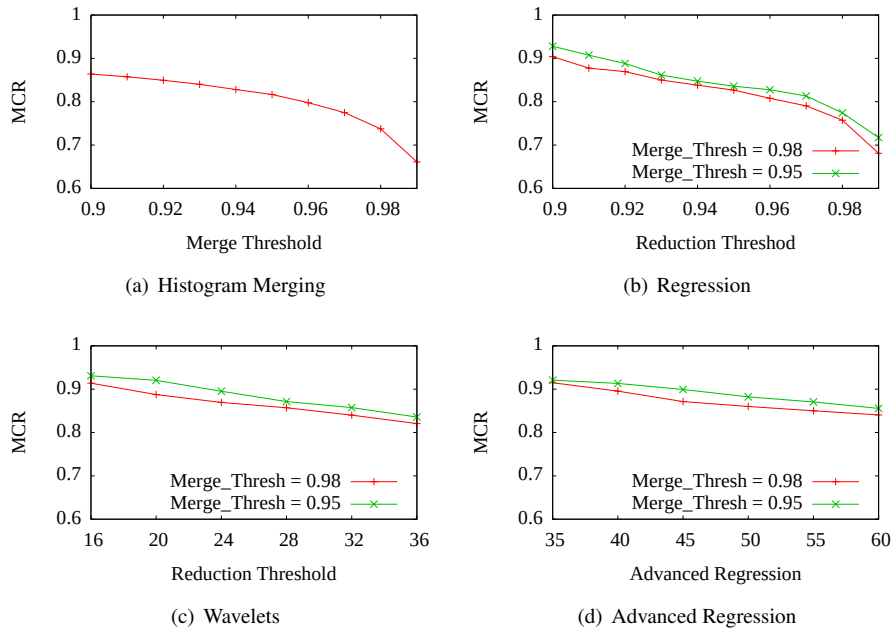
We evaluate how much storage we can save by using histogram merging and bucket reduction. In particular, a bucket in a histogram requires three float values to indicate the lower and upper bound of the bucket and the bucket probability. For a wavelet

histogram, a coefficient takes a float value (i.e., 16 bytes).

We introduce the *memory compression ratio MCR* to measure the storage reduction. In particular, the *MCR* for histogram merging is computed as $\frac{M_{init}-M_{merge}}{M_{init}}$, where M_{init} and M_{merge} represents the required storage to represent the initial time dependent histograms and the merged histograms. The *MCR* for bucket reduction is computed as $\frac{M_{merge}-M_{redu}}{M_{merge}}$, where M_{redu} represents the required storage to represent the histograms after bucket reduction based on merged histograms.

Figure 2.13 shows the *MCR* for both histogram merging and bucket reduction. Figure 2.13(a) shows that, when merge threshold is set to 0.9, the storage required by the initial histograms can be reduced by 85%. When the merge threshold is 0.98, the reduction is 50%. Recall that the accuracy of using the same merge threshold is quite close to the initial histograms (see Figure 2.11(a)).

We fix two sets of merged histograms (with merging threshold 0.95 and 0.98) and test the *MCR* using different bucket reduction methods with varying reduction thresholds. Figures 2.13 (b)–(d) show that bucket reduction can further reduce the required storage based on merged histograms: smaller thresholds achieve better *MCR*.



(a) Histogram Merging (b) Regression

(c) Wavelets (d) Advanced Regression

Figure 2.13: *MCR* Study

Figure 2.14 reports the required storage in order to achieve different accuracies for various approaches. To achieve a higher accuracy (i.e., a smaller *Acc* value), more storage is required. When the accuracy is fixed, Wavelet based histograms consume the least storage; both Wavelet and advanced regression outperforms T-drive, and regression uses slightly more storage than T-drive. Figure 2.13(a) indicates that using 240 MB, we gain the highest accuarcy in our experiment, making it possible to fit our ERN on mobile devices with limited storage space.

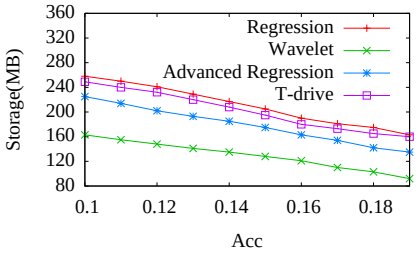


Figure 2.14: *MCR* vs. Accuracy

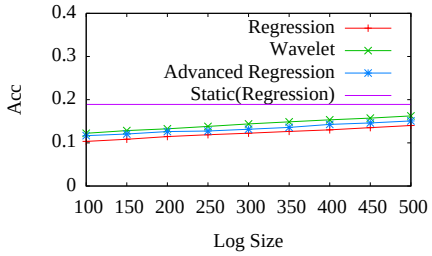


Figure 2.15: Dynamic Maintenance

2.5.5 Dynamic Maintenance

To test the accuracy of dynamic maintenance, we use the first 18 months of trajectories to build an ERN. We then simulate a GPS records stream based on the trajectories in the remaining 6 months. We evaluate the accuracy of the update-to-date histograms using dynamic maintenance with varying *MaxLogSize*. Figure 2.15 shows how dynamic maintenance affects *Acc* values. When the log size is smaller, the histograms are updated more frequently, and thus it results in better estimation accuracy.

2.5.6 Summary

The results covered in this section show that our histogram compression and aggregation techiniques can approximate GHG emissions for edges and routes with acceptable accuracy and limited storage usage. Our dynamic maintenance techinque ensures that we can capture the real-time traffic status on a road network, which makes real-time routing and re-routing possible.

Chapter 3

Travel Cost Inference from Sparse, Spatio-Temporally Correlated Time Series Using Markov Models

This work has been accepted by the 39th International Conference on Very Large Data Bases [2] – one of the best conferences in the data management field.

When monitoring a complex system, the resulting measurements can often be modeled as a collection of time series. These reflect the internal dynamics of the systems and hold the potential for understanding and predicting aspects of the system's behaviour.

While this general scenario applies to a wide variety of settings, we consider the setting of a road network. Figure 3.1 shows two travel time series obtained from GPS records collected from two adjacent road segments in a road network. Each time series records the average cost associated with traversing its segment during different time intervals. The two time series are correlated: (i) the peak on road 2 in the 28-th interval may result in the peak on road 1 in the 29-th interval; (ii) the peak on road 1 in the 35-th interval may cause the peak on road 2's time series in the 36-th interval.

With a travel-cost time series available (e.g., derived from GPS data) for each road in a road network, a mathematical model of the network can be instantiated and then used for predicting the future travel cost behaviour associated with the road network. Thus, it becomes possible to obtain effective routing services in the underlying road network that minimise travel time (e.g., [4]) or GHG emissions, termed eco-routing [7]. However, achieving this is non-trivial, as it is necessary to contend with three challenging characteristics of the time series.

Sparse: Travel-cost time series are built from GPS data obtained from probe vehicles that cannot cover every time interval. As a result, the time series are sparse, meaning that during some intervals, no cost measurements are available for some road segments. For example, both time series in Figure 3.1 lack travel times in the 23-rd and 24-th intervals, and road 2 also lacks travel times in intervals 31–33. Thus, the time series we consider are fundamentally sparse, which is different from regular time series [29, 30], where each interval has a value.

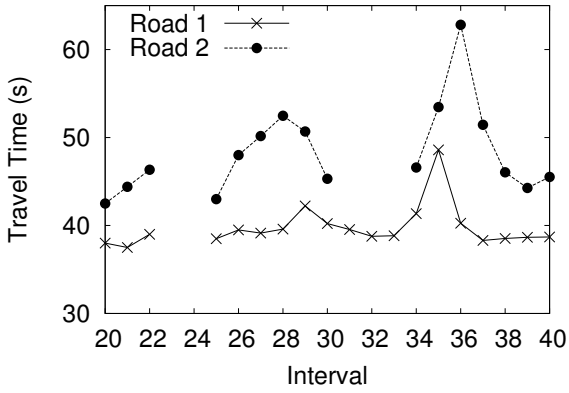


Figure 3.1: Spatio-Temporally Correlated Sparse Time Series

Dependent: The travel costs associated with a road segment are temporally dependent. For example, congestion on a road segment disappears only gradually (see intervals 28–30 of road 2 in Figure 3.1). Likewise, time series are spatially correlated. For example, congestion on one road segment may correlate with congestion on spatially adjacent road segments. This is illustrated by the 28-th interval on road 2 and the 29-th interval on road 1. When modeling the overall travel cost behaviour of a road network, it is important to capture the dependencies within time series and the correlations among different time series.

Heterogenous: Different road segments may exhibit very different behaviours. For example, some segments may have clear morning and afternoon peak hours, while others may not. We model an individual time series as the output generated by a state transition machine (e.g., a Hidden Markov Model, HMM [29, 30]) that operates on a set of states. For instance, the traffic on a road segment may change between two states, e.g., *clear* and *congested*, and each state may output different travel costs. A prerequisite of using HMMs is that the cardinality of the state set that the underlying process operates on is given a priori. Due to the heterogeneity across time series, it is not feasible to use the same number of states for every time series. Rather, each time series must have its own state set.

We model multiple travel-cost time series using a *Spatio-Temporal Hidden Markov Model* (*STHMM*) while taking into account the above three challenging characteristics. We compress a sparse time series into a compact time series and identify a state set for each compact time series. The dependencies within a time series and the correlations among different time series are modeled by connecting pertinent states of the road segments while considering the topology of the road network. Smoothing techniques are applied to reduce the effects of data sparsity. Finally, future travel costs are inferred based on the learned STHMM.

We believe that this is the first study to provide general techniques, including state formulation and parameter learning, that contend with sparse, heterogenous, and spatio-temporally correlated time series in a combined manner. This paper is accepted as full research paper at VLDB, 2013. The paper makes four contributions. First, it proposes a general framework that is capable of modeling the traffic behaviour of a road network and thereby enables routing services that optimise different travel-related costs. Second, a spatio-temporal hidden Markov model is formalised to model the

traffic behaviour of a road network. Third, learning algorithms are proposed to obtain individual state sets for all road segments and to determine the parameters needed to configure an STHMM. Fourth, comprehensive experiments are conducted to elicit the design properties of the proposed framework and algorithms.

The remainder of this paper is organised as follows. Section 4.2 covers preliminaries. Section 3.2 defines an STHMM, and Section 3.3 describes the learning algorithms needed for determining an STHMM. Section 4.4 reports on the empirical evaluation. Finally, related work is covered in Section 4.1, and conclusions and research directions are offered in Section 3.6.

3.1 Preliminaries

3.1.1 Road Network Model

A road network is modeled as a directed, labeled graph $G = (V, E, W)$, where V and $E \subseteq V \times V$ is a vertex set and an edge set, respectively, and $W : E \times \mathbb{T} \rightarrow \mathbb{R}^+$ captures time dependent edge weights. A vertex $v_i \in V$ models a road intersection or an end of a road. An edge $e_k = (v_i, v_j) \in E$ models a directed road segment, indicating that travel is possible from its source v_i to its destination v_j. We use the notation $e_k.s$ and $e_k.t$ to denote the source and destination of edge e_k. Figure 3.2 shows a road network with 5 vertices and 6 edges.

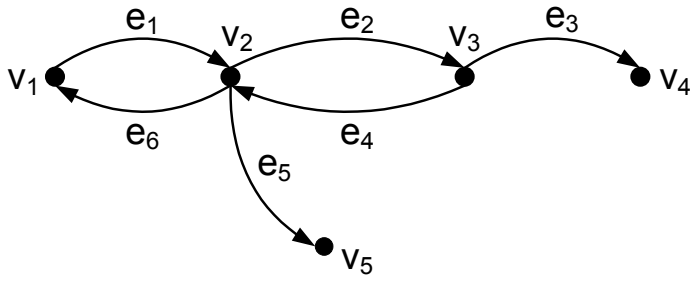


Figure 3.2: A Simple Road Network

3.1.2 Travel-Cost Time Series

We construct time series from GPS data obtained from probe vehicles. We use GPS data instead of road side sensor data because GPS data is becoming available in increasing volumes, is less costly to obtain, and provides good coverage of a road network. However, the proposed techniques also apply to time series constructed from road-side sensor data, as they share the same characteristics.

A trajectory $\mathscr{T} = \langle p_1, p_2, \ldots, p_X \rangle$ is a sequence of GPS records pertaining to a particular trip, where each record p_i specifies a $(location, time)$ pair of the vehicle. With the help of map matching [5], a GPS record can typically be mapped to a specific location in the underlying road network.

Map matching transforms a trajectory $\mathscr{T}$ into a sequence of cost records $\langle l_1, l_2, \ldots, l_m \rangle$, where each cost record l_j is of the form $(e, t_s, cost)$, where e is an edge traversed by trajectory $\mathscr{T}$, t_s is the time at which traversing edge e starts, and $cost$ is the travel cost of traversing edge e. Some travel costs, notably travel times, can be obtained directly

from the GPS records, while other travel costs, e.g., GHG emissions, can be derived from the GPS records [7].

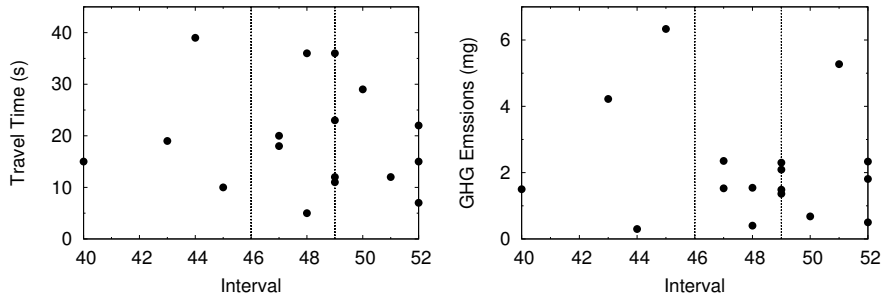
Next, we introduce a parameter α that specifies the finest interval to be used in defining time series. Given a GPS data set collected during Z days, we then have $T = \lceil \frac{Z \cdot 24 \cdot 60}{\alpha} \rceil$ intervals. For example, we may use $\alpha = 15$ minutes because this is typically the finest time granularity used in the transportation area [31]. When $Z = 30$ days, we then have $T = 2,880$ intervals.

We define the travel-cost time series on edge e_i as

$$\mathscr{TS}_i = \langle C_i^{(1)}, C_i^{(2)}, \ldots, C_i^{(T)} \rangle,$$

which is a sequence of T travel cost sets. Travel cost set $C_i^{(t)} = \{l_j.cost | l_j.e = e_i \wedge l_j.t_s \in [(t-1)\cdot\alpha, t\cdot\alpha)\}$, contains the travel costs observed in the t-th interval $[(t-1)\cdot\alpha, t\cdot\alpha)$. If no GPS records are collected on e_i during the t-th interval, $C_i^{(t)}$ is empty.

Figure 3.3 shows travel-time based and GHG-emissions based time series for the same edge, where α is set to 15 minutes. Both time series are sparse: for example, the 46-th interval contains no data. Figure 3.3 also illustrates that an average value for an interval [29] cannot capture the traffic behaviour during the interval, which may vary considerably throughout the interval. In the example, the travel time in the 49-th interval varies considerably. In the proposed STHMM, the travel cost distribution in the t-th interval is modeled based on $C_i^{(t)}$.



(a) Travel Time (b) GHG Emissions

Figure 3.3: Examples of Travel-Cost Time Series

The distributions of travel time and GHG emissions may be quite different (e.g., see the 49-th intervals in Figures 3.3(a) and (b)). A general technique that is able to support different travel costs must handle such variation. Section 4.4 shows that the proposed STHMM is able to predict both travel time and GHG emissions.

3.1.3 Framework Overview

In short, we aim to update time-dependent edge weights in G based on real-time travel cost information. Figure 4.4 gives an overview of the system, which has three major components: offline learning, online inference, and routing services.

In offline learning, a state formulation and parameter learning module takes as input a collection of time series that are obtained from historical GPS trajectories. It outputs an STHMM. As real-time GPS records stream in, the online inference component infers near-future travel costs by using the learned STHMM. The time-dependent edge

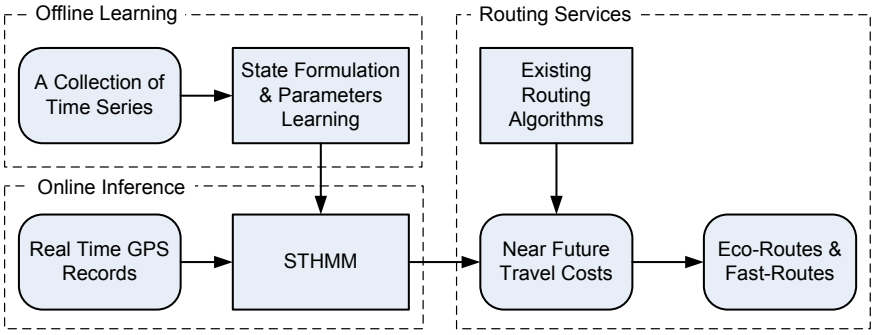


Figure 3.4: Framework Overview

weights in road network G can be updated based on the inferred travel costs. In the routing services, any routing algorithms that support time-dependent edge weights [4, 32, 33] can be applied to road network G to enable various kinds of routing such as eco-routing or time-based routing.

3.2 Travel Cost Modeling

We employ HMMs to capture the temporal dynamics of the travel costs on individual edges; then we define a *Spatio-Temporal Hidden Markov Model (STHMM)* to model the spatio-temporal correlation of travel costs among all edges in a road network.

3.2.1 Modeling Temporal Dependence

Without loss of generality, we assume that the travel costs of an edge are affected by the underlying traffic on the edge and vary as the traffic changes over time.

Consider a particular edge e_i. A state set $S_i = \{s_i^{(x)}\}$ models all possible traffic conditions on edge e_i. For example, states $s_i^{(1)}$ and $s_i^{(2)}$ may model congestion and clear, respectively. We use a state sequence $\mathcal{Q}_i = \langle q_i^{(1)}, q_i^{(2)}, \ldots, q_i^{(T)} \rangle$ to model the transition of the traffic condition on e_i from one state to another over the T intervals. State $q_i^{(t)} \in S_i$ $(1 \leqslant t \leqslant T)$ models the traffic condition in the t-th interval, and it depends only on its previous state $q_i^{(t-1)}$, i.e., the traffic condition in the $(t-1)$-st interval. In Figure 3.5, $q_i^{(t)} = s_i^{(1)}$ indicates that edge e_i is in state congestion in the t-th interval, and it is dependent on the state in the $(t-1)$-st interval $q_i^{(t-1)} = s_i^{(1)}$, which is also in congestion.

We use a travel-cost time series $\mathcal{TS}_i = \langle C_i^{(1)}, C_i^{(2)}, \ldots, C_i^{(T)} \rangle$ to model the variation of the travel costs on edge e_i during the T intervals. Here, $C_i^{(t)}$ contains the travel costs observed in the t-th interval and is dependent on the corresponding traffic conditions, i.e., state $q_i^{(t)}$. Figure 3.5 shows that the travel costs in $C_i^{(t)}$ generally exceed those in $C_i^{(t+1)}$. This corresponds to the traffic in the t-th interval being congested, while the traffic is clear in the $(t+1)$-st interval.

Based on the above, we use a Hidden Markov Model (HMM) [34] to model the relationship between traffic conditions and travel costs. An HMM is a statistical model

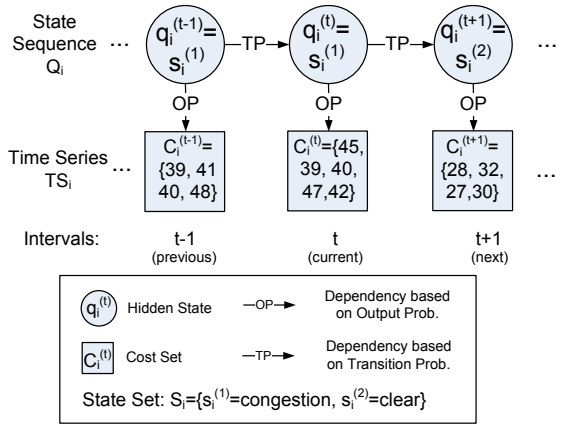


Figure 3.5: Traffic Conditions and Travel Costs on Edge e_i

for modeling a time series, e.g., the travel-cost time series $\mathscr{TS}_i$ on edge e_i, which is generated by a Markov process.

A Markov process operates on a set of hidden states (e.g., a traffic state set S_i) to generate a state sequence (e.g., $\mathscr{Q}_i$). A state in the state sequence depends only on its previous state, and a state generates an output, thus resulting in the time series. In our setting, the traffic condition in the t-th interval only depends on the traffic conditions in the $(t-1)$-st interval, and travel costs in the t-th interval depend on the traffic conditions in the t-th interval.

An HMM is defined in terms of three parameters PI, A, and B:

1. Initial probabilities $PI = \{\pi^{(x)}\}$, $1 \leqslant x \leqslant |S_i|$, where $\pi^{(x)} = \mathbf{P}(q_i^{(1)} = s_i^{(x)})$ is the probability that state sequence $\mathscr{Q}_i$ starts with state $s_i^{(x)}$.

2. Transition probabilities $A = \{a^{(x,y)}\}$, $1 \leqslant x,y \leqslant |S_i|$, where $a^{(x,y)} = \mathbf{P}(q_i^{(t)} = s_i^{(y)} | q_i^{(t-1)} = s_i^{(x)})$ is the probability that state $s_i^{(x)}$ transits to state $s_i^{(y)}$ in the next step.

3. Output probabilities $B = \{b^{(x)}(c)\}$, $1 \leqslant x \leqslant |S_i|$, where $b^{(x)}(c) = \mathbf{P}(c \in C_i^{(t)} | q_i^{(t)} = s_i^{(x)})$ is the probability that cost value c is observed given the state is $s_i^{(x)}$.

Note that only the time series $\mathscr{TS}_i$ is observable from the available GPS records, while the states are hidden and cannot be observed directly. The formulation of state set S_i is detailed in Section 3.3.1. The state sequence $\mathscr{Q}_i$ and the time series $\mathscr{TS}_i$ are interrelated. Given the travel costs $C_i^{(t)}$ in the t-th interval, the corresponding state $q_i^{(t)}$ can be inferred using output probabilities. The state in the next interval $q_i^{(t+1)}$ can then be predicted from $q_i^{(t)}$ and the transition probabilities. The travel costs at the next interval, $C_i^{(t+1)}$, can in turn be inferred from $q_i^{(t+1)}$ and output probabilities. The derivation of these probabilities is detailed in Section 3.3.2.

3.2.2 Modeling Spatio-Temporal Dependence

The state of an edge influences not only its own next state, but also the next states of nearby edges. For example, an edge is more likely to be congested if its neighboring

edges are congested. To model the spatio-temporal dependencies of travel costs over an entire road network, we define an STHMM.

3.2.2.1 N-th Order Neighbors

To define an STHMM, we need the concept of the n-th order neighbors of an edge e_i, denoted as $L_i^{(n)}$. Given an edge e_i, its 1-st order neighbors $L_i^{(1)}$ is defined in Equation 3.1.

$$L_i^{(1)} = \{e_i\} \bigcup \quad \{e_j | (e_j.t = e_i.s \vee e_j.s = e_i.t) \wedge \neg(e_j.t = e_i.s \wedge e_j.s = e_i.t)\} \tag{3.1}$$

Here, $L_i^{(1)}$ includes edge e_i itself and the edges that share a source or target vertex with e_i, but excludes the oppositely directed edge that corresponds to the same physical road segment as e_i. Considering Figure 3.2, $L_2^{(1)} = \{e_1, e_2, e_3\}$, since edges e_2 and e_1 share v_2, and edges e_2 and e_3 share v_3. Edge e_4 is not in $L_2^{(1)}$ as it is the oppositely directed edge.

The n-th order $(n > 1)$ neighbors of edge e_i are defined recursively as $L_i^{(n)} = \bigcup_{e_j \in L_i^{(n-1)}} L_j^{(1)}$. For example, since $L_1^{(1)}$=$\{e_1, e_2, e_5\}$, $L_1^{(2)}$=$L_1^{(1)} \cup L_2^{(1)} \cup L_5^{(1)}$= $\{e_1, e_2, e_5\} \cup \{e_1, e_2, e_3\} \cup \{e_1, e_4, e_5\}$ = $\{e_1, e_2, e_3, e_4, e_5\}$.

3.2.2.2 Spatio-Temporal Hidden Markov Model

To model the traffic evolution in an entire road network, we need to consider the interactions among the traffic on all edges. A naive model is to treat different edges' traffic evolutions as independent: the traffic on an edge evolves by itself and does not interact with the traffic on other edges. This yields $|E|$ individual HMMs, each of which models the traffic on a single edge. This naive model fails to capture any interactions among the traffic on different edges and is unable to properly model global traffic changes.

We proceed to define an STHMM that couples the traffic of n-th order neighbors to model the interactions among these edges. An n-th order STHMM couples the HMM of an edge with the HMMs of its n-th order neighbors. In an n-th order STHMM, edge e_i's current state $q_i^{(t)}$ is not only dependent on its previous state $q_i^{(t-1)}$, but also on the previous state of each of its n-th order neighbors, i.e., all the states $q_j^{(t-1)}$ where $e_j \in L_i^{(n)}$.

The 1-st order STHMM of the road network in Figure 3.2 is shown in Figure 3.6, where only states are shown and the time series are omitted. Taking edge e_1 as an example, since $L_1^{(1)}$=$\{e_1, e_2, e_5\}$, state $q_1^{(t)}$ is dependent on states $q_1^{(t-1)}$, $q_2^{(t-1)}$, and $q_5^{(t-1)}$. Thus, states $q_1^{(t-1)}$, $q_2^{(t-1)}$, and $q_5^{(t-1)}$ are connected to $q_1^{(t)}$ in Figure 3.6.

3.2.2.3 Parameter Space

In an STHMM, edge e_i's transition probability is a conditional probability conditioned on the previous states of its n-th order neighbors $e_{i_j} \in L_i^{(n)}$. Thus, we have $\mathbf{P}(q_i^{(t)} | q_{i_1}^{(t-1)}, \ldots, q_{i_k}^{(t-1)})$, where $q_{i_j}^{(t-1)} (1 \leqslant j \leqslant k$ and $k = |L_i^{(n)}|)$ is the previous states of edge e_{i_j}. Considering the STHMM in Figure 3.6, the transition probability factors of edges e_1 and e_6 are $\mathbf{P}(q_1^{(t)} | q_1^{(t-1)}, q_2^{(t-1)}, q_5^{(t-1)})$ and $\mathbf{P}(q_6^{(t)} | q_4^{(t-1)}, q_6^{(t-1)})$, respectively.

An STHMM is governed by parameters TAU, H, and D:

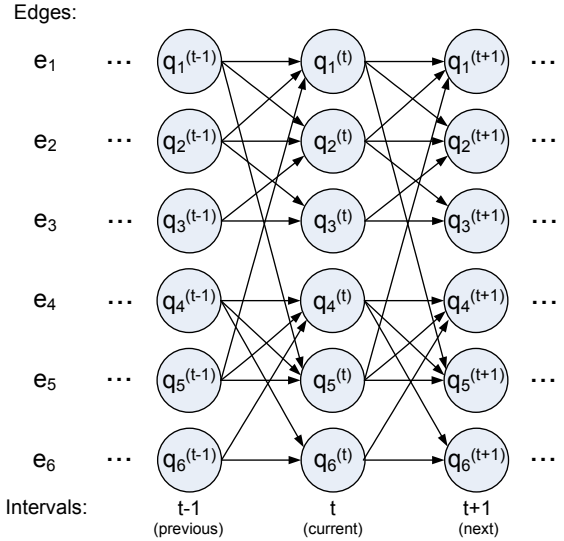


Figure 3.6: 1-st Order Spatio-Temporal Hidden Markov Model

1. Initial probabilities: $TAU = \{\tau_i^{(x)}\}$, $1 \leqslant i \leqslant |E|$, $1 \leqslant x \leqslant |S_i|$, where $\tau_i^{(x)}$=$\mathbf{P}(q_i^{(1)} = s_i^{(x)})$ is the probability that edge e_i starts with state $s_i^{(x)}$.

2. Transition probabilities: $H = \{h_{i,i_1,\ldots,i_k}^{x_i,x_{i_1},\ldots,x_{i_k}}\}$, $1 \leqslant i \leqslant |E|$, $1 \leqslant x_i \leqslant |S_i|$, $e_{i_j} \in L_i^{(n)}, 1 \leqslant x_{i_j} \leqslant |S_{i_j}|, 1 \leqslant j \leqslant k = |L_i^{(n)}|$. Further, $h_{i,i_1,\ldots,i_k}^{x_i,x_{i_1},\ldots,x_{i_k}} = \mathbf{P}(q_i^{(t)} = s_i^{(x_i)} | q_{i_1}^{(t-1)} = s_{i_1}^{(x_{i_1})}, \ldots, q_{i_k}^{(t-1)} = s_{i_k}^{(x_{i_k})})$ is the probability that the current state of edge e_i is $s_i^{(x_i)}$ given the previous states of its n-th order neighbors $e_{i_1}, \ldots, e_{i_k}$ being $s_{i_1}^{(x_{i_1})}, \ldots, s_{i_k}^{(x_{i_k})}$.

3. Output probabilities: $D = \{d_i^{(x)}(c)\}$, $1 \leqslant i \leqslant |E|$, $1 \leqslant x \leqslant |S_i|$, where $d_i^{(x)}(c) = \mathbf{P}(c \in C_i^{(t)} | q_i^{(t)} = s_i^{(x)})$ is the probability of observing output c from edge e_i given that its state is $s_i^{(x)}$.

3.3 Learning an STHMM

To obtain an STHMM instance that appropriately models the dynamics of traffic in a road network, we must formulate hidden states for all edges, and we must instantiate the parameters that govern the STHMM. Figure 3.7 gives an overview of the whole process.

The process of learning an STHMM starts with the state formulation phase (Section 3.3.1). For each edge $e_i \in E$, its sparse time series $\mathscr{TS}_i$ is compressed into a compact time series $\overline{\mathscr{TS}}_i$. The state formulation module clusters the travel cost sets in compact time series $\overline{\mathscr{TS}}_i$ into a set of clusters, where each cluster indicates a state. Thus, state set S_i for edge e_i is obtained. In addition, the output probabilities are determined.

The next phase is parameter learning (Section 3.3.2). For each edge e_i, based on the obtained state set S_i, a state estimation module estimates possible states for each interval in the sparse time series $\mathscr{TS}_i$, thus obtaining an uncertain state sequence $\mathscr{Q}_i$.

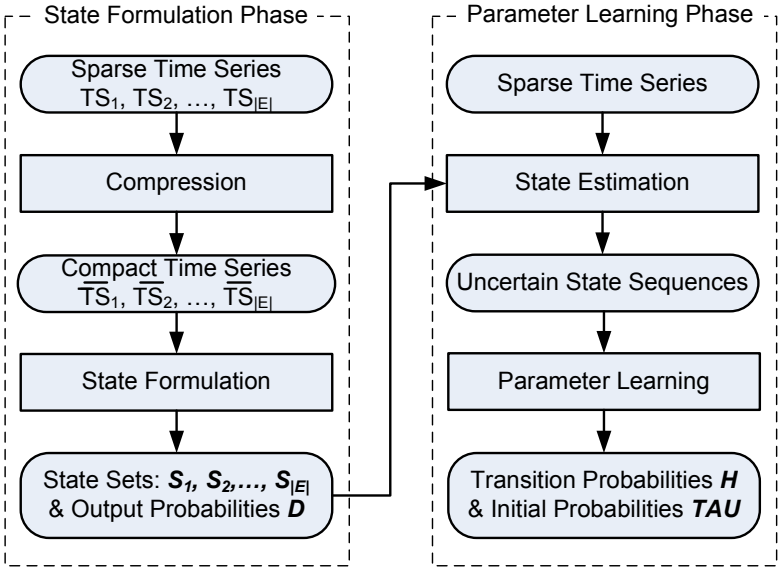


Figure 3.7: Overview of the STHMM Learning Process

Coupling the uncertain state sequences according to their n-th order neighbors, the transition and initial probabilities of the STHMM are determined. Finally, we describe how to use a learned STHMM to infer travel costs (Section 3.3.3).

3.3.1 State Formulation

3.3.1.1 Compact Time Series

The time series $\mathscr{TS}_i = \langle C_i^{(1)}, \ldots, C_i^{(T)} \rangle$ on edge e_i may be sparse because many or some of the $C_i^{(t)}$ may contain no or few travel costs due to the lack of GPS records in the corresponding intervals. To contend with this sparsity, we compress $\mathscr{TS}_i$ into a compact time series $\overline{\mathscr{TS}}_i$ in which each interval contains relatively more travel costs, which renders subsequent analysis (detailed in Sections 3.3.1.2 and 3.3.1.3) easier and more effective.

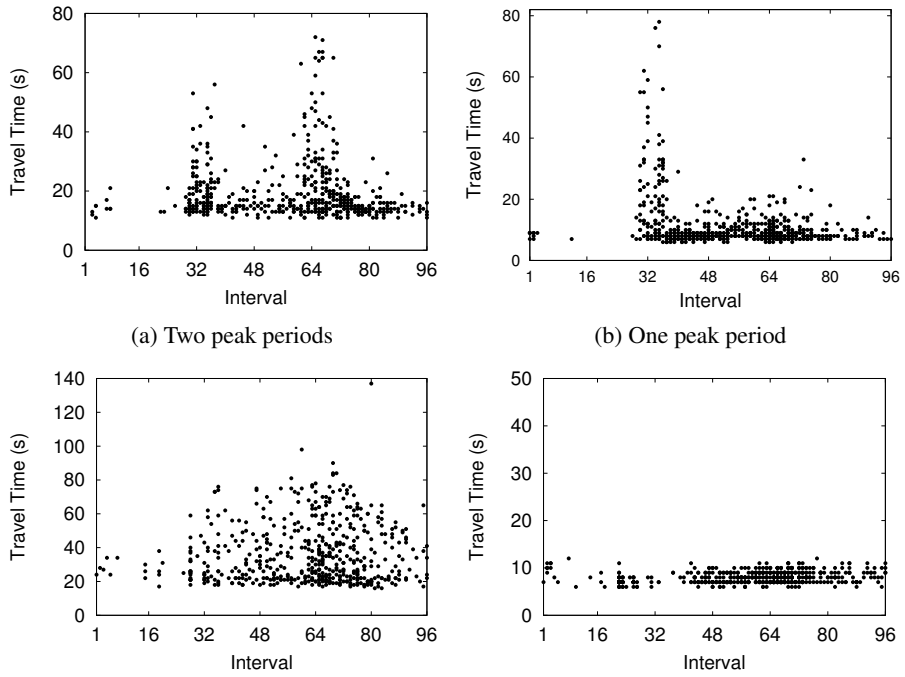
Recall that $\mathscr{TS}_i$ records travel costs on e_i during Z days or $T = \lceil \frac{Z \cdot 24 \cdot 60}{\alpha} \rceil$ intervals. Its compressed version $\overline{\mathscr{TS}}_i$ consolidates these costs into a period of interest P that contains $M = \lceil \frac{P}{\alpha} \rceil$ intervals. In the example in Section 3.1.2, $Z = 30$ days, $\alpha = 15$ minutes, and $T = 2,880$ intervals. With P being a 24-hour period, $\overline{\mathscr{TS}}_i$ contains $M = 96$ intervals.

Next, we define a compact time series $\overline{\mathscr{TS}}_i$ as follows.

$$\overline{\mathscr{TS}}_i = \langle \overline{C}_i^{(1)}, \overline{C}_i^{(2)}, \ldots, \overline{C}_i^{(M)} \rangle, \text{ where } \overline{C}_i^{(x)} = \bigcup_{k \bmod M = x} C_i^{(k)}$$

Intuitively, set $\overline{C}_i^{(1)}$ in $\overline{\mathscr{TS}}_i$ contains all travel costs observed on edge e_i in the interval $[0{:}00, 0{:}15)$ during each of the Z days. Thus, set $\overline{C}_i^{(1)}$ is the union of $C_i^{(1)}, C_i^{(97)}, C_i^{(193)}, \ldots, C_i^{(2785)}$ in $\mathscr{TS}_i$.

Figure 3.8 shows four compact travel time series obtained from four different edges. The travel times in Figure 3.8(a) exhibit clear morning and afternoon peaks, while the times shown in Figure 3.8(b) have only a clear morning peak. The travel times in Figure 3.8(c) do not display a clear trend of variation, but vary significantly, from 16 seconds to 137 seconds. The travel times in Figure 3.8(d) remain almost constant at around 8 seconds.



(a) Two peak periods

(b) One peak period

(c) No clear peak periods

(d) No obvious variations

Figure 3.8: Compact Travel Time Series

Figure 3.8 suggests that the traffic on different edges evolves quite differently. In order to use an STHMM to model the traffic dynamics, it is thus important that each edge e_i can be given its own state set S_i. To achieve this, we propose a *Time Sensitive Gaussian Mixture* (*TSGM*) method to formulate each S_i, where the obtained states satisfy two properties: (i) in a state, the traffic on an edge behaves similarly, thus making the travel cost follow a distribution; (ii) a state is dependent on its previous state. As shown in Algorithm 7, the *TSMG* method consists of two steps, *cost clustering* and *time-cost clustering*, detailed below.

Algorithm 7: TSMG

Input : double: λ; CompTimeSeries: $\overline{\mathcal{TS}}_1 \ldots \overline{\mathcal{TS}}_{|E|}$;
Output: State sets for all edges: $S_1, S_2, \ldots, S_{|E|}$;

1 **for** *each edge* e_i *in the road network* **do**
2 $\quad GMM_i \leftarrow CostClustering(\overline{\mathcal{TS}}_i)$;
3 $\quad S_i \leftarrow TimeCostClustering(\overline{\mathcal{TS}}_i, GMM_i, \lambda)$;

3.3.1.2 Cost Clustering

For each edge e_i, the cost clustering step identifies a probability density function (pdf) that describes the distribution of all travel costs in $\overline{\mathscr{TS}}_i$, regardless of the interval in which they are observed. As a Gaussian Mixture Model (GMM) is able to approximate any complex pdf [30], the cost clustering step identifies a GMM, denote as GMM_i, to describe the distribution of all travel costs observed on e_i (line 2 in Algorithm 7).

Specifically, GMM_i is determined by grouping all the cost values in $\overline{\mathscr{TS}}_i$ into K_i clusters. Each cluster indicates a representative group of travel costs, whose distribution is described by a single Gaussian distribution, termed a Gaussian component. Equation 3.2 gives the formal definition.

$$GMM_i(c) = \sum_{k=1}^{K_i} m_{i,k} \cdot \mathscr{N}(c|\mu_{i,k}, \delta_{i,k}^2) \tag{3.2}$$

Here, $m_{i,k}$ is the k-th mixing coefficients of the k-th Gaussian components, and these satisfy $\sum_{k=1}^{K_i} m_{i,k} = 1$; and the k-th Gaussian component $\mathscr{N}(c|\mu_{i,k}, \delta_{i,k}^2)$ has mean $\mu_{i,k}$ and variance $\delta_{i,k}^2$.

If K_i is given in advance, basic clustering algorithms, e.g., K-Means, can be applied directly. However, deciding an appropriate K_i before the step starts is difficult, and the obtained clusters may not be optimal. An overly small K_i may not fully capture all the representative travel costs on the edge, thus resulting in under-fitting; and an overly large K_i may capture the travel costs over-specifically, yielding over-fitting [30].

We apply the procedure given in Algorithm 8 to select an appropriate K_i. It starts with $K_i = 1$ and then increments K_i by 1 until the benefit (e.g., likelihood) of using K_i is smaller than that of using $K_i - 1$.

Algorithm 8: CostClustering

Input : CompTimeSeries: $\overline{\mathscr{TS}}_i = \langle \overline{C}_i^{(1)}, \ldots, \overline{C}_i^{(M)} \rangle$;
Output: GaussianMixtureModel: GMM_i;

```
1  GMM preGMM ← null, newGMM ← null;
2  double preLH ← −∞, newLH ← −∞; int k ← 0;
3  CC ← ⋃_{x=1}^{M} C̄_i^(x);
4  Split CC into f equal subsets cc[1], ..., cc[f];
5  repeat
6      preGMM ← newGMM;
7      preLH ← newLH; newLH ← 0;
8      k ← k + 1;
       /* f-fold likelihood evaluation                    */
9      for j = 1 ... f do
10         train ← CC \ cc[j]; test ← cc[j];
11         newGMM ← EstimateGMM(train, k);
12         newLH ← newLH + EvaLH(test, newGMM);
13 until newLH ⩽ preLH;;
14 GMM_i ← preGMM;
15 return GMM_i;
```

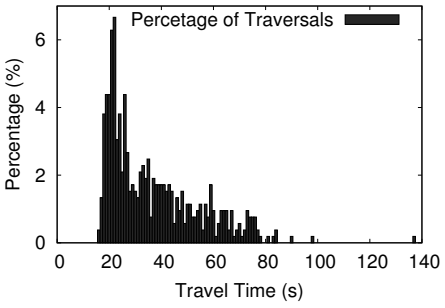
All the cost values in $\overline{\mathscr{TS}}_i$ are recorded in CC, and CC is split into f equal subsets

(lines 3–4). In the k-th iteration (lines 5–13), a GMM *newGMM* with $K_i = k$ Gaussian components and the likelihood *newLH* of using *newGMM* are obtained.

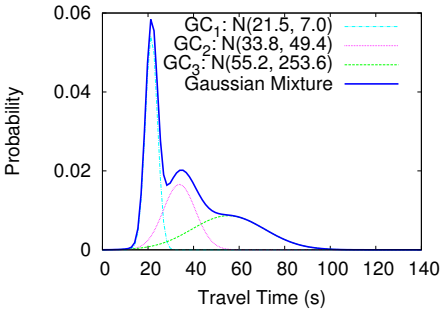
The likelihood of using k Gaussian components is evaluated using f-fold cross validation (lines 9–12). Each of f subsets is used as a testing set *test* once, and each of the remaining $f-1$ subsets is used as a training set *train* (line 10). The parameters of a Gaussian mixture model *newGMM* with k Gaussian components are estimated based on the training set *train* using a classical Expectation-Maximisation algorithm [30] (*EstimateGMM*$(train,k)$, line 11). The likelihood that the test data *test* is generated by the GMM *newGMM* is evaluated (*EvaLH*$(test,newGMM)$ in line 12).

Consider the edge shown in Figure 3.8(c), where travel costs range from 0 to 140. Figure 3.9(a) shows the percentage of traversals (on the y-axis) of the edge with each cost value (on the x-axis). For instance, the highest bar indicates that 6.7% of the traversals took 22 seconds. By applying cost clustering, it is found that a GMM with $K_i = 3$ Gaussian components best describes the travel-time distribution—see Figure 3.9(b).

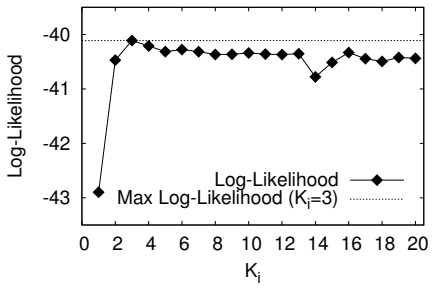
By using 10-fold cross validation, the likelihoods of choosing different numbers of K_i are reported in Figure 3.9(c), which indicates that $K_i = 3$ is the optimal choice. This example also suggests that although it is infeasible to model travel-cost distributions by a single pdf (e.g., Gaussian, uniform or exponential distribution), properly chosen GMMs can describe such distributions very well.



(a) Percentage of Traversals



(b) GMM with $K_i = 3$



(c) Likelihood vs Varying K_i

GC_1, GC_2, GC_3, KL
s_i^1: [0.35, 0.30, 0.35, 0.031]
s_i^2: [0.33, 0.45, 0.22, 0.047]
s_i^3: [0.64, 0.35, 0.01, 0.602]
s_i^4: [0.15, 0.62, 0.23, 0.306]
s_i^5: [0.31, 0.08, 0.61, 0.132]
s_i^6: [0.16, 0.29, 0.55, 0.076]
s_i^7: [0.62, 0.01, 0.37, 0.093]
s_i^8: [0.53, 0.18, 0.29, 0.066]

(d) Final States

Figure 3.9: State Formulation

3.3.1.3 Time-Cost Clustering

For each edge e_i, given the GMM GMM_i with K_i Gaussian components, time-cost clustering (line 3 in Algorithm 7) identifies the state set S_i on e_i. First, each $\overline{C}_i^{(x)}$ in the compact time series $\overline{\mathscr{TS}}_i$ is transformed into a K_i+1 dimensional point. Such a point captures both travel-cost distribution in an interval and the temporal dependency with its previous interval. Thus, $\overline{\mathscr{TS}}_i = \langle \overline{C}_i^{(1)}, \ldots, \overline{C}_i^{(M)} \rangle$ is transformed into M K_i+1 dimensional points $\mathbf{f}_i^{(1)}, \ldots, \mathbf{f}_i^{(M)}$. Second, these points are clustered, such that each cluster refers to a state on edge e_i.

Transforming Compact Time Series to Points: In cost clustering, we identified K_i Gaussian components describing K_i representative travel-cost groups on edge e_i. The travel costs in a particular interval on the edge should also follow the K_i representative travel cost groups. Thus, the distribution of the travel costs in the x-th interval $\overline{C}_i^{(x)}$ is estimated with a new GMM $GMM_i^{(x)}$ with the K_i Gaussian components determined in the cost clustering phase, as defined by Equation 3.3.

$$GMM_i^{(x)}(c) = \sum_{k=1}^{K_i} \hat{m}_{i,k}^{(x)} \cdot \mathscr{N}(c|\mu_{i,k}, \delta_{i,k}^2) \tag{3.3}$$

Here, $\mathscr{N}(c|\mu_{i,k}, \delta_{i,k}^2)$ is the k-th Gaussian component, as in Equation 3.2. However, the $\hat{m}_{i,k}^{(x)}$ are new mixing coefficients that satisfy $\sum_{k=1}^{K_i} \hat{m}_{i,k}^{(x)} = 1$.

Although $GMM_i^{(x)}$ and GMM_i share the same K_i Gaussian components, the mixing coefficients in the two GMMs are typically different because the travel costs observed in each interval typically differ. For example, consider the edge in Figure 3.9(b). $GMM_i^{(x)}$ may have a larger coefficient for Gaussian component $\mathscr{N}(21.5, 7.0)$ for an offpeak interval, but a smaller coefficient for a peak interval.

Next, each $\overline{C}_i^{(x)}$ is transformed into a K_i+1 dimensional point:

$$\mathbf{f}_i^{(x)} = (\hat{m}_{i,1}^{(x)}, \ldots, \hat{m}_{i,K_i}^{(x)}, KL_i^{(x)})$$

The first K_i coordinates are the new mixing coefficients, and the last coordinate $KL_i^{(x)}$ measures the differences between the distribution of the costs in the x-th interval $\overline{C}_i^{(x)}$ and the distribution of the costs in its previous interval $\overline{C}_i^{(x-1)}$, which reflects the temporal dependency of the distributions of costs in two adjacent intervals. In particular, $KL_i^{(x)}$ is evaluated based on Kullback-Leibler divergence [30], as defined in Equation 3.4.

$$KL_i^{(x)} = \int_{-\infty}^{+\infty} GMM_i^{(x-1)}(c) \cdot \ln \frac{GMM_i^{(x-1)}(c)}{GMM_i^{(x)}(c)} \, \mathrm{d}c \tag{3.4}$$

Algorithm 9 transforms a compact time series to a set of points by identifying a point for each $\overline{C}_i^{(x)}$ in $\overline{\mathscr{TS}}_i$. First, new mixing coefficients in $GMM_i^{(x)}$ are initialised as the coefficients in GMM_i that is obtained from cost clustering (lines 2–3).

If the x-th interval is *sparse*, i.e., $\overline{C}_i^{(x)}$ contains fewer than a threshold of *mCounts* cost values, the procedure skips re-estimating the new mixing coefficients. Rather than using a new GMM re-estimated based on the few cost values in a sparse interval, it is better to use the original GMM_i, which captures the common behaviour over the whole

Algorithm 9: CompactTimeSeriesToPoints

Input : CompTimeSeries: $\overline{\mathscr{TS}}_i = \langle \overline{C}_i^{(1)}, \ldots, \overline{C}_i^{(M)} \rangle$;
GMM: GMM_i;
Output: Point Set: F_i;

1 **for** $x = 1 \ldots M$ **do**
2 **for** $k = 1 \ldots GMM_i.K_i$ **do**
3 $GMM_i^{(x)}.\hat{m}_{i,k}^{(x)} \leftarrow GMM_i.m_{i,k}$;
4 **if** $|\overline{C}_i^{(x)}| > mCounts$ **then**
5 **repeat**
6 Initialise an array $sum[GMM_i.K_i]$;
7 **for** *each cost value* $c \in \overline{C}_i^{(x)}$ **do**
8 Initialise an array $lh[GMM_i.K_i]$;
9 **for** $k = 1 \ldots GMM_i.K_i$ **do**
10 $lh[k] \leftarrow GMM_i^{(x)}.\hat{m}_{i,k}^{(x)} \cdot \mathscr{N}(c|\mu_{i,k}, \delta_{i,k}^2)$;
11 **for** $k = 1 \ldots GMM_i.K_i$ **do**
12 $lh[k] \leftarrow \frac{lh[k]}{\sum_{k=1}^{K_i} lh[k]}$;
13 $sum[k] \leftarrow sum[k] + lh[k]$;
14 **for** $k = 1 \ldots GMM_i.K_i$ **do**
15 $GMM_i^{(x)}.\hat{m}_{i,k}^{(x)} \leftarrow \frac{sum[k]}{|\overline{C}_i^{(x)}|}$;
16 **until** *converged*
17 **for** $x = 2 \ldots M$ **do**
18 $KL_i^{(x)} \leftarrow EvaKL(GMM_i^{(x-1)}, GMM_i^{(x)})$;
19 $\mathbf{f}_i^{(x)} \leftarrow (\hat{m}_{i,1}^{(x)}, \ldots, \hat{m}_{i,K_i}^{(x)}, KL_i^{(x)})$;
20 $F_i \leftarrow \mathbf{f}_i^{(1)} \cup \ldots \cup \mathbf{f}_i^{(M)}$;
21 **return** F_i;

period of interest P, to describe the distribution of costs in the sparse interval. This avoids over-fitting to the cost values in sparse intervals.

For example, the 3-rd interval of the edge shown in Figure 3.8(c) contains only one cost value. Mixing coefficients estimated based on this single value may be over-fitted to this value, which is not optimal. Sparse intervals typically occur during periods with low traffic, for which the value very much depends on the particular driver and therefore can vary considerably.

New mixing coefficients are estimated for the remaining *non-sparse* intervals (lines 4–16). In each iteration, the mixing coefficients are updated based on normalised likelihood values of the mixing coefficients obtained from the previous iteration.

Finally, Kullback-Leibler divergence is evaluated for each interval, and the point for each interval is formulated (lines 17–20). Function $EvaKL(\cdot,\cdot)$ returns a KL divergence value that is normalised to the unit range $[0,1]$, e.g., by dividing by the largest KL divergence value on edge e_i.

Clustering Points: In Section 3.2.1, we assumed that (i) the traffic in the same

hidden state behaves similarly, and (ii) the current state depends on its previous state. Thus, we consider both *travel cost distance* and *temporal dependency distance* when clustering points $\mathbf{f}_i^{(1)}, \ldots, \mathbf{f}_i^{(M)}$.

Dis_C and Dis_T measure the travel-cost distance and the temporal dependency distance between a point $\mathbf{f}_i^{(x)}$ and the center of a cluster U_n, respectively. Recall that the first K_i coordinates of a point $\mathbf{f}_i^{(x)}$ refers to the travel-cost distribution in the x-th interval and that the KL value (i.e., the last coordinate) captures the temporal dependency of the travel cost distributions in the $(x-1)$-st and x-th intervals. Thus, Dis_C computes a distance using the first K_i coordinates of $\mathbf{f}_i^{(x)}$ and of the center of U_n, and it relates to assumption (i); and Dis_T computes a distance using the KL value of $\mathbf{f}_i^{(x)}$ and of the center of U_n, and it relates to assumption (ii). Specifically, we have:

$$\begin{aligned} Dis_C(\mathbf{f}_i^{(x)}, U_n) &= \textstyle\sum_{k=1}^{K_i} m_{i,k} \cdot (\hat{m}_{i,k}^{(x)} - \bar{\hat{m}}_{i,k}^{(n)})^2 \\ Dis_T(\mathbf{f}_i^{(x)}, U_n) &= (KL_i^{(x)} - \bar{KL}_i^{(n)})^2 \end{aligned}$$

Here, $\bar{\hat{m}}_{i,k}^{(n)}$ and $\bar{KL}_i^{(n)}$ are the coordinates of the center of cluster U_n, which is equal to the average value of the k-th coordinates, and the KL value of the points in cluster U_n, respectively.

Based on the above, the distance between a point $\mathbf{f}_i^{(x)}$ and the center of a cluster U_n is defined in Equation 3.5 as a weighted (using parameter λ), linear combination of Dis_C and Dis_T.

$$Dis(\mathbf{f}_i^{(x)}, U_n) = \lambda \cdot Dis_C(\mathbf{f}_i^{(x)}, U_n) + (1-\lambda) \cdot Dis_T(\mathbf{f}_i^{(x)}, U_n) \tag{3.5}$$

The time-cost clustering procedure is described in Algorithm 10. K-Means clustering [30] that uses the distance function defined in Equation 3.5 is applied. The algorithm calls K-Means with $k = 1$ and increments k until a termination criterion is satisfied.

Algorithm 10: TimeCostClustering

Input : CompTimeSeries: $\overline{\mathscr{TS}}_i$; GMM: GMM_i; double λ;
Output: A state set: S_i;

```
1  F_i ← CompactTimeSeriesToPoints(TS_i, GMM_i);
2  int k ← 0;
3  repeat
4      k ← k + 1;
5      ClusterSet U ← K − Means(F_i, k, λ);
6      PreviousDis ← PreviousDis ∪ Dis(U);
7  until TermHeur(PreviousDis);
8  StateSet S_i ← U;
9  return S_i;
```

The algorithm chooses an appropriate number of clusters, each corresponding to a hidden state. In one extreme, if a state is created for each point, the states have the best quality because each state has a unique travel cost distribution and time dependency. However, the parameter space (notably the transition probabilities) of an STHMM increases exponentially with the number of states, thus rendering parameter learning expensive or infeasible. Towards the other extreme, if choosing only one or

a small number of clusters, the STHMM is unlikely to capture adequately the traffic dynamics, thus decreasing the travel cost inference accuracy.

To choose the number k of clusters, a termination heuristic (line 7) is applied. We track the overall distance value when having k clusters $U = U_1, \ldots, U_k$, as defined in Equation 3.6.

$$Dis(U) = \sum_{n=1}^{k} \sum_{\mathbf{f}_i^{(x)} \in U_n} Dis(\mathbf{f}_i^{(x)}, U_n) \tag{3.6}$$

For most edges, as k increases, the overall distances initially drop quickly, but then subsequently drop only slowly. If the distance decrease is low for several consecutive steps, the process terminates, and the k obtained prior to the onset of the low distance decrease is chosen.

Finally, each cluster is regarded as a state, and a state is represented by the coordinates of its center. For example, eight clusters are identified for the edge shown in Figure 3.8(c). The coordinates of the cluster centers are shown in Figure 3.9(d).

Discussion: In time-cost clustering, instead of estimating distinct GMMs with different Gaussian components for the travel costs in different intervals, we use the K_i Gaussian components identified during cost clustering. This has two benefits. First, it provides reliable travel cost distributions for sparse intervals, which reduces the effects of the data sparsity. Second, it allows transformation of the travel cost distribution in each interval into a $K_i + 1$ dimensional point. Clustering of points using weighted Euclidean distance (based on Equation 3.5) is much more efficient than clustering distributions. Measuring the similarity between two distributions accurately and reliably typically requires expensive sampling and the use of many sampling points.

3.3.2 Parameter Learning

Having obtained sets of states for all edges, the parameters that specify an STHMM as defined in Section 3.2.2.3 need to be identified. We determine output, transition, and initial probabilities.

3.3.2.1 Output Probabilities

Since a state $s_i^{(x)} \in S_i$ of edge e_i is a cluster of points containing mixing coefficients of GMMs, the output probability of the state is also defined as a GMM. The output probability $d_i^{(x)}(c)$ of state $s_i^{(x)}$ on edge e_i is defined as follows.

$$d_i^{(x)}(c) = \sum_{k=1}^{K_i} \bar{m}_{i,k}^{(x)} \cdot \mathcal{N}(c|\mu_{i,k}, \delta_{i,k}^2)$$

The Gaussian component $\mathcal{N}(c|\mu_{i,k}, \delta_{i,k}^2)$ is as defined in Equation 3.2. The k-th mixing coefficient is the average of all the k-th coordinates of the points in state $s_i^{(x)}$: $\bar{m}_{i,k}^{(x)} = \sum_{\mathbf{f}_i^{(y)} \in s_i^{(x)}} \hat{m}_{i,k}^{(y)} / |s_i^{(x)}|$.

3.3.2.2 Transition Probabilities

The original, sparse time series are used for learning transition probabilities, as they provide more observations of state transitions than do the compact time series and thus

yield more accurate learning. Given a sparse time series, the possible states of each interval are estimated based on the cost values in the interval. Thus, sparse time series are transformed into uncertain state sequences, based on which transition probabilities are determined using maximum likelihood estimation.

State Estimation: Given a set of travel costs $C_i^{(t)}$ during the t-th interval on edge e_i, the probability that the corresponding state is $s_i^{(x)} \in S_i$, denoted as $\mathbf{P}(q_i^{(t)} = s_i^{(x)} | C_i^{(t)})$, is estimated using Equation 3.7 according to the Bayes' theorem.

$$\mathbf{P}(q_i^{(t)} = s_i^{(x)} | C_i^{(t)}) = \frac{\mathbf{P}(C_i^{(t)} | q_i^{(t)} = s_i^{(x)}) \cdot \mathbf{P}(q_i^{(t)} = s_i^{(x)})}{\sum_{x=1}^{|S_i|} \mathbf{P}(C_i^{(t)} | q_i^{(t)} = s_i^{(x)}) \cdot \mathbf{P}(q_i^{(t)} = s_i^{(x)})} \tag{3.7}$$

Here $\mathbf{P}(C_i^{(t)} | q_i^{(t)} = s_i^{(x)})$ is the probability that costs in $C_i^{(t)}$ are observed in state $s_i^{(x)}$ (i.e., output probabilities), and thus we have:

$$\mathbf{P}(C_i^{(t)} | q_i^{(t)} = s_i^{(x)}) = \prod_{c \in C_i^{(t)}} \mathbf{P}(c | q_i^{(t)} = s_i^{(x)}) = \prod_{c \in C_i^{(t)}} d_i^{(x)}(c)$$

If $C_i^{(t)}$ is empty due to data sparsity, $\mathbf{P}(C_i^{(t)} | q_i^{(t)} = s_i^{(x)})$ is omitted from Equation 3.7, i.e., we set $\mathbf{P}(C_i^{(t)} | q_i^{(t)} = s_i^{(x)})$ to 1.

Further, the prior $\mathbf{P}(q_i^{(t)} = s_i^{(x)})$ is decided based on the temporal context, i.e., the t-th interval. Assume that the t-th interval of edge e_i's sparse time series corresponds to the t'-th interval of its compact time series. If the corresponding point in the t'-th interval $\mathbf{f}_i^{(t')}$ is assigned to a state $s_i^{(\gamma)}$ (i.e., $\mathbf{f}_i^{(t')} \in s_i^{(\gamma)}$) in the state formulation phase then $s_i^{(\gamma)}$ is used as the default state.

Although the default state $s_i^{(\gamma)}$ is treated as the most probable state of e_i during the t-th interval, this does not mean that the remaining states $s_i^{(\gamma')}$ ($\gamma' \neq \gamma$ and $s_i^{(\gamma')} \in S_i$) are not possible at all. Motivated by the notion of additive smoothing [35], we smooth the probabilities among every possible state by assigning a small probability ε to each non-default state and assigning a big probability $1 - \varepsilon \cdot (|S_i| - 1)$ to the default state. Formally, we have:

$$\mathbf{P}(q_i^{(t)} = s_i^{(x)}) = \begin{cases} 1 - \varepsilon \cdot (|S_i| - 1) & s_i^{(x)} = s_i^{(\gamma)} \\ \varepsilon & s_i^{(x)} \neq s_i^{(\gamma)} \end{cases} \tag{3.8}$$

Having estimated the possible states for each interval in a sparse time series $\mathscr{TS}_i$, an uncertain state sequence is formulated. The t-th element in the uncertain state sequence contains the probabilities for each state in S_i, namely $\mathbf{P}(q_i^{(t)} = s_i^{(x)} | C_i^{(t)})$ for $1 \leqslant x \leqslant |S_i|$.

Figure 3.10 shows uncertain state sequences for edges e_1, e_2, and e_5. The costs during the 1-st, 2-nd, and 3-rd intervals are only observed on edges e_1 and e_5, edges e_1 and e_2, and edge e_1, respectively. Thus, possible states on $q_1^{(1)}$, $q_5^{(1)}$, $q_1^{(2)}$, $q_2^{(2)}$, and $q_1^{(3)}$ can be obtained based on the corresponding cost values using Equation 3.7, while all the remaining states are obtained using Equation 3.8 (where $\varepsilon = 0.01$ as an example).

Learning Transition Probabilities: Transition probabilities are learned from the obtained uncertain state sequences using maximum likelihood estimation. The state transition probability of edge e_i, $h_{l,i_1,\ldots,i_k}^{x_i,x_{i_1},\ldots,x_{i_k}}$, is defined as the probability of e_i being in

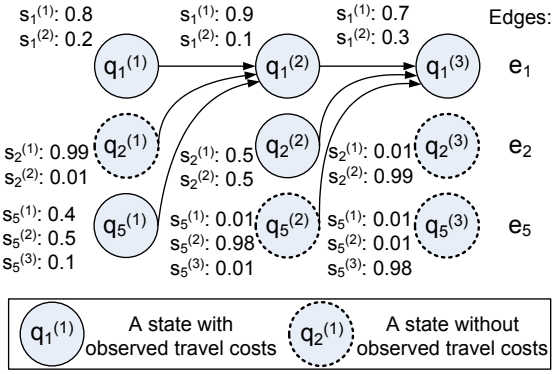


Figure 3.10: Learning Transition Probabilities

state $s_i^{(x_i)}$ given that the previous states of e_i's n-th order neighbors being $s_{i_1}^{x_{i_1}} \dots s_{i_k}^{x_{i_k}}$, respectively. It is estimated using Equation 3.9.

$$h_{i,i_1,\dots,i_k}^{x_i,x_{i_1},\dots,x_{i_k}} = \frac{\sum_{t=2}^{T} \mathbf{P}(q_i^{(t)} = s_i^{(x_i)}) \cdot \prod_{j=1}^{k} \mathbf{P}(q_{i_j}^{(t-1)} = s_{i_j}^{(x_{i_j})})}{\sum_{t=1}^{T-1} \prod_{j=1}^{k} \mathbf{P}(q_{i_j}^{(t)} = s_{i_j}^{(x_{i_j})})} \tag{3.9}$$

An example of determining transition probabilities for edge e_1 is shown in Figure 3.10. Using Equation 3.9, $h_{1,1,2,5}^{(1,1,2,3)}$, i.e., the probability of e_1 being $s_1^{(1)}$ given that the previous states of its neighbors e_1, e_2, and e_5 are $s_1^{(1)}$, $s_2^{(2)}$, and $s_5^{(3)}$, is estimated as $h_{1,1,2,5}^{(1,1,2,3)} = \frac{0.8 \cdot 0.01 \cdot 0.1 \cdot 0.9 + 0.9 \cdot 0.5 \cdot 0.01 \cdot 0.7}{0.8 \cdot 0.01 \cdot 0.1 + 0.9 \cdot 0.5 \cdot 0.01} = 0.730$

3.3.2.3 Initial Probabilities

The initial probability describes the first state of each edge's traffic evolution. As only one uncertain state sequence is identified for each edge, only one training instance is available, which is insufficient to accurately estimate an edge's initial probability.

To derive a meaningful initial probability for each edge, its uncertain state sequence is split based on the period of interest P that was used to obtain the compact time series. Thus, a group of uncertain state sequences is obtained for each edge, which provides more training instances for the estimation. Figure 3.11 shows an example, where P is a day. The full uncertain state sequence is split into short uncertain state sequences, each corresponding to a day.

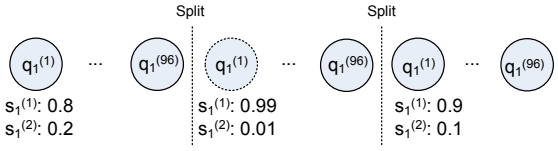


Figure 3.11: Learning Initial Probabilities

Since initial probabilities are independent among different edges, an edge's initial probability can be learnt by only considering the first states in the short uncer-

tain state sequences of the edge. Given edge e_i, assume that the first states in all the short uncertain state sequences are in set SS_i. Then the initial probability of edge e_i is defined by Equation 3.10. For example, using the short uncertain states sequences of edge e_1 shown in Figure 3.11, the probability of the initial state being $s_1^{(1)}$ is: $\tau_1^{(1)} = (0.8 + 0.99 + 0.9)/3 = 0.897$.

$$\tau_i^{(x)} = \frac{\sum_{q_i^{(1)} \in SS_i} \mathbf{P}(q_i^{(1)} = s_i^{(x)})}{|SS_i|} \tag{3.10}$$

3.3.3 Travel Cost Inference

An STHMM with learned parameters enables travel cost inference. Since the inference is similar to the inference on classical HMMs [34], we omit the details and only briefly discuss two cases.

The first concerns the beginning of travel cost inference, where no real-time GPS data is available. In this case, initial probabilities are applied to infer the next states.

The second case occcurs when GPS data streams into the system. Here, the current state is estimated using Equation 3.7 based on the travel costs from the latest interval. Based on the learned transition probabilities, the next states are inferred. Using the inferred next states and the learned output probabilities, the distributions of travel costs in the next intervals are also available. For example, the expected values of the estimated distribution can be used to assign near-future edge weights to road network G.

3.4 Empirical Study

We report on a study that aims to elicit design properties of the proposed framework and algorithms.

3.4.1 Experimental Setup

Data Set: We use 181 million GPS records collected at 1 Hz (i.e., one GPS record per second) in North Jutland, Denmark during week days from April 2007 to March 2008. The data is from an experiment where young drivers start out with a rebate on their car insurance and then are warned if they speed and are penalised financially if they continue to speed.

The GPS data is map matched [5] to OpenStreetMap's road network for Denmark[1], where 34%, 29%, 15%, 9%, and 13% of the data occurs on tertiary, secondary, residential, motorway, and other roads[2], respectively. Most of the data is from urban and suburban regions. The data set is divided into a training set (for learning an STHMM), and a testing set (for testing the accuracy of inferred travel costs). By default, we use the first half year (April to September) of data for training, and the remaining half year for testing.

Since the data only covers part of Denmark and some covered edges have little data, we consider the subset of the road network that is composed of edges that have at least 500 cost records, denoted as E. If an edge has less than 500 records in a year (i.e.,

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

[2] According to OpenStreetMap road categories: `http://wiki.openstreetmap.org/wiki/Highway_tag_usage`.

less than 2 records per day), the edge is either unlikely to have much traffic and traffic variation, or the vehicles in the GPS data set do not cover the edge. Such edges are not of interest to us. The resulting, smaller road network contains $|E| = 1,916$ edges. Thus, the proposed STHMM learns $1,916$ correlated time series. We also note that our study with $1,916$ correlated time series exceeds the sizes of existing studies by two orders of magnitude [36, 37, 38, 39]. In fact, existing studies consider at most 10 correlated time series.

Period of Interest P: It is common to focus on "hot" periods when studying traffic behaviour [37] because such periods are most interesting. In contrast, travel-time estimation during midnight is of relatively little interest. Here, we consider $P = [6{:}00, 20{:}00)$, since the GPS data is collected primarily in P. Although we consider a "hot" period and "hot" edges, the obtained time series remain quite sparse. Table 3.1 reports the percentage of intervals that do not have any costs in the time series obtained using different α.

α (minutes)	15	30	60
Sparsity	89.2%	79.1%	60.2%

Table 3.1: Sparsity of Traffic Time Series

Travel Costs: We consider travel time (TT) and GHG emissions (GE). Travel times are obtained as the difference between the corresponding time points of the last and first GPS records on an edge. We use the VT-micro model [40] to estimate the GHG emissions based on instantaneous velocities and accelerations, which are derived from the available GPS records. A recent benchmark [7] indicates that VT-micro is appropriate for this purpose.

Parameters: We vary the parameters α, λ, and n, which are used in the state formulation and parameter learning steps, according to Table 5.1, where default values are shown in bold. Default values are used unless stated otherwise. Parameter f used in cost clustering (in Algorithm 8) is set to 10, i.e., using 10-fold cross validation. Threshold *mCounts*, used in Algorithm 9, is set to 5. Smoothing parameter ε, used in Section 3.3.2.2, is set to 0.02.

Parameters	Values
α	**15**, 30, 60 (minutes)
λ	0, 0.1, 0.2 , **0.3 (for *TT*)**, 0.4, **0.5 (for *GE*)**, 0.6, 0.7, 0.8, 0.9, 1
n	0, **1**, 2, 3, 4, 5, 6, 7, 8, 9, 10

Table 3.2: Parameter Settings

Accuracy Measurements: When we infer travel costs for an edge, the intervals that have been traversed at least once are of interest to us because they have ground-truth travel costs, which enables us to measure inference accuracy. We call these *test intervals*, and we consider only these in the experiments.

We quantify the effectiveness of the estimated travel costs using average sum of squared loss (*ASSL*). The *ASSL* value of edge e_i is defined as $ASSL(e_i) = \frac{1}{M_i} \cdot \sum_{j=1}^{M_i} (EST_j - GT_j)^2$, where M_i is the total number of test intervals of edge e_i and EST_j and GT_j are the estimated and the ground truth costs for the j-th test interval, respectively. In the

j-th test interval, the STHMM estimates a state that describes the cost distribution during the interval, so we use the expected cost as EST_j. We compare the STHMM with a baseline method, where the EST_j is the average of the cost values observed during the same interval from the training set. The average of all cost values observed in the j-th test interval is used as the ground truth GT_j. The overall accuracy is the average *ASSL* value of every edge, where $ASSL = \frac{1}{|E|} \cdot \sum_{e_i \in E} ASSL(e_i)$.

Implementation Detail: All algorithms are implemented in Java using JDK 1.7. To ease the management of Gaussian mixture models, the jEMF package[3] is applied. Some clustering algorithms are implemented based on Weka[4]. A computer with Windows 7 Enterprise, a 3.40GHz Intel Core i7-2600 CPU, and 16 GB main memory is used for all experiments.

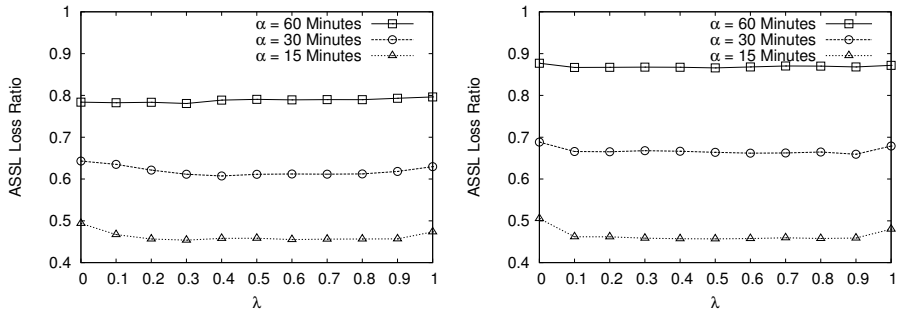
3.4.2 Effects of α and λ

To observe the effects of parameters α and λ, we plot the *ASSL loss ratio* (i.e., $\frac{ASSL_{STHMM}}{ASSL_{Baseline}}$) while varying α and λ in Figure 3.12. Recall that λ controls the relative importance of travel cost distance and temporal dependency distance in time-cost clustering.

Given a fixed λ, the STHMM is always superior to the baseline method for both *TT* and *GE*. For example, for *TT* and $\alpha = 15$, the best ratio is around 45%, meaning that the *ASSL* of the STHMM is 45% of the baseline's *ASSL*. This clearly demonstrates the effectiveness of our STHMM approach.

The STHMMs with finer-grained intervals (i.e., smaller α) produce better *ASSL* loss ratios. It is expected that the traffic between two consecutive 15-minute intervals is more inter-dependent than between two consecutive 60-minute intervals. Longer intervals lower the traffic dependencies between consecutive intervals, yielding a worse *ASSL* loss ratio.

We fix α and study the effect of varying λ. If we only consider cost similarity ($\lambda = 1$) or only consider temporal similarity ($\lambda = 0$), the *ASSL* loss ratios on both *TT* and *GE* are higher than when both similarities are considered, as seen in Figure 3.12. We also see that the loss ratio is not very sensitive to λ, which makes it easy to choose an appropriate λ value. We use $\lambda = 0.3$ and $\lambda = 0.5$ as defaults for subsequent experiments on *TT* and *GE*, respectively, as these values produce the best results.

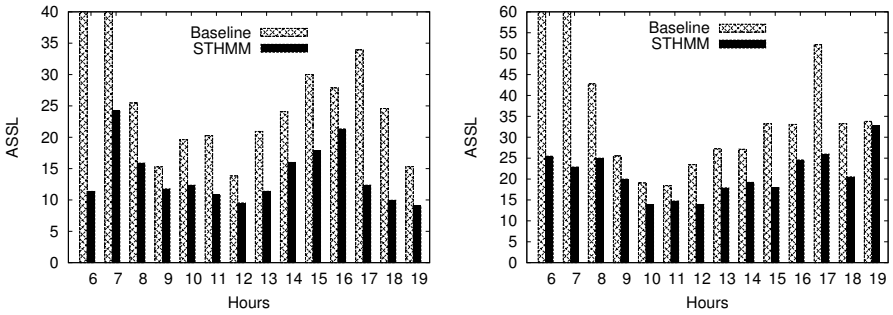


(a) Travel Time (b) GHG Emissions

Figure 3.12: Effects of α and λ

[3] http://www.lix.polytechnique.fr/~nielsen/MEF/
[4] http://www.cs.waikato.ac.nz/ml/weka/

Next, to observe the effectiveness of the STHMM across time, we plot the *ASSL* values for the baseline and the STHMM with α=15 at an hourly granularity in Figure 3.13. For both *TT* and *GE*, the STHMM always has a lower *ASSL* for each hour. The results suggest that the STHMM models traffic evolution during finer-grained intervals much more effectively than the baseline method and is able to provide much more accurate dynamic edge weights.



(a) Travel Time (b) GHG Emissions

Figure 3.13: Hourly Inference, $\alpha = 15$

The total run time of learning an STHMM and the average run time of performing *TT* inference per test interval on the learned STHMM are shown in Figure 3.14 (results on *GE* are similar and are thus omitted). Recall that both state formulation and parameter learning are conducted off-line and are not time critical. The figure shows that as α increases, the total run time of both phases decreases. Given a training period, a smaller α yields more intervals, thus creating more data to be considered in both phases. For $\alpha = 15$, more intervals with varying travel-cost distributions need to be handled during time-cost clustering, yielding state sets with higher cardinalities. Higher cardinalities increase the parameter spaces of the transition probabilities, which makes parameter learning take longer than when $\alpha = 30$ or 60. However, the longest total run time of both phases is below 92 minutes, which is acceptable for an off-line computation.

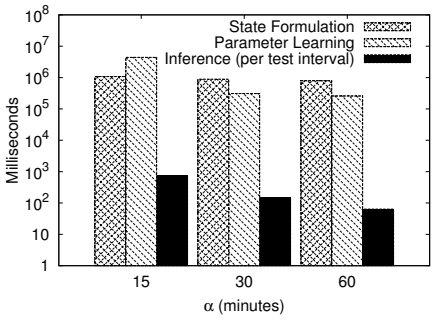


Figure 3.14: Efficiency, *TT*

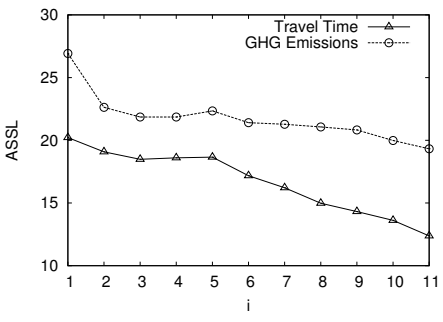


Figure 3.15: Training Size

It is also important to note that both state formulation and parameter learning are parallelizable. State formulation for an edge only uses the compact time series from the edge, and parameter learning for an edge only employs a group of uncertain state sequences from the edge's n-th order neighbors. By distributing the compact time

series and pertinent groups of uncertain state sequences to different computer nodes, both state formulation and parameter learning for different edges can be conducted in parallel. Frameworks such as MapReduce are capable of distributing pertinent data to different nodes and thus enable parallel state formulation and parameter learning. This way, the total run time can be further reduced using known techniques.

Travel cost inference is conducted on-line, meaning that as real-time GPS data streams in, near-future travel costs are to be inferred with little delay. Thus, travel cost inference is time critical. The "Inference (per test interval)" columns in Figure 3.14 suggest that travel cost inference is quite efficient and is capable of supporting real-time inference.

3.4.3 Effects of Training Data

To observe **the effect of the sizes of training sets**, we use data from the first i months for training, and the remaining data for testing, where $1 \leqslant i \leqslant 11$. For example, when $i = 3$, data from April–June 2007 is used for training. The *ASSL* values using different training sets are reported in Figure 3.15. As more data is used for training, the inference accuracy for both *TT* and *GE* also increases (i.e., lower *ASSL* values).

Note that the STHMM always outperforms the baseline, especially when the training set is small. In particular, when $i = 1$, the *ASSL* values of the STHMM is only 19% and 27% of those of the baseline for *TT* and *GE*, respectively, indicating that the STHMM works well when the training set is small and sparse.

It is of interest to **incorporate recent data** and learn an up-to-date STHMM periodically. Assume that we learn a new STHMM every month. We consider two strategies. When predicting travel costs for a new month (e.g., September), *Strategy 1* considers the data collected from all previous months (e.g., April–August) to learn a new STHMM, while *Strategy 2* only considers the data collected from the two most recent months (e.g., July–August). We compare these with a static approach (that always uses an STHMM learned from data collected from April–May).

The results on *TT* are shown in Figure 3.16. The periodically learned STHMMs outperform the static one, especially when the test months are further away from the training months (e.g., from October 2007 to March 2008). The two strategies are almost equally effective (i.e., similar *ASSL* values), but have quite different efficiencies. As time passes, *Strategy 1* uses more and more training data and thus takes longer and longer time (up to around 2 hours). *Strategy 2* always uses only two months of data for training and takes from 25–35 minutes, depending on the numbers of GPS records in the different months. Thus, *Strategy 2* is preferable.

As summer and winter may have different traffic conditions, it is of interest to study the **effect of the seasonality**. We learn a summer STHMM with data from June and July and a winter STHMM with data from December and January. The results on *TT* shown in Figure 3.17 indicate that the summer STHMM achieves better accuracy for summer months (e.g., May and August), while the winter STHMM is best for winter months (e.g., November, February, and March). For the spring and fall months (e.g., April, September, October), both STHMMs perform similarly. The findings are consistent with domain knowledge of summer/winter traffic in Denmark. We also include *Strategy 2* in Figure 3.17: it offers a reasonable all-year accuracy.

We suggest that (i) if a region has different summer and winter traffic, using separate summer and winter STHMMs yield the best results; otherwise, (ii) using *Strategy 2* is also effective. The effects of incorporating recent data and the seasonality on *GE* are similar, and thus are omitted.

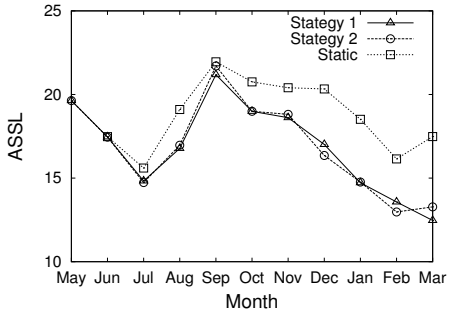


Figure 3.16: Recent Data, *TT*

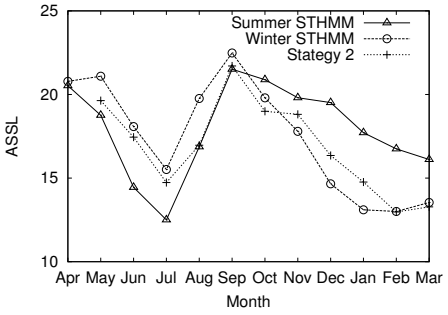


Figure 3.17: Seasonality, *TT*

3.4.4 Effects of n-th Order Coupling

We proceed to consider the effects of different coupling levels, i.e., using different n-th order STHMMs. When $n = 0$, the STHMM degrades to the naive model that is composed of $|E|$ independent HMMs (refer to Section 3.2.2.2). The naive model can be regarded as an improved version of the state-of-the-art approach [41] that applies a Markov model to predict future travel times. It formulates states using travel time clusters. In contrast, in our naive model, a state is a cluster represented by a Gaussian mixture model describing a travel cost distribution, and a KL divergence value representing its temporal dependency w.r.t. its previous state. Also, the state-of-the-art approach is only tested on travel time, not on GHG emissions.

We choose a subset of edges $E' \subset E$ with state set cardinalities that exceed 1 to observe the effect of varying n. The reason for using E' is twofold: if an edge's state set cardinality is 1, (i) the edge stays in its single state, regardless of the states of its n-th neighbors, and (ii) the edge also cannot effect its neighbor edges' traffic because it is always in the single state. Thus, if the cardinality is 1, the n-th ($n \geqslant 1$) order STHMM should yield the same results as the naive model.

The chosen E' contains 653 edges for *TT* and 723 edges for *GE*. As the distributions of *TT* and *GE* can be quite different (cf. the 49-th interval in Figure 3.3) even when both are derived from the same GPS data, the state set cardinalities on the same edge are also quite different for *TT* and *GE*. Thus, the E' contains different edges for *TT* and *GE*.

We plot the average and maximal cardinalities of n-th order neighbors derived from E' in Figure 3.18. When n is large (> 4), some edges have many neighbor edges, causing an exponential increase in the parameter spaces of transition probabilities in the STHMM. For instance, when $n = 5$, an edge may have 28 neighbor edges. If each edge has 4 states, the transition probability of the edge needs to maintain $4 \cdot 4^{28}$ entries, which causes prohibitively high computation and storage overheads. Fortunately, an approximation algorithm (AA) [38, 36] is able to reduce the parameter space from exponential to linear. We use AA to learn the transition probabilities in the STHMM for n up to 10. The details of AA are omitted for brevity.

When $n \leqslant 4$, we use the proposed learning algorithms to study the effectiveness of the n-th order STHMM based on 50 randomly chosen edges from E'. We do not include all edges in E' because parameter learning on some edges is quite time consuming when $n = 3$ or 4. The *couple ratio* $= \frac{ASSL_{n\text{-th order STHMM}}}{ASSL_{Naive}}$ is shown in Figure 3.19.

The n-th ($n \geqslant 1$) order STHMM outperforms the naive model. In particular, the benefit of using the 1-st order STHMM is most significant for inferring both *TT* and

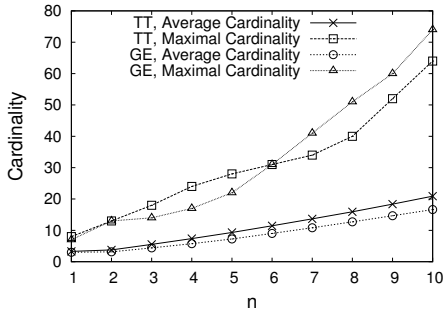


Figure 3.18: Cardinality

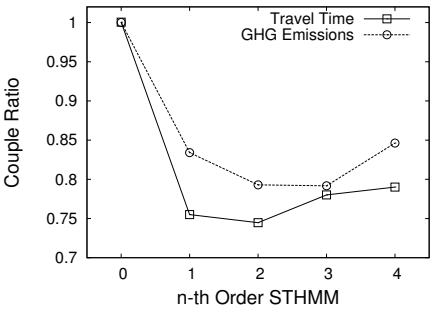


Figure 3.19: Couple Ratio

GE (see the two steep slopes from $n = 0$ to $n = 1$). The 2-nd and 3-rd order STHMMs yield the best results for *TT* and *GE*, respectively. When $n > 1$, the benefit of using a higher-order STHMM is not prominent, and for $n = 4$ the couple ratios even increase. The results suggest that the traffic interactions among different edges are relatively localised and that a fully coupled model [38, 39, 36, 37] is not well suited for road network traffic modeling.

The average run time of parameter learning per edge and the average run time of inference per test interval are shown in Figure 3.20. Figure 3.14 gives the run time of state formulation, which does not change when varying n. As n increases, the parameter space of the STHMM also increases, thus increasing the average run time of parameter learning. The 1-st order STHMM and the naive model have similar run times, and the n-th order STHMM is expensive to learn when $n > 1$. Considering the effectiveness shown in Figure 3.19, the 1-st order STHMM stands out as offering a high effectiveness gain while limiting the additional run time cost.

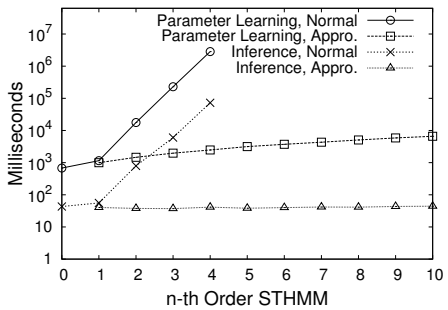


Figure 3.20: Efficiency, TT

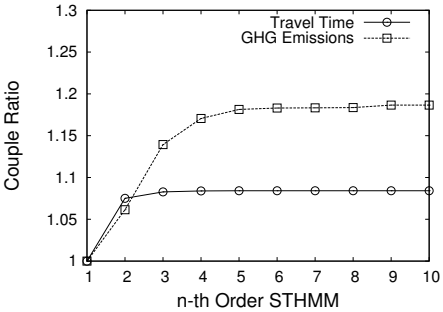


Figure 3.21: Appr. Ratio

Next, we apply AA to all edges in E' to study the effectiveness and efficiency of n-th order STHMMs for $1 \leqslant n \leqslant 10$. The *Appr Ratio*$= \frac{ASSL_{n\text{-th order STHMM AA}}}{ASSL_{1\text{-st order STHMM AA}}}$ is shown in Figure 3.21, and the average run time of AA is also shown in Figure 3.20. AA is much more efficient, but also less effective. Higher order ($n > 5$) STHMMs almost keep the same effectiveness, suggesting that traffic on further-away edges seldom affect the traffic of the edge of interest.

We recommend the use of a lower order ($n < 3$, especially $n = 1$) STHMMs for travel cost inference in road networks. These provide significant benefits at low training

times.

3.5 Related Work

Time Series Analysis: Hidden Markov Models (HMMs) [30, 34] and coupled HMMs [38, 39, 36, 37] are proposed to model individual time series and to model the interactions among multiple time series, respectively. The proposed STHMM has several unique characteristics. First, while the existing approaches consider regular, non-sparse time series (i.e., one value per interval), the heterogenous, sparse time series that we consider may have multiple or no value(s) in an interval.

Second, HMMs and coupled HMMs assume the state set of an individual time series is given a priori, e.g., using domain knowledge. This assumption does not hold in our setting where identifying a distinct state set for each traffic time series is an important challenge. A recent approach, pHMM [29], segments a regular time series into line segments and clusters these line segments to obtain states. Although pHMM is capable of computing state sets for regular time series without relying on prior knowledge, it is inapplicable in our setting because linear transformation of regular time series does not apply to our sparse and heterogenous time series. Rather, the STHMM identifies a distinguishable state set as a set of Gaussian mixture models for each sparse time series.

Third, parameter learning for the STHMM differs from classical HMM parameter learning, e.g., using the Baum-Welch algorithm [34], which needs to estimate output probabilities while estimating initial and transition probabilities. In contrast, the STHMM identifies the output probabilities in the state formulation phase, which simplifies the estimation for initial and transition probabilities in the parameter learning phase. Further, coupled HMMs couple each time series with all the other time series and thus have huge parameter spaces. Existing studies [38, 36] focus on proposing approximate models along with randomised learning algorithms to avoid inaccurate or inefficient inferencing as much as possible. Our STHMM considers the structure of the underlying road network to couple only those time series that need coupling, thus reducing the parameter space substantially.

Travel Cost Inference: Most existing work on travel cost estimation focuses on travel time estimation, and only few studies consider GHG emissions. Further, most estimation approaches are quite static in nature. Although a recent approach [**?**] learns time-dependent travel times and GHG emissions based weights for roads (even for the roads without any GPS data), the weights remain static and do not consider real-time traffic data. The current state-of-the-art proposal for travel time inference, which is also the most related to ours, uses a Markov model to update the travel time on a road based on real-time traffic data from the road. However, the real-time support considers each road segment in isolation, does not exploit the correlation of travel times among road segments, and does not contend with sparsity and heterogeneity. Section 3.4.4 includes an empirical comparison with an improved version of this approach. One study [37] uses coupled HMMs to model dynamic travel times using loop detector data, which assumes that the state set of each time series is given in advance. Further, since loop detectors can constantly report travel time, sparsity is not considered.

3.6 Conclusion and Outlook

We study real-time travel cost inferencing from multiple correlated, sparse time series based on GPS records from probe vehicles. A spatio-temporal hidden Markov model is formalised to model multiple correlated traffic time series, while considering sparsity, dependency, and heterogeneity in an integrated manner. An empirical study considering travel time and GHG emissions demonstrates that the framework and algorithms are effective and efficient.

As the volumes of GPS data increase, it becomes possible to study even larger numbers of correlated traffic time series. The resulting STHMM parameter space increase renders the learning more difficult. First, exact learning may not be feasible, calling instead for approximate sampling based learning. Second, scalable learning algorithms are of particular interest when we face large numbers of correlated time series.

Chapter 4

Using Incomplete Information for Complete Weight Annotation of Road Networks

This work has been accepted by IEEE Transaction of Knowledge and Data Engineering [3] – one of the best journals in the field of data and knowledge management.

Reduction in greenhouse gas (GHG) emissions is crucial in combating global climate change. For example, the EU has committed to reduce GHG emissions to 20% below 1990 levels by 2020 [42]. To achieve these reductions, the transportation sector needs to achieve reductions. For example, in the EU, emissions from transportation account for nearly a quarter of the total GHG emissions [43], making transportation the second largest GHG emitting sector, trailing only the energy sector.

While improved vehicle and engine design are likely to yield GHG emission reductions, eco-routing is readily deployable and is a simple yet effective approach to reducing GHG emissions from road transportation [44]. Specifically, eco-routing can effectively reduce fuel usage and CO_2 emissions. Studies suggest that by providing eco-routes to drivers, approximately 8–20% in fuel savings and lower CO_2 emissions are possible in different settings, e.g., during peak versus off-peak hours, on highways versus areal roads, for light versus heavy duty vehicles [45, 46]. For example, an interesting municipal solid waste collection scenario, where a truck collects solid waste from several locations on Santiago Island, demonstrates a 12% fuel reduction due to eco-routes [47].

Vehicle routing relies on a weighted-graph representation of the underlying road network. To achieve effective eco-routing, it is essential that accurate edge weights that capture environmental costs, e.g., fuel consumption or GHG emissions, associated with traversing the edges are available. Given a graph with appropriate weights, eco-routes can be efficiently computed by existing routing algorithms, e.g., based on Dijkstra's algorithm or the A^* algorithm. However, accurate weights that capture environmental impact are not always readily available for a road network. This paper addresses the task of obtaining such weights for a road network from a collection of measured (*trip*, *cost*) pairs, where the *cost* can be any cost associated with a trip, e.g., GHG emissions, fuel consumption, or travel time.

Because the trips given in the input collection of pairs generally do not cover all edges of the road network and also do not cover all times of the day, data sparsity is

a key problem. The cost of a trip, e.g., GHG emissions, differs during peak versus off-peak hours. Thus, it is inappropriate to use costs associated with peak-hour trips for obtaining edge weights to be used for eco-routing during off-peak hours.

Considering the road network and trips shown in Fig. 4.1, assume that the GHG emissions of trip 1 (traversed from 7:30 to 7:33) and trip 2 (traversed from 23:15 to 23:17) are also given, and assume that we are interested in assigning GHG emission weights to all edges in the network. The assignment of these weights to a large number

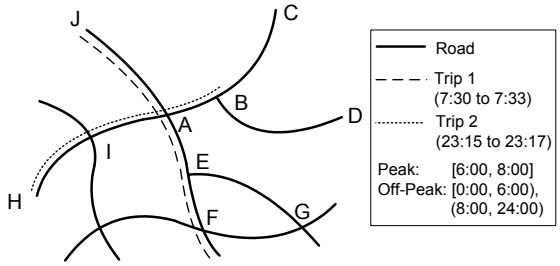


Figure 4.1: Trips on A Road Network

of edges, e.g., BC, BD, EG, and FG, cannot be done directly since they are not covered by any trip. However, for example, BD can be annotated by considering its neighbor road segment AB which is covered by trip 2.

Assuming that the period from 6:00 to 8:00 is the sole peak-hour period (the remaining times being off-peak), trip 1 is not useful for assigning an off-peak weight to the edge AE because trip 1 traversed AE during peak hours. By taking into account the off-peak weights of IA and AB (covered by trip 2), it is, however, possible to obtain an off-peak weight for AE.

This paper proposes general techniques that take as input (i) a collection of ($trip$, $cost$) pairs, where $trip$ captures the edges used and the times when the edges are traversed and the $cost$ represents the cost of the entire trip; and (ii) an unweighted graph model of the road network in which the trips occurred. The techniques then assign travel cost based weights to all edges in the graph.

To the best of our knowledge, this paper is the first to study complete weight annotation of road networks using incomplete information. And this work is accepted as a journal paper on IEEE Transactions on Knowledge and Data Engineering, 2013. In particular, the paper makes four contributions. First, a novel problem, road network weight annotation, is proposed and formalised. Second, a general framework for assigning time-varying trip cost based weights to the edges of the road network is presented, along with supportive models, including a directed, weighted graph model capable of capturing time-varying edge weights and a trip cost model based on time varying edge weights. Third, two novel and judiciously designed objective functions are proposed to contend with the data sparsity. A weighted PageRank-based objective function aims to measure the variance of weights on road segments with similar traffic flows, and a second objective function aims to measure the weight difference on road segments that are directionally adjacent. Fourth, comprehensive empirical evaluations with real data sets are conducted to elicit pertinent design properties of the proposed framework.

The remainder of this paper is organised as follows. Following a survey of related work in Section 4.1, Section 4.2 covers problem definition and a general framework for solving the problem. Section 4.3 details the objective functions. Section 4.4 reports the empirical evaluation, and Section 4.5 concludes and discusses research directions.

4.1 Related Work

Little work has been done on weight annotation of road neworks. Trip cost estimation is a core component of our weight annotation solution. Given a set of $(trip, cost)$ pairs as input, trip cost estimation aims to estimate the costs for trips that do not exist in the given input set. Weight annotation can be regarded as a generalised version of trip cost estimation, since if pertinent weights can be assigned to a road network, the cost of any trip on the road network can be estimated. For example, if a GHG emissions based weighted graph is available, the GHG emissions of a certain trip can be estimated as the sum of the weights of the road segments that the trip traverses.

Most existing work on trip cost estimation [48, 49, 50, 51] focuses on travel-time estimation. In other words, their work focuses on travel time as the trip cost. In general, the methods for estimating the travel times of trips can be classified into two categories: (i) segment models and (ii) trip models.

Segment models [50, 51, 52, 53] concern travel time estimation for individual road segments. For example, observers (e.g., Bluetooth sensors or loop detectors deployed along road segments) monitor the traffic on road segments, recording the flows of vehicles along the road segments. Thus, travel-time estimation tends to concern particular road segments. For example, some studies model travel time on a particular road segment as a time series and apply autoregressive models [50] to estimate the travel time on the road segment. T-Drive [51] models time-dependent travel time distributions on road segments using sets of histograms and enables the inference of future travel times using Markov chains [41]. One study incorporates Lagrangian measurements [53] into existing traffic flow models for motorways to estimate travel time distributions on specific motorways.

Segment models assume "hot" road segments where, preferably, substantial data is available. However, far from every road segment may have enough historical data in practical settings, e.g., due to the limited deployment of costly sensors. Segment models are not well suited for the weight annotation problem because the given $(trip, cost)$ pairs typically fail to cover the whole network, meaning that many road segments lack the data needed to apply such models.

The trip models focus on estimating the costs of individual trips. Specifically, the costs of trips are considered more interesting than the costs of individual road segments. Given a collection of trips and their corresponding travel times, one study [49] proposes a Gaussian process regression based method to predict the travel times for unseen trips. However, the study has the limitation that all the trips are required to share the same source and target. This limitation renders the study of limited interest to us, since we aim at annotating every edge with a pertinent weight. Trajectory regression [48] was proposed recently to infer the travel times of arbitrary trips. The method is able to estimate the travel times of trips consisting of road segments with no or little traversal history by considering the travel time correlation of spatially adjacent road segments.

Trajectory regression is the most related method to our weight annotation problem. However, our study distinguishes itself with several unique characteristics. First, we propose a general framework for annotating edges in a road network with a range of trip cost based weights and are not constrained to travel time. Second, we identify the cost correlation of road segments sharing similar traffic flows, and we quantify this by using weighted PageRank values. Third, we consider the temporal cost correlation of adjacent road segments. For example, although two road segments AB and BC are adjacent, the cost of traversing AB during peak hours is not necessarily correlated to the cost of traversing BC during off-peak hours. Fourth, we take into account the directionality

of road segments and consider only *directional adjacency* when determining the cost correlation of spatially adjacent road segments. Last but not least, we conduct comprehensive experiments on real data sets (real trips and real road networks) to demonstrate the effectiveness of annotating road networks with both travel time based weights and GHG emissions based weights. The earlier study on trajectory regression [48] considers only synthetic data and estimates only travel times of trips.

In the intelligent transportation system research field [44, 54, 40], other travel costs (besides travel time) of trips are studied. For example, fuel consumption and GHG emissions of a trip can be computed based on instantaneous vehicle velocities and accelerations, the slopes of the road segments traversed, and the engine type. However, these methods are designed to estimate the costs of individual trips and are not readily applicable to the problem of annotating graph edges with trip cost based weights, notably edges that do not have any traversed trips.

4.2 Preliminaries

We cover the modeling that underlies the proposed framework, and we provide an overview of the framework and its setting.

We use blackboard bold upper case letter for sets, e.g., $\mathbb{E}$, bold lower case letters for vectors, e.g., $\mathbf{d}$, and bold upper case letters for matrices, e.g., $\mathbf{M}$. Unless stated otherwise, the vectors used are column vectors. The i-th element of vector $\mathbf{d}$ is denoted as $\mathbf{d}[i]$, and the element in the i-th row and j-th column of matrix $\mathbf{M}$ is denoted as $\mathbf{M}[i, j]$. Matrix $\mathbf{M^T}$ is $\mathbf{M}$ transposed. An overview of key notation used in the paper is provided in Table 4.1.

Notation	Description
G, G'	The primal graph and the dual graph.
G'_k	The dual graph in traffic category tag tag_k.
$\mathbb{V}$, $\mathbb{E}$	The vertex set and the edge set.
$\mathbb{V}'$, $\mathbb{E}'$	The dual vertex set and the dual edge set.
$\mathbf{d}$	The cost variable vector for all edges.
$PR_k(v'_i)$	The weighted PageRank value of dual vertex v'_i in traffic category tag tag_k.

Table 4.1: Key Notation

4.2.1 Modeling a Temporal Road Network

A road network is modeled as a directed, weighted graph $G = (\mathbb{V}, \mathbb{E}, L, F, H)$, where $\mathbb{V}$ and $\mathbb{E}$ are the vertex and edge sets, respectively; L is a function that records the lengths of edges; F is a function that maps times to traffic categories; and H is a function that assigns time-varying weights to edges. We proceed to cover each component in more detail.

A vertex $v_i \in \mathbb{V}$ represents a road intersection or an end of a road. An edge $e_k \in \mathbb{E} \subseteq \mathbb{V} \times \mathbb{V}$ is defined by a pair of vertices and represents a directed road segment that connects the (intersections represented by) two vertices. For example, edge (v_i, v_j) represents a road segments that enables travel from vertex v_i to vertex v_j. For convenience, we call this graph representation of a road network the *primal graph*.

Fig. 4.2 captures the upper right part of the road network shown in Fig. 4.1 in more detail. Here, *Avenue 1* and *Avenue 2* are bidirectional roads, and *Street 3* is a one-way road that only allows travel from vertex B to vertex D.

The corresponding primal graph is shown in Fig. 4.3. In order to capture the bidirectional *Avenue 1*, two edges (A,B) and (B,A) are generated. Since *Street 3* is a one-way road, only one edge, (B,D), is created.

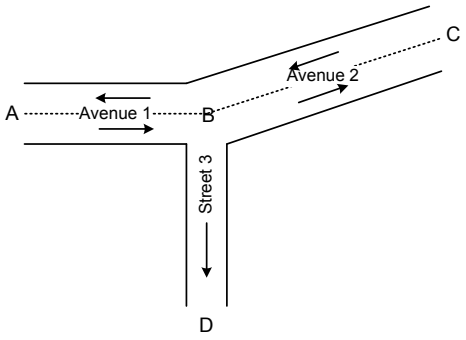


Figure 4.2: Road Network

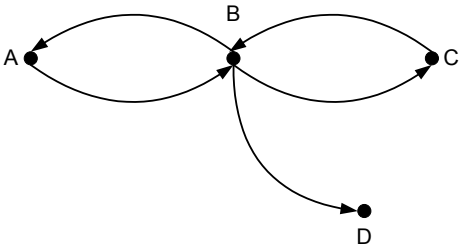


Figure 4.3: Primal Graph

It is essential to model a road network as a directed graph because the cost associated with traveling in two different directions may differ very substantially. For example, traveling uphill is likely to have a higher fuel cost than traveling downhill. As another example, the congestion may also vary greatly for the two directions of a road.

Function $L : \mathbb{E} \rightarrow \mathbb{R}$ takes as input an edge and outputs the length of the road segment that the edge represents. If road segment AB is 135 meters long, we have $G.L((A,B)) = G.L((B,A)) = 135$.

Next, the cost of traversing the same edge may differ across time. This is typically due to varying degrees of congestions. Thus, GHG emissions or fuel consumption are likely to differ during peak versus off-peak times. To this end, function $F : TD \rightarrow TAGS$ models the varying traffic intensity during different periods. Specifically, F partitions time TD and assigns a *traffic category tag* in $TAGS$ to each partition. The granularity of the tags are chosen so that the traffic intensity can be assumed to be constant during the time associated with the same tag. For example, $F([0{:}00,\ 7{:}00)) = OFFPEAK$, $F([7{:}00,\ 9{:}00)) = PEAK$, $F([9{:}00,\ 17{:}00)) = OFFPEAK$, etc.

Finally, function $H : \mathbb{E} \times TAGS \rightarrow \mathbb{R}$ assigns time dependent weights to all edges. In particular, H takes as input an edge and a traffic tag, and outputs the weight for the edge during the traffic tag.

Specifically, $G.H(e_i, tag_j) = d_{(e_i,\ tag_j)} \cdot G.L(e_i)$, where $d_{(e_i,\ tag_j)}$ indicates the cost per unit length of traversing edge e_i during tag tag_j and $G.L(e_i)$ is the length of edge e_i. To maintain the different costs on different edges during different traffic tags, function H maintains $|E| \cdot |TAGS|$ cost variables, denoted as $d_{(e_i,\ tag_j)}$ (where $1 \leqslant i \leqslant |E|$ and $1 \leqslant j \leqslant |TAGS|$).

We organise all the cost variables into a **cost vector d** $\in \mathbb{R}^{(|E| \cdot |TAGS|)}$ and **d**= $[d_{(e_1,\ tag_1)}, \ldots, d_{(e_{|E|},\ tag_1)}, d_{(e_1,\ tag_2)}, \ldots, d_{(e_{|E|},\ tag_2)}, \ldots, d_{(e_1,\ tag_{|TAGS|})}, \ldots, d_{(e_{|E|},\ tag_{|TAGS|})}]^{\mathbf{T}}$. The x-th element of the vector, i.e., $\mathbf{d}[x]$, equals $d_{(e_i,\ tag_j)}$ and $x = pos(i,j) = (j-1) \cdot |TAGS| + i$. Note that if the cost vector **d** becomes available, the function $G.H$ also becomes available.

The proposed model is attractive in our setting. It is simpler than existing models capable of capturing time-varying weights (e.g., time-expanded graphs [55] and time-aggregated graphs [56]), and yet it is sufficiently expressive for the problem we solve.

4.2.2 Trips and Trip Costs

Since vehicle tracking using GPS is widespread and growing, we take into account trips derived from GPS observations. A GPS trajectory $gpsTr = (gps_1, gps_2, \ldots, gps_n)$ is a sequence of GPS observations, where a GPS observation gps_i specifies the location of a vehicle at a particular time point. After map matching and some pre-processing, a GPS trajectory is transformed into a trip $t = (l_1, l_2, \ldots, l_m)$ that consists of a sequence of *link records* l_i of the form:

$$\textit{link record } l_i: \ (e, t_s, t_e),$$

where $e \in \mathbb{E}$ indicates an edge in G and t_s and t_e indicate the time points of the first and last GPS observations on edge e_i.

If a graph G is available that contains relevant edge costs, the cost of a trip $t = (l_1, l_2, \ldots, l_m)$ can be estimated by Equation 4.1.

$$cost(t) = \sum_{l_i \in t} \sum_{tag_j \in TAGS} weight(l_i, tag_j) \cdot G.H(l_i.e, tag_j), \tag{4.1}$$

where

$$weight(l_i, tag_j) = \frac{\sum_{I \in G.F^{-1}(tag_j)} |I \cap [l_i.t_s, l_i.t_e]|}{|[l_i.t_s, l_i.t_e]|}.$$

Here, $G.F^{-1}$ indicates the inverse function of F defined in G, which takes as input a traffic tag and outputs the set of its corresponding time intervals. Next, $|\cdot|$ denotes the length of an interval. For example, given a trip that contains link record $l_i = (e_j, 6:51, 7:05)$ and the traffic tags given in Section 4.2.1, the cost of the trip is $\frac{10}{15} \cdot G.H(e_j, OFFPEAK) + \frac{5}{15} \cdot G.H(e_j, PEAK) = \frac{10}{15} \cdot d_{(e_j, OFFPEAK)} \cdot G.L(e_j) + \frac{5}{15} \cdot d_{(e_j, PEAK)} \cdot G.L(e_j)$.

4.2.3 Framework Overview

Fig. 4.4 gives an overview of the framework for assigning trip cost based weights to a road network. Various types of raw data collected from a road network, such as GPS observations with corresponding CAN bus data and sensor data, are fed into a pre-processing module. While the GPS observations are obligatory, the CAN bus and sensor data are optional.

Pre-processing module: The GPS observations are map matched and transformed into trips as defined in Section 4.2.2. Next, a cost is associated with each trip. If only GPS observations are available, some costs, e.g., travel time, can be associated with trips directly. Other costs, e.g., GHG emissions, can be derived. For example, models are available in the literature that are able to provide an estimate of a trip's GHG emissions and fuel consumption based on the GPS observations of the trip [44]. If CAN bus data and sensor data are also available along with the GPS data, actual and more accurate fuel consumption and GHG emissions can be obtained directly, and thus can be associated with trips.

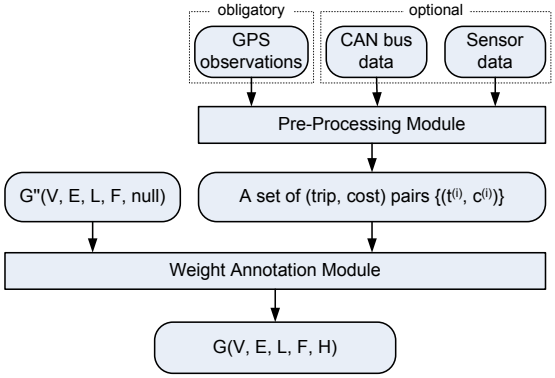


Figure 4.4: Framework Overview

The pre-processing module outputs a set of $(trip, cost)$ pairs $\{(t^{(i)}, c^{(i)})\}$, which then serve as input to the edge annotation module. For example, if the goal is to assign GHG emissions based weights, cost value $c^{(i)}$ indicates the GHG emissions of trip $t^{(i)}$. Note that the cost $c^{(i)}$ is the total cost associated with the i-th trip, meaning that the cost for each individual link record in the i-th trip is not required to be known. This makes it easier to collect $(trip, cost)$ pairs. Because pairs may be obtained in wide variety of ways, the proposed framework has the potential for wide applicability.

Weight annotation module: The $(trip, cost)$ pairs along with a corresponding unweighted graph $G'' = (\mathbb{V}, \mathbb{E}, L, F, null)$ are fed into the weight annotation module. This module assigns pertinent weights to the edges of the graph, and it outputs an weighted graph $G = (\mathbb{V}, \mathbb{E}, L, F, H)$.

Recall that function $G.H$ from Section 4.2.1 is defined by the cost vector $\mathbf{d}$. Given a set of $(trip, cost)$ pairs $\mathbb{TC} = \{(t^{(i)}, c^{(i)})\}$, the core task of this module is to estimate appropriate cost variables in vector $\mathbf{d}$. We formulate the weight annotation problem as a supervised learning problem, namely a regression problem [30] that employs $\mathbb{TC}$ as the training data set to estimate cost variables in vector $\mathbf{d}$.

The regression problem is solved by minimizing a judiciously designed objective function composed of three sub-objective items. The first item measures the misfit between the given actual cost and the estimated cost (i.e., the cost obtained from the cost model described in Equation 4.1) for every trip in $\mathbb{TC}$. The second item measures the differences between the cost variables of two edges whose expected traffic flows (based on topological structures) are similar. The third item measures the differences between the cost variables of two edges which are directionally adjacent. Further, other appropriate metrics that can quantify the difference between the cost variables of two edges can also be incorporated into the module. Finally, minimizing the objective function is handled by solving a system of linear equations.

4.3 Objective Functions

Since we regard the problem as a regression problem, we elaborate on the design of the proposed objective function and the solution to minimizing the objective function.

4.3.1 Residual Sum of Squares

In order to obtain an appropriate estimation of the cost vector $\mathbf{d}$, we need to make sure that for every (*trip*, *cost*) pair $(t^{(i)}, c^{(i)}) \in \mathbb{TC}$, the misfit between the actual cost (e.g., $c^{(i)}$) and the estimated cost (e.g., $cost(t^{(i)})$ evaluated by Equation 4.1, which employs $\mathbf{d}$), is as small as possible. To quantify the misfit, the residual sum of squares (*RSS*) function is applied, where

$$RSS(\mathbf{d}) = \sum_{(t^{(i)},c^{(i)})\in\mathbb{TC}} (c^{(i)} - cost(t^{(i)}))^2.$$

To facilitate the following discussion, we derive a matrix representation of the *RSS* function, as shown in Equation 4.2.

$$RSS(\mathbf{d}) = ||\mathbf{c} - \mathbf{Q}^{\mathbf{T}}\mathbf{d}||_2^2 \tag{4.2}$$

Let the cardinality of the set $\mathbb{TC}$ be N (i.e., $|\mathbb{TC}| = N$). We define a vector $\mathbf{c} \in \mathbb{R}^N = [c^{(1)}, c^{(2)}, \ldots, c^{(N)}]^{\mathbf{T}}$, where $c^{(i)}$ is the given actual cost of the trip $t^{(i)}$, and $(t^{(i)}, c^{(i)}) \in \mathbb{TC}$. A matrix $\mathbf{Q} \in \mathbb{R}^{|\mathbf{d}|\times N} = [\mathbf{q^{(1)}}, \mathbf{q^{(2)}}, \ldots, \mathbf{q^{(N)}}]$ is introduced to enable us to rephrase Equation 4.1 into a matrix representation. Specifically, $\mathbf{q^{(k)}}$ is the k-th column vector in $\mathbf{Q}$ which corresponds to trip $t^{(k)}$. If trip $t^{(k)}$ contains a link record l whose corresponding edge is e_i (i.e., $l.e = e_i$), then $\mathbf{q^{(k)}}[pos(i,j)] = G.L(e_i) \cdot weight(l, tag_j)$ where $1 \leqslant j \leqslant |TAGS|$; otherwise, it is set to 0.

Different from ordinary regression problems, minimizing Equation 4.2 is insufficient for determining every cost variable in $\mathbf{d}$ because the trips in $\mathbb{TC}$ may not cover all the edges in the road network, e.g., all the edges in $\mathbb{E}$. For the edges that are never traversed by any trip in $\mathbb{TC}$, their corresponding cost variables in $\mathbf{d}$ cannot be determined by only minimizing the *RSS* function.

In this case, annotating the edges that do not appear in $\mathbb{TC}$ with weights seems to be difficult and even unsolvable. In the following, we try to use the topology of the road network to further propagate and constrain the cost variables in order to assign an appropriate weight to every edge.

4.3.2 Topological Constraint

The topology of a road network is highly correlated with human movement flow [57, 58], including the movement of both pedestrians and vehicles. Edges with similar movement flows can be expected to have similar cost variables. Thus, if an edge is covered in $\mathbb{TC}$, its cost variable information can be propagated to the edges that have similar movement flows. To this end, we study how to quantify movement flow based similarity between edges using topological information of road networks.

4.3.2.1 Modeling Traffic Flows with PageRank

We transfer the idea of using PageRank for the modeling of web surfers to the modeling of vehicle movement in road networks. The original PageRank employs the hyperlink structure of the web to build a first-order Markov chain, where each web page corresponds to a state [59]. The Markov chain is governed by a transition probability matrix $\mathbf{M}$. If web page i has a hyperlink pointing to web page j then $\mathbf{M}[i,j]$ is set to $\frac{1}{outDegree(i)}$; otherwise, it is set to 0. $\mathbf{M}[i,j]$ indicates the probability of transition from state i to state j. PageRank models a user browsing the web as a Markov process based on matrix $\mathbf{M}$,

and the final PageRank vector is the stationary distribution vector $\mathbf{x}$ of matrix $\mathbf{M}$. The PageRank of web page i, i.e., $\mathbf{x}[i]$, indicates the probability that the user visits page i or, equivalently, the fraction of time the user spends on page i in the long run [59].

The modeling movements of vehicles on a road network as stochastic processes is well studied in the transportation field [60]. In particular, the modeling of vehicle movements as Markov processes is an easy-to-use and effective approach [58]. Thus, we build a first-order Markov chain with a transition probability matrix derived from both the topology of the road network and the trips that occur in the road network. A state corresponds to an edge in the primal graph (i.e., a directed road segment), not a vertex (i.e., a road intersection).

The PageRank value of a state indicates the probability that a vehicle travels on the edge or, equivalently, the fraction of time a vehicle spends on the edge in the long run. Thus, the PageRank value is expected to reflect the traffic flow on the edge. Further, a series of topological metrics [57], including centrality-based metrics, small-world metrics, space-syntax metrics, and PageRank metrics, have been applied to capture human movement flows in urban environments. When using a graph representation of an urban environment, it is found that the classical and weighted PageRank metrics are highly correlated with human movements [57, 61]. Thus, if two edges have similar PageRank values, the traffic flow on the two segments should be similar.

When modeling web surfers, PageRank assumes that the Markov chain is time-homogeneous, meaning that the probability of transferring from page i to page j has the same fixed value at all times. In other words, matrix $\mathbf{M}$ is static across time. In contrast, the time-homogenous assumption does not hold for vehicles traveling in road networks. For example, during peak hours, the transition probability from edge i to edge j may be substantially different from the probability during off-peak hours. Thus, we maintain a distinct transition probability matrix $\mathbf{M}_k$ for each traffic category tag tag_k. During a particular traffic tag, we assume the Markov chain to be time-homogeneous.

4.3.2.2 PageRank on Dual Graphs

PageRank was originally proposed to assign prestige to web pages in a web graph, where web pages are modeled as vertices and the hyper-links between web pages are modeled as edges. Unlike the web graph, we are not interested in the prestige of vertices (i.e., road intersections) in the primal graph representation of a road network; rather, we are interested in the prestige of edges (i.e., directed road segments).

In order to assign PageRank values to edges, the primal graph $G = (\mathbb{V}, \mathbb{E}, L, F, H)$ is transformed into a dual graph $G' = (\mathbb{V}', \mathbb{E}')$, where each vertex in $\mathbb{V}'$ corresponds to an edge in the primal graph, and where each edge in $\mathbb{E}'$, denoted by a pair of vertices in $\mathbb{V}'$, corresponds to a vertex in the primal graph. Since functions L, F, and H are not of interest in this section, we do not keep them in the dual graph.

To avoid ambiguity, we use the terms edge and vertex when referring to primal graphs and use *dual edge* and *dual vertex* when referring to dual graphs. Further, we use the term weight when referring to the weight of an edge in a primal graph, and we use *dual weight* in the context of dual edges in a dual graph.

We define a mapping $D2P : \mathbb{V}' \cup \mathbb{E}' \rightarrow \mathbb{V} \cup \mathbb{E}$ to record the correspondence between the elements in the dual and primal graphs. Fig. 4.5 show the dual graph that corresponds to the primal graph shown in Fig. 4.3. Since the dual vertex AB corresponds to the edge (A,B) in Fig. 4.3, $D2P(AB) = (A,B)$. Similarly, since the dual edge (CB,BA) corresponds to the vertex B in Fig. 4.3, $D2P((CB,BA)) = B$.

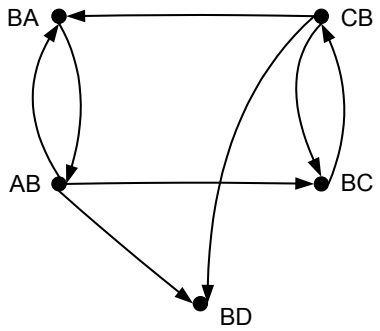


Figure 4.5: Dual Graph

The dual graph is able to model an important characteristic of a road network: at a particular intersection, the probability of which segment a vehicle follows depends on the segment via which the vehicle entered the intersection. Considering the road network shown in Fig 4.2, at intersection (i.e., vertex) B, a vehicle can proceed to follow segments (i.e., edges) (B,A), (B,C), or (B,D). If a vehicle entered the intersection using segment (C,B), it may be unlikely that the vehicle takes a u-turn to follow segment (B,C), while is more likely that it will use the other segments. Similar cases exist if a vehicle arrived at the intersection using segment (A,B).

Modeling this characteristic in a primal graph is not easy. For example, we need to maintain two sets of probabilities on edge (B,C), for the vehicles came from edge (C,B) versus edge (A,B). In contrast, modeling this in a dual graph is straightforward, as how a vehicle entered a particular intersection is clearly represented as a dual vertex. For example, the probabilities on dual edges (CB,BC) and (AB,BC) record the probabilities that a vehicle entered intersection B from edge (C,B) and edge (A,B), respectively, and continues along edge (B,C).

Given the dual graph $G' = (\mathbb{V}', \mathbb{E}')$, original PageRank values are defined formally as follows.

$$PR(v'_i) = \frac{1-df}{|\mathbb{V}'|} + df \cdot \sum_{v'_j \in IN(v'_i)} \frac{PR(v'_j)}{|OUT(v'_j)|}, \quad v'_i \in \mathbb{V}', \tag{4.3}$$

where $PR(v'_i)$ indicates the PageRank value of dual vertex v'_i; $IN(v'_i)$ indicates the set of in-link neighbors of v'_i, i.e., $IN(v'_i) = \{v'_x | (v'_x, v'_i) \in \mathbb{E}'\}$; and $OUT(v'_j)$ indicates the set of out-link neighbors of v'_j, i.e., $OUT(v'_j) = \{v'_x | (v'_j, v'_x) \in \mathbb{E}'\}$. Further, $df \in [0,1]$ is a damping factor, which is normally set to 0.85 for ranking a web graph.

The intuition behind Equation 4.3 is that the PageRank values are composed of two parts: jumping to another random vertex and continuing the random walk. This assumption works fine on the web graph, but we need to adapt this to the different characteristics of the graph representing a road network. In a road network, it is impossible for a vehicle to choose a random edge to traverse when at an intersection. Rather, it can only choose to continue along one of the out-link (dual) edges. Based on this observation, we set the damping factor df to 1. Some existing empirical studies [57] also suggest that with the damping factor set to 1, the resulting PageRank values have the best correlation with the human movement flows.

4.3.2.3 Weighted PageRank Computation

Definition of Dual Weights: In the original PageRank algorithm, a vertex propagates its PageRank value evenly to all its out-link neighbors. In other words, the dual weight for each dual edge from dual vertex v'_j is set uniformly to $\frac{1}{|OUT(v'_j)|}$. The uniform weights on the web graph indicate that a web surfer chooses its next target web page without any preferences to continue its random surfing. However, in a road network, such non-preference surfing usually does not occur. For example, the next step where a vehicle continues often depends on where the vehicle came from, as discussed in Section 4.3.2.2. Also, if *Avenue 1* and *Avenue 2* are the main roads in the road network shown in Fig. 4.2, more vehicles travel from *AB* to *BC* than from *AB* to *BD*. Further, during different traffic category tags, the transitions between dual vertices may also be quite different.

With the availability of very large collections of GPS data, we are able to capture the probability that a vehicle transits from one road segment to another at an intersection during different traffic category tags. Assume we only distinguish between peak and off-peak hours, i.e., there are only two corresponding tags in *TAGS*. Suppose we obtain the number of trips occurred on dual edges, as shown in Table 4.2.

Tags	(AB, BC)	(AB, BD)	(AB, BA)
PEAK	30	10	0
OFFPEAK	5	5	0

Table 4.2: Numbers of Trips Occurred on Dual Edges

For example, among all the trips that occurred on dual vertex *AB* during the peak hours, 30 trips proceeded to follow *BC*, and 10 trips followed *BD*; during off-peak hours, 5 trips followed *BC*, and 5 trips followed *BD*. These observations suggest that the dual weight on dual edge (*AB*, *BC*) should be greater than the dual weight on dual edge (*AB*, *BD*) during peak hours; while they should be the same during off-peak hours.

As the dual graph has different dual weights for different traffic tags, we need to maintain a dual graph for each traffic tag. Specifically, the training data set $\mathbb{TC}$ is partitioned into $\mathbb{TC}_1$, $\mathbb{TC}_2$, ..., $\mathbb{TC}_{|TAGS|}$ according to the traversal times. Partition $\mathbb{TC}_k$ consists only of the trips that are occurred during the time period indicated by the traffic tag tag_k, i.e., $G.F^{-1}(tag_k)$.

The dual weight of a dual edge (v'_i, v'_j) during tag tag_k is related to the ratio of the number of trips that traversed the dual vertices v'_i and v'_j to the number of trips that traversed the dual vertex v'_i, during tag tag_k. Further, to contend with data sparsity, Laplace smoothing is applied to smooth the dual weight values for the dual edges that are not covered by any trip in $\mathbb{TC}$. The dual weight of dual edge (v'_i, v'_j) for the dual graph within tag_k (denoted as G'_k) is computed based on Equation 4.4.

$$W_k(v'_i, v'_j) = \frac{|Trip_k(v'_i, v'_j)| + 1}{\sum_{v'_x \in OUT(v'_i)} |Trip_k(v'_i, v'_x)| + |OUT(v'_i)|}, \tag{4.4}$$

where $Trip_k(v'_i, v'_j)$ returns the set of trips in partition $\mathbb{TC}_k$ that traversed the dual vertices v'_i and v'_j.

Continuing the example shown in Table 4.2, although no trip goes from the dual vertex *AB* directly back to *BA* in $\mathbb{TC}$, this does not mean that such a trip will not occur in the future. Thus, we need to give a small, non-zero value to the dual weight of dual

edge (AB, BA). Using the dual weights provided by Equation 4.4, the dual weights of the out-linking dual edges of dual vertex AB are: $W_{PEAK}(AB, BC) = \frac{31}{43}$, $W_{PEAK}(AB, BD) = \frac{11}{43}$, and $W_{PEAK}(AB, BA) = \frac{1}{43}$; and $W_{OFFPEAK}(AB, BC) = \frac{6}{13}$, $W_{OFFPEAK}(AB, BD) = \frac{6}{13}$, and $W_{OFFPEAK}(AB, BA) = \frac{1}{13}$.

Note that for a given dual vertex v'_i, if no trips in $\mathbb{TC}$ are available to assign the dual weights during a traffic tag tag_k, i.e., $|Trip_k(v'_i, v'_x)| = 0$ for every $v'_x \in OUT(v'_i)$, Equation 4.4 assigns weights with $\frac{1}{|OUT(v'_i)|}$ to each dual edge, which is exactly what the original PageRank algorithm does. For instance, if no trips are available for dual vertex AB (i.e., if the numbers in Table 4.2 are all zeros), the dual weights for $W_k(AB, BC)$, $W_k(AB, BD)$, and $W_k(AB, BA)$ are all $\frac{1}{3}$.

Computing Weighted PageRank Values: Based on the dual weights obtained from Equation 4.4, we construct the transition probability matrices $\mathbf{M_k} \in \mathbb{R}^{|\mathbb{V}'| \times |\mathbb{V}'|}$. Specifically, the ith row and jth column element in $\mathbf{M_k}$, i.e., $\mathbf{M_k}[i, j]$, equals $W_k(v'_i, v'_j)$ if the dual edge (v'_i, v'_j) exists in the dual graph; otherwise, it equals 0. Note that the sum of all elements in a row equals 1, i.e., $\sum_{j=1}^{|\mathbb{V}'|} \mathbf{M_k}[i, j] = 1$ for every $1 \leqslant i \leqslant |\mathbb{V}'|$.

Let vector $\mathbf{v_k} \in \mathbb{R}^{|\mathbb{V}'|}$ record the PageRank values for every dual vertex in G'_k. Specifically, $\mathbf{v_k}[i] = PR_k(v'_i)$, which is the PageRank value of v'_i during traffic category tag tag_k. This way, the PageRank values can be computed iteratively as follows until converged.

$$\mathbf{v_k}^{(n+1)} = \mathbf{M_k}^{\mathbf{T}} \cdot \mathbf{v_k}^{(n)},$$

where $\mathbf{v_k}^{(n)}$ is the PageRank vector in the n-th iteration.

4.3.2.4 PageRank-Based Topological Constraint Objective Function

After obtaining the weighted PageRank values for every dual edge, the topological similarity between two edges in the primal graph is quantified in Equation 4.5.

$$S_k^{PR}(e_i, e_j) = \frac{\min(PR_k(v'_{e_i}), PR_k(v'_{e_j}))}{\max(PR_k(v'_{e_i}), PR_k(v'_{e_j}))} \tag{4.5}$$

The topological similarity between edges e_i and e_j, denoted as $S_k^{PR}(e_i, e_j)$, is defined based on the weighted PageRank values of the two dual vertices representing the edges. To be specific, v'_{e_i} and v'_{e_j} indicate the corresponding dual vertices of edges e_i and e_j, i.e., $D2P(v'_{e_i}) = e_i$ and $D2P(v'_{e_j}) = e_j$. Note that Equation 4.5 returns a high similarity if two edges have similar weighted PageRank scores and that it returns a low similarity, otherwise.

Based on the topological similarity, a PageRank-based Topological Constraint ($PRTC$) function is incorporated into the overall objective function. The intuition behind the $PRTC$ function is that for the same traffic category tag, if two edges have similar traffic flows (as measured by Equation 4.5), their cost variables tend to be similar as well. The $PRTC$ function is defined in Equation 4.6.

$$PRTC(\mathbf{d}) = \sum_{k=1}^{|TAGS|} PRTC(\mathbf{d}, k), \tag{4.6}$$

where

$$PRTC(\mathbf{d}, k) = \sum_{i,j=1}^{|G.\mathbb{E}|} S_k^{PR}(e_i, e_j) \cdot (d_{(e_i, tag_k)} - d_{(e_j, tag_k)})^2.$$

The value of the *PRTC* function over the cost vector $\mathbf{d}$ is the sum of $PRTC(\mathbf{d},k)$ for every $1 \leqslant k \leqslant |TAGS|$. The function $PRTC(\mathbf{d},k)$ computes the weighted (decided by S_k^{PR}) sum of the squared differences of between each pair of road segments' cost variables during traffic tag tag_k.

The *PRTC* function has two important features: (i) if the PageRank values of two edges are similar, the similarity value S_k^{PR} is large, thus making the difference between their cost variables obvious; (ii) if two edges' PageRank values are dissimilar, the similarity value S_k^{PR} with a small value smoothes down the difference between their cost variables. This way, minimizing the *PRTC* function corresponds to minimizing the overall difference between two cost variables whose corresponding road segments have similar traffic flows.

To obtain the matrix representation of the *PRTC* function, we introduce a matrix $\mathbf{A} \in \mathbb{R}^{|\mathbf{d}|\times|\mathbf{d}|}$, which is a block diagonal matrix.

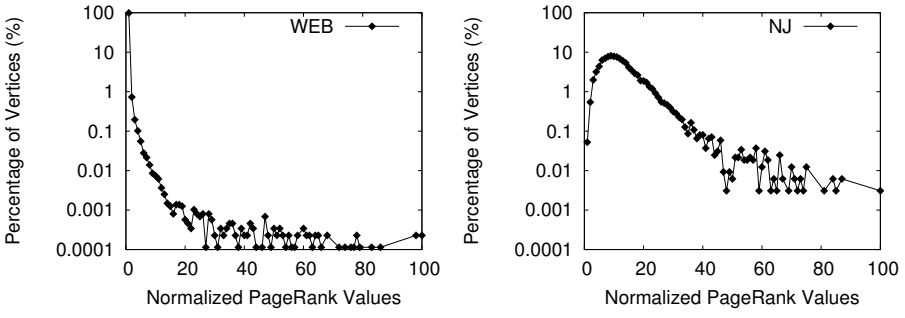
$$\mathbf{A} = \begin{bmatrix} \mathbf{A_1} & & & \\ & \mathbf{A_2} & & \\ & & \cdots & \\ & & & \mathbf{A}_{|TAGS|} \end{bmatrix} \tag{4.7}$$

where $\mathbf{A_k} \in \mathbb{R}^{|E|\times|E|}$ and $\mathbf{A_k}[i,\ j] = S_k^{PR}(e_i,\ e_j)$, which obviously is a symmetric matrix. Let matrix $\mathbf{L_A}$ be the graph Laplacian induced by the similarity matrix $\mathbf{A}$. Specifically, $\mathbf{L_A}[i,\ j] = \delta_{i,j}\cdot\sum_x \mathbf{A}[i,x] - \mathbf{A}[i,j]$, where $\delta_{i,j}$ returns 1 if i equals j, and 0 otherwise. The matrix representation of *PRTC* function is shown in Equation 4.8.

$$PRTC(\mathbf{d}) = \mathbf{d^T L_A d} \tag{4.8}$$

4.3.2.5 Properties of PageRank on Road Networks

Web graphs and road network graphs are quite different, rendering it of interest to study the distributions of PageRank values on the two kinds of graphs. Fig. 4.6 shows the normalised (to (1, 100]) PageRank values with respect to the percentage of vertices having the PageRank values, on a graph (*WEB*) representing a part of the Web[1] and a dual graph (*NJ*) representing the road network of North Jutland, Denmark.



(a) The Web (b) Road Networks

Figure 4.6: PageRank on the Web and a Road Network

Fig. 4.6 suggests that PageRank values on *NJ* are distributed more uniformly than for *WEB*. With this type of distribution, many vertices have the same or very simi-

[1] http://snap.stanford.edu/data/web-Google.html

lar high PageRank values, which renders the distribution ineffective for ranking when compared to *WEB*. However, the distribution is effective for our objective of identifying road segments with similar traffic flows based on PageRank values.

4.3.3 Adjacency Constraint

The *PRTC* function is derived from the overall structure of the road network. In this section, we consider a finer-grained topological aspect of the road network, namely, **directional adjacency**.

An important feature of a road network is that an event at one road segment may propagate to influence adjacent road segments. Consider a typical event in a road network, e.g., traffic congestion. If congestion occurs on road segment (A,B) in Fig. 4.2, road segment (B,C) may also experience congestion, or at least the traffic on (B,C) is affected by the congestion that occurs on (A,B). Thus, the cost variables of two directionally adjacent road segments should be similar.

The directional adjacency we discus here is represented clearly in the dual graph. If and only if two dual vertices are connected by an dual edge in the dual graph, the two corresponding road segments are directionally adjacent. For example, although edges (B, D) and (B, C) (in Fig. 4.3) intersect, their cost variables may not necessarily tend to be similar because no vehicle can travel between these two edges. Directional adjacency is distinct from the "non-directional" adjacency considered in previous work [48].

Another point worth noting is that if two road segments represent opposite directions of the same physical road segment, they are not directionally adjacent. It is natural that an event on a physical road only yields congestion in one direction, but not both directions. Considering the edges (A,B) and (B,A) (in Fig. 4.3), their corresponding vertices in the dual graph (AB and BA in Fig. 4.5) are connected by two edges, however, their cost variables are not necessarily similar.

Directional adjacency is also temporally sensitive. For example, although edges (A,B) and (B,C) are directionally adjacent, the general traffic situation (indicated by the cost variable) on edge (A,B) during peak hours is not necessarily correlated with the traffic on edge (B,C) during non-peak hours.

To incorporate directional adjacency, we incorporate a Directionally Adjacent Temporal Constraint (*DATC*) function into the overall objective function.

$$DATC(\mathbf{d}) = \sum_{k=1}^{k=|TAGS|} DATC(\mathbf{d},k), \tag{4.9}$$

where

$$DATC(\mathbf{d},k) = \sum_{i,j=1}^{|G.\mathbb{E}|} W'_k(v'_{e_i},v'_{e_j}) \cdot (d_{(e_i,tag_k)} - d_{(e_j,tag_k)})^2,$$

and where v'_{e_i} and v'_{e_j} have the same meaning as in Equation 4.5. $W'_k(v'_{e_i},v'_{e_j})$ is as defined in Equation 4.4 if v'_{e_i} and v'_{e_j} do not indicate the same physical road segment; and $W'_k(v'_{e_i},v'_{e_j})$ equals 0 otherwise. For instance, although $W_{PEAK}(AB,BA) = \frac{1}{43}$ as discussed in Section 4.3.2.3, $W'_{PEAK}(AB,BA) = 0$ since AB and BA indicate the same physical road segment, *Avenue* 1.

The *DATC* function aims to make the cost variables satisfy the following property: given road segments e_i and e_j, if a many of the trips that follow e_i also follow e_j, as indicated by $W'_k(v'_{e_i}, v'_{e_j})$, the cost variables on the two edges tend to be more correlated.

Similar to the discussion in Section 4.3.2.4, we introduce a block diagonal matrix $\mathbf{B} \in \mathbb{R}^{|\mathbf{d}|\times|\mathbf{d}|}$ with the same format as matrix $\mathbf{A}$ (defined in Equation 4.7). In particular, in each block matrix, $\mathbf{B_k}[i,j] = \max(W'_k(v'_{e_i}, v'_{e_j}), W'_k(v'_{e_i}, v'_{e_j}))$, which guarantees that matrix $\mathbf{B_k}$, and hence matrix $\mathbf{B}$, are symmetric. Note that it is not possible that both $W'_k(v'_{e_i}, v'_{e_j})$ and $W'_k(v'_{e_j}, v'_{e_i})$ are non-zero because if edge $D2P(v'_{e_i})$ is directionally adjacent to edge $D2P(v'_{e_j})$ then edge $D2P(v'_{e_j})$ cannot be directionally adjacent to edge $D2P(v'_{e_i})$. Let $\mathbf{L_B}$ to be the graph Laplacian derived by matrix $\mathbf{B}$. The *DATC* function is represented by Equation 4.10.

$$DATC(\mathbf{d}) = \mathbf{d^T L_B d} \tag{4.10}$$

4.3.4 Solving The Problem

Combining the three individual objective functions and a classical *L2* regulariser, we obtain the overall objective function $O(\mathbf{d})$:

$$O(\mathbf{d}) = RSS(\mathbf{d}) + \alpha \cdot PRTC(\mathbf{d}) + \beta \cdot DATC(\mathbf{d}) + \gamma \cdot ||\mathbf{d}||_2^2,$$

where α, β, and γ are hyper-parameters that control the tradeoff among the losses on *RSS*, *PRTC*, *DATC*, and the *L2* regulariser. The matrix representation of the objective function is shown in Equation 4.11.

$$O(\mathbf{d}) = ||\mathbf{c} - \mathbf{Q^T d}||_2^2 + \alpha \cdot \mathbf{d^T L_A d} + \beta \cdot \mathbf{d^T L_B d} + \gamma \cdot ||\mathbf{d}||_2^2 \tag{4.11}$$

By differentiating Equation 4.11 w.r.t. vector $\mathbf{d}$ and setting it to 0, we get

$$[\mathbf{QQ^T} + \alpha \cdot \mathbf{L_A} + \beta \cdot \mathbf{L_B} + \gamma \cdot \mathbf{I}]\mathbf{d} = \mathbf{Qc}. \tag{4.12}$$

The solution to Equation 4.12 is the optimal solution to the cost vector, denoted as $\widehat{\mathbf{d}}$, that minimises the overall objective function in Equation 4.11. The linear system in Equation 4.12 can be solved efficiently by several iterative algorithms such as the conjugate gradient algorithm [62].

Finally, feeding the optimised cost variable vector $\widehat{\mathbf{d}}$ to function $G.H$, the time varying weights of the graph become available.

4.3.5 Discussion

In addition to the topology of a road network, other aspects of edges may be useful for identifying similarities among edges, e.g., the shapes and capacities of edges and the points of interest along edges [63]. Such information is not always available in digital maps and can be difficult to obtain. However, it is of interest to extend the proposed methods to take additional information, when available, into account. To achieve general applicability of the paper's methods, we minimise the requirements of the input graph G'': both *PRTC* and *DATC* rely solely on the topology of a road network, which can be obtained easily from any digital map.

The weight annotation problem is finally handled by solving a system of linear equations, i.e., Equation 4.12. Alternative edge similarity metrics (e.g., considering the shapes and capacities of edges) can be easily incorporated into the linear system by

adding new terms of the form $\varphi \cdot \mathbf{L_M}$, where φ is the hyper-parameter and $\mathbf{L_M}$ is the Laplacian matrix derived by an alternative similarity metric. An alternative similarity metric *sim* should satisfy symmetry: $sim(e_i, e_j) = sim(e_j, e_i)$. Both *PRTC* and *DATC* satisfy symmetry.

The core operations in solving a system of linear equations using a conjugate gradient algorithm are matrix multiplication and transposition. This means that existing scalable matrix computation algorithms [64, 65] can be applied directly to make the proposed framework scalable and applicable to large road networks.

4.4 Experimental Study

We study the effectiveness of the proposed method for weight annotation of road networks with both *travel time* (*TTWA*) and *GHG emissions* (*GEWA*).

4.4.1 Experimental Setup

Road Networks: We use two road networks. The SK network is from Skagen, Denmark and has a primal graph with 543 vertices and $1,244$ edges. The NJ network contains almost all of North Jutland, Denmark and has a primal graph with $17,956$ vertices and $39,372$ edges.

Trips: We use GPS observations collected from 28 vehicles in the period 2007-10-01 to 2007-10-15. When the vehicles were moving, positions were sampled at 1 Hz. The data is collected as part of an experiment where young drivers start out with a substantial rebate on their car insurance and then are warned if they exceed the speed limit and are penalised financially if they continue to speed.

We apply an existing tool for map matching GPS observations onto road segments, thus obtaining 431 trips in the SK network and $11,516$ trips in the NJ network.

For *TTWA*, we use the total travel time for each trip, which can be obtained directly from the GPS observations of the trip, as the cost.

For *GEWA*, we use the GHG emissions of each trip as trip cost. Ideally, the exact fuel consumption should be obtained from CAN bus sensor data. Since such data is hard to obtain in a scalable fashion, we use instead the VT-micro model [40] that is able to compute the GHG emissions of trips based on the instantaneous velocities and accelerations derived from the GPS records of the trips in a robust fashion [44]. The 1 Hz GPS sampling frequency makes the VT-Micro model easy to use.

Traffic Category Tags: In transporation research, *PEAK* and *OFFPEAK* periods are used widely to distinguish different traffic flows over the course of a day [66]. Thus, we use *PEAK* and *OFFPEAK* as traffic category tags. Further, we distinguish between weekdays from weekend days, as traffic differs between weekdays and weekend days. To appropriately assign *PEAK* and *OFFPEAK* tags to the data set, we plot the numbers of GPS records according to their corresponding observed time at an one-hour granularity for weekdays and weekend days, respectively. Based on the generated histograms, we identify *PEAK* and *OFFPEAK* periods for weekdays. We find no clear peak periods during weekends and thus use *WEEKENDS* as the single tag for weekends. Table 4.3 provides the mapping (i.e., the function $G.F$) from time periods to tags.

T-Drive [51] is able to assign distinct and fine-grained traffic tags to individual edges. The precondition of the method is that sufficient GPS data is associated with

Periods		Tags
Weekdays	[0:00, 7:00)	*OFFPEAK*
Weekdays	[7:00, 8:00)	*PEAK*
Weekdays	[8:00, 15:00)	*OFFPEAK*
Weekdays	[15:00, 17:00)	*PEAK*
Weekdays	[17:00, 24:00)	*OFFPEAK*
Weekends	[0:00, 24:00)	*WEEKENDS*

Table 4.3: Traffic Category Tag Function $G.F$

edges. However, a substantial fraction of all edges have no GPS data in our setting. Thus, we use traffic tags at the coarse granularity shown in Table 4.3.

Implementation Details: The PageRank computation is implemented in C using the iGraph library version 0.5.4 [67]. All remaining experiments are implemented in Java, where the conjugate gradient algorithm for solving a linear system is implemented using the MTJ (matrix-toolkits-java) package [68].

We use the threshold 0.95 to filter the entries in the PageRank-based similarity matrix **A** (Equation 4.7): if the value of an entry in **A** is smaller than 0.95, the entry is set to 0. We use the speed limits associated with roads to classify the edges into two categories, *highways* (with speed limits above 90 km/h) and *urban roads* (with speed limits below 90 km/h). We only apply adjacency constraint on pairs of edges in the same category.

Due to the space limitation, the experiments only report the results using the best set of hyper-parameters, which are is obtained by manual tuning on a separate data set using cross validation. This is a well known method [30] for choosing hyper-parameters.

4.4.2 Experimental Results

4.4.2.1 Effectiveness Measurements

To gain insight into the accuracy of the obtained trip cost based weights, we split the set of $(trip, cost)$ pairs into a training set $\mathbb{TC}_{train}$ and a testing set $\mathbb{TC}_{test}$. We use the the training set to annotate the spatial network with weights, and we use the the testing set to evaluate the accuracy of the weights. In the following experiments, we randomly choose 50% of the pairs for training and the remaining 50% for testing, unless explicitly stated otherwise.

Since no ground-truth time-dependent weights exist for the two road networks, the accuracy of the obtained weights can only be evaluated using the trips in testing set $\mathbb{TC}_{test}$. If the obtained weights (using $\mathbb{TC}_{train}$) actually reflect the travel costs, the difference between the actual cost and the estimated cost using the obtained weights (i.e., by using Equation 4.1 defined in Section 4.2.2) for each trip in the testing set $\mathbb{TC}_{test}$ should be small.

We use the sum of squared loss (SSL) value (defined in Equation 4.13) between the actual cost $c^{(i)}$ and the estimated cost $cost(t^{(i)})$ over every trip in the testing set $\mathbb{TC}_{test}$ to measure the accuracy of the obtained weights.

$$SSL(\mathbb{TC}_{test}) = \sum_{(t^{(i)},c^{(i)})\in\mathbb{TC}_{test}} (c^{(i)} - cost(t^{(i)}))^2 \tag{4.13}$$

For example, if the GHG emissions based weights really reflect the actual GHG emissions, the sum of squared loss between the actual GHG emissions and the estimated

GHG emissions over every testing trip should tend to be small. The smaller the sum of squared loss, the more accurate the weights.

To gain insight into the effectiveness of the proposed objective functions, we compare four combinations of the functions:

1. F_1=$RSS(\mathbf{d}) + \gamma \cdot ||\mathbf{d}||_2^2$.
2. F_2=$RSS(\mathbf{d}) + \alpha \cdot PRTC(\mathbf{d}) + \gamma \cdot ||\mathbf{d}||_2^2$.
3. F_3=$RSS(\mathbf{d}) + \beta \cdot DATC(\mathbf{d}) + \gamma \cdot ||\mathbf{d}||_2^2$.
4. F_4=$RSS(\mathbf{d}) + \alpha \cdot PRTC(\mathbf{d}) + \beta \cdot DATC(\mathbf{d}) + \gamma \cdot ||\mathbf{d}||_2^2$.

Function F_1 only considers the residual sum of squares. Functions F_2 and F_3 take into account the PageRank-based topological constraint and the directional adjacency constraint, respectively. Function F_4 takes into account both constraints.

As the objective function used in trajectory regression [48] also considers adjacency, we can view the method using function F_3 as an improved version of trajectory regression because (i) function F_3 works not only for travel times, but also other travel costs, e.g., GHG emissions; (ii) function F_3 considers the temporal variations of travel costs, while trajectory regression does not; and (iii) function F_3 considers directional adjacency, while trajectory regression models a road network as a undirected graph and only considers undirected adjacency.

The sum of squared loss value for using objective function F_i is denoted as $SSL_{F_i}(\mathbb{TC}_{test})$. In order to show the relative effectiveness of the proposed objective functions, we report the ratios $Ratio_{F_2}$=$\frac{SSL_{F_2}(\mathbb{TC}_{test})}{SSL_{F_1}(\mathbb{TC}_{test})}$, $Ratio_{F_3}$=$\frac{SSL_{F_3}(\mathbb{TC}_{test})}{SSL_{F_1}(\mathbb{TC}_{test})}$, and $Ratio_{F_4}$=$\frac{SSL_{F_4}(\mathbb{TC}_{test})}{SSL_{F_1}(\mathbb{TC}_{test})}$.

Coverage, defined in Equation 4.14, is introduced as another measurement.

$$Cove_{F_i}(\mathbb{TC}_{train}) = \frac{|\{e|e \in G.\mathbb{E} \wedge annotated(e)\}|}{|G.\mathbb{E}|}, \tag{4.14}$$

where $annotated(e)$ holds if edge e is annotated with weights using $\mathbb{TC}_{train}$. Function $Cove_{F_i}$ indicates the ratio of the number of edges whose weights have been annotated by using objective function F_i to the total number of edges in the road network. The higher the coverage is, the more edges in the road network are annotated with weights, and thus the better performance.

4.4.2.2 Travel Time Based Weight Annotation

Effectiveness of objective functions: Table 4.4 reports the results on travel time based weight annotation. Column SSL_{F_1} reports the absolute *SSL* values over all test trips when using objective function F_1 for both data sets. NJ has much larger *SSL* values than SK because it has much more testing trips. For both road networks, the weights annotated using objective function F_4 have the least *SSL* values.

	SSL_{F_1}	$Ratio_{F_2}$	$Ratio_{F_3}$	$Ratio_{F_4}$
SK	88,656	99.2%	44.0%	43.8%
NJ	14,823,752	92.2%	49.2%	43.1%

Table 4.4: Effectiveness on *TTWA*

We also observe that the PageRank based topological constraint works more effectively on NJ than on SK. The reason is that Skagen is a small town in which few road

segments have similar topology (e.g., similar weighted PageRank values). In the NJ network, the PageRank based topological constraint gives a better accuracy improvement since more road segments have similarly weighted PageRank values.

The coverage reported in Table 4.5 also justifies the observation. When using objective function F_1, only the edges in the set of training trips can be annotated, which can be expected to be a small portion of the road network. When using objective function F_2, the coverage of the SK network increases much less than for the NJ network. This suggests that in a large road network, the PageRank based topological constraint substantially increases the coverage of the annotation, thus improving the overall annotation accuracy.

	$Cove_{F_1}$	$Cove_{F_2}$	$Cove_{F_3}$	$Cove_{F_4}$
SK	22.8%	28.8%	100%	100%
NJ	34.8%	86.7%	99.6%	100%

Table 4.5: Coverage of Weight Annotation

The directed adjacency topological constraint yields similar accuracy improvements on both road networks, and the accuracy improvement is more substantial than the improvement given by the PageRank based topological constraint. This is as expected because a road network is fully connected, and *DATC* is able to finally affect almost every edge, which gives more information for the edges that are not traversed by trips in the training set. This can be observed from the third column of Table 4.5.

For both road networks, *PRTC* and *DATC* together give the best accuracy, as shown in column $Ratio_{F_4}$ in Table 4.4. This finding offers evidence of the overall effectiveness of the proposed objective functions.

Accuracy comparison with a baseline: The test tips contain edges that are not covered by any training trips. Therefore, existing methods [51] that can estimate travel time based on historical data are inapplicable as baseline.

If the speed limit of every edge in a road network is available, we can use speed limit derived weights as a baseline for travel time based weight annotation. While it is difficult to obtain a speed limit for every road segment in a road network, we can use default values were values are missing. In the NJ network, 62 edges lack a speed limit and are assigned a default value (50 km/h).

Given an edge e and its speed limit $sl(e)$ and length $G.L(e)$, the corresponding travel time based weight for e is $\lambda \cdot \frac{G.L(e)}{sl(e)}$ if e is an urban road (where $\lambda \geq 1$) and $\frac{G.L(e)}{sl(e)}$ if e is a highway.

The factor λ is used because vehicles tend to travel at speeds below the speed limit on urban roads and at the speed limit on highways. Previous work [48] uses $\lambda = 2$, meaning that vehicles normally travel at half the speed limit in urban regions. However, we find that $\lambda = 1$ works the best for our data. The reason may be two-fold: (i) the data we use is collected from young drivers who tend to drive more aggressively than average drivers. (ii) the SK and NJ networks are relatively congestion-free when compared to Kyoto, Japan, which is simulated in previous work [48].

The above allows us to treat the speed limit derived weights as a baseline method for travel time based weight annotation. To observe the accuracy of the baseline method, its accuracy is also evaluated using *SSL* over every testing trip. Specifically, the baseline with $\lambda = 2$ is denoted as $SSL_{BL,\lambda=2}(\mathbb{TC}_{test})$, and the baseline with $\lambda = 1$ is denoted as $SSL_{BL,\lambda=1}(\mathbb{TC}_{test})$. The two resulting baselines are compared with the proposed method, and the results are reported in Table 4.6, where $Ratio_{\lambda=2}$=$\frac{SSL_{F_4}(\mathbb{TC}_{test})}{SSL_{BL,\lambda=2}(\mathbb{TC}_{test})}$ and

$Ratio_{\lambda=1}$=$\frac{SSL_{F_4}(\mathbb{TC}_{test})}{SSL_{BL,\lambda=1}(\mathbb{TC}_{test})}$. The ratios $Ratio_{\lambda=1}$ on the two road networks show that the weights obtained by our method are substantially better than the best cases of the weight obtained from the speed limits.

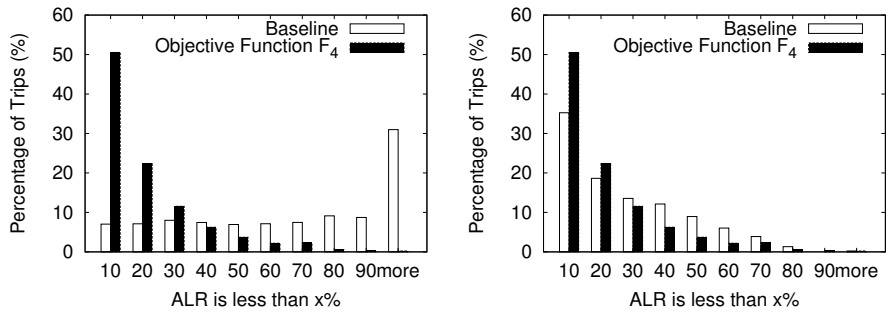
	$Ratio_{\lambda=2}$	$Ratio_{\lambda=1}$
SK	36.0%	78.8%
NJ	24.2%	90.8%

Table 4.6: Comparison With Baselines on *TTWA*

The same deviation has quite a different meaning for long versus short trips. For example, a 50-second deviation can be considered as a very good estimation error for a 30-minute trip, while it is a poor estimation error for a 2-minute trip. Thus, to better understand how the overall *SSL* values are distributed, we plot the number of test trips whose *absolute loss ratio* (*ALR*) values are within x percentage in Fig. 4.7. Given a test pair $(t^{(i)}, c^{(i)}) \in \mathbb{TC}_{test}$, its *ALR* value equals the absolute difference between the estimated and actual costs divided by the actual cost, as defined in Equation 4.15.

$$ALR((t^{(i)},c^{(i)})) = \frac{absolute(cost(t^{(i)}) - c^{(i)})}{c^{(i)}} \tag{4.15}$$

Our method shows the best result as the majority of the test trips have smaller *ALR* values. Assume that we consider and *ALR* below 30% as a good estimation. Fig. 4.7 shows that 84.3% of test trips have good estimations using the proposed method. In contrast, only 67.4% and 22.1% of test trips have good estimations using baseline methods with $\lambda = 1$ and $\lambda = 2$, respectively.



(a) Baseline with $\lambda = 2$

(b) Baseline with $\lambda = 1$

Figure 4.7: *ALR* Comparison on *TTWA* of NJ

We do not integrate speed limits into our method because (i) for edges without available speed limits, the obtained weights are quite sensitive to the assigned default speed limits: inaccurate defaults deteriorate the performance severely; and (ii) speed limits do not give obvious benefits when annotating edges with GHG emissions based weights, as we will see shortly in Section 4.4.2.3 (in particular, in Fig. 4.8).

4.4.2.3 GHG Emissions Based Weight Annotation

Effectiveness of objective functions: Table 4.7 reports the results on GHG emissions based weight annotation. In general, the results are consistent with the results from

	SSL_{F_1}	$Ratio_{F_2}$	$Ratio_{F_3}$	$Ratio_{F_4}$
SK	175.931	99.9%	40.3%	30.0%
NJ	87,362,465	94.5%	66.2%	44.3%

Table 4.7: Effectiveness on *GEWA*

the travel time based weight annotation (as shown in Table 4.4): (i) The PageRank-based topological constraint works more effectively on the NJ network than on the SK network; (ii) the directed adjacency constraint works more effectively than the PageRank-based topological constraint; (iii) the weights obtained by using both *PRTC* and *DATC* give the best accuracy. The coverage when using the different objective functions is exactly the same as what was reported in Table 4.5.

Comparison with a baseline: As we did for travel times, we use speed limits to devise a baseline for GHG emissions based weight annotation. Assuming a vehicle travels on an edge at constant speed (e.g., the speed limit of the edge), we can simulate a sequence of instantaneous velocities. For example, let an edge be 100 meters long and the speed limit be 60 km/h. The simulated trip on the road segment is represented by a sequence of 6 records, each with 60 km/h as the instantaneous velocity. This allows us to apply the VT-micro model to estimate GHG emissions based edge weights. Since in the previous set of experiments, we have already found that the speed limit (i.e., $\lambda = 1$) is the best fit for our data we simply use the speed limit here.

We obtain $Ratio_{\lambda=1} = 24.7\%$ for SK and $Ratio_{\lambda=1} = 29.8\%$ on NJ. Fig. 4.8 shows the percentage of test trips whose *ALR* values are less than *x%* using the baseline with $\lambda = 1$ and the proposed method, respectively. These results clearly show the better performance of the proposed method, as the majority of test trips have smaller *ALR* values.

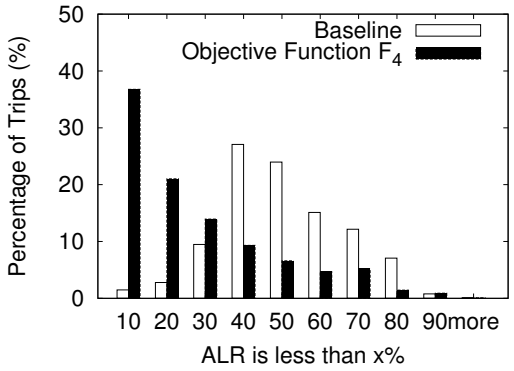


Figure 4.8: *ALR* Comparison on *GEWA* of NJ

4.4.2.4 Effectiveness of the Size of Training Trips

In this section, we study the accuracy when varying the training set size. Specifically, on the NJ network, we reserve 20% of the (*trip*, *cost*) pairs as the testing set, denoted as $\mathbb{TC}_{test}$, and the remaining 80% as the training set, denoted as $\mathbb{TC}_{train}$. In order to observe the accuracy of weight annotation on different sizes of $\mathbb{TC}_{train}$, we use 100%, 80%, 60%, 40% and 20% of $\mathbb{TC}_{train}$ to annotate the weights, respectively. The results are shown in Fig. 4.9.

For travel time, when only 20% of $\mathbb{TC}_{train}$ is used, the accuracy of our method is

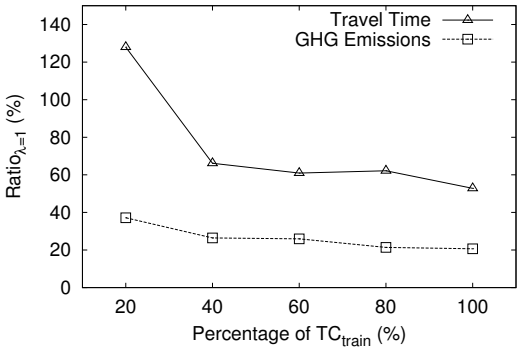


Figure 4.9: Results on Different Size of $\mathbb{TC}_{train}$

worse than the baseline method with $\lambda = 1$ because the baseline has a rough estimation for the costs of all edges, while the 20% of $\mathbb{TC}_{train}$ covers only 16.3% of the edges in the road network. Although our method propagates weights to edges that are not covered by the training trips, the accuracy suffers when the initial coverage of the training trips is low. When 40% of $\mathbb{TC}_{train}$ is used, the accuracy of our method is much better than that of the baseline. In this case, the training trips cover 23.3% of all edges. As the training set size increases, the accuracy of the travel time weights also increases. When we use all trips in $\mathbb{TC}_{train}$, the accuracy of our method is almost twice that of the baseline.

For GHG emissions, we observe a similar trend: with more training trips, the accuracy of the corresponding weights improves, and our method always outperforms the baseline when annotating edges with GHG emissions based weights.

This experiment justifies that (i) our method works effectively even when the coverage of the trips in the training set is low; (ii) if the coverage of the trips in the training set increases, e.g., by providing more $(trip, cost)$ pairs as training set, the accuracy of the obtained weights also increases.

4.5 Conclusion and Outlook

Reduction in GHG emissions from transportation calls for effective eco-routing, and road network graphs where all edges are annotated with accurate weights that capture environmental costs, e.g., fuel usage or GHG emissions, are needed for eco-routing. However, such weights are not always readily available for a road network. This paper proposes a general framework that takes as input a collection of $(trip, cost)$ pairs and assigns trip cost based weights to a graph representing a road network, where trip cost based weights may reflect GHG emissions, fuel consumption, or travel time. By using the framework, edge weights capturing environmental impact can be computed for the whole road network, thus enabling eco-routing. To the best of our knowledge, this is the first work that provides a general framework for assigning trip cost based edge weights based on a set of $(trip, cost)$ pairs.

Two directions for future work are of particular interest. It is of interest to explore whether accuracy improvement is possible by using distinct *PEAK* and *OFFPEAK* tags for different road segments. Likewise, it is of interest to explore means of updating weights in real time. A module that takes as input real time streaming data, e.g., real

time GPS observations along with costs, can be incorporated into the framework.

Chapter 5

Risk Assessment and Conclusions

5.1 Risk Assessment

The main risk in the work package lies in the difficulty of obtaining real fuel consumption data on the majority of road segments in Denmark, in order to evaluate the accuracy of the proposed techniques. To compensate for this deficiency, we have obtained a small amount of CANBus data to record the real fuel consumption on some road segments. We have also conducted additional experiments to compare the fuel consumption estimated by existing models to the fuel recorded by the small amount of CANBus data. The conclusion is that it is possible to use GPS data on fuel estimation models to estimate fuel consumption.

Since the proposed algorithms and methods are designed purely based on raw GPS trajectories, they can be employed independently of CANBus data. Thus, the problem of lacking real fuel consumption data in conducting the research is reduced to a low level.

5.2 Conclusions

Reduction in GHG emissions from road transportation calls for effective eco-routing, and road network graphs where all edges are annotated with accurate weights that capture environmental costs, e.g., fuel usage or GHG emissions, are needed for eco-routing. However, such weights are not always readily available for a road network.

This deliverable D3.4 describes advanced eco-routing techniques developed and implemented to solve task "T3.3 Advanced Eco-Routing Methods of WP3". In this deliverable, we study and propose techniques to derive eco-weights for a road network. With the eco-weights, existing route algorithms can be employed to compute eco-routes for any given source-destination pairs. In particular, we study (1) how to assign time-dependent, uncertain eco-weights to road segments that are covered by sufficient GPS records; (2) how to update uncertain eco-weights as real-time GPS data streams in; (3) how to assign time-dependent eco-weights to road segments that are uncovered or insufficiently covered by GPS records.

The proposed methods are evaluated using a large GPS data set collected in Den-

mark. The experimental results indicate that these new methods are effective, efficient, and scalable up to country-level road networks. This suggests that the objectives of D3.4, which includes (1) to develop efficient eco-routing algorithms, (2) to develop scalable eco-routing algorithms, and (3) to conduct extensive experimental evaluations, have been fully achieved. Further, the proposed methods have been partially used as part of a prototype system[1] developed by AAU and AU.

Deliverable D3.4 has significantly improved state-of-the-art approaches, and all objectives of the deliverable were fully met.

[1] `http://daisy.aau.dk/its`

Glossary

Term	**Description**
GHG	Green House Gas.
GPS	Global Positioning System.
CANBus	Detailed information about a single vehicle, e.g., actual fuel consumption, engine RPM, and throttle position.
Trajectory	A sequence of GPS observations.
STHMM	Spatio-temporal Hidden Markov Model.
GEWA	GHG emissions weight annotation.
TTWA	Travel times weight annotation.
MCR	Memory compression ratio.
ALR	Absolute loss ratio.
TT	Travel times.
GE	GHG emissions.
ERN	Eco Road Network.
NMI	Normalised mutual information.

Table 5.1: Glossary

Bibliography

[1] Yu Ma, Bin Yang, and Christian S Jensen. Representing time-dependent uncertain eco-weights for road networks using histograms. *IEEE Transaction on Intelligent Transportation Systems*, In Submission.

[2] Bin Yang, Chenjuan Guo, and Christian S Jensen. Travel cost inference from sparse, spatio-temporally correlated time series using markov models. *Proceedings of the VLDB Endowment*, 6, 2013.

[3] Bin Yang, Manohar Kaul, and Christian S Jensen. Using incomplete information for complete weight annotation of road networks. *TKDE*, to appear.

[4] Bolin Ding, Jeffrey Xu Yu, and Lu Qin. Finding time-dependent shortest paths over large graphs. In *EDBT*, pages 205–216, 2008.

[5] F.C. Pereira, H. Costa, and N.M. Pereira. An off-line map-matching algorithm for incomplete map databases. *European Transport Research Review*, 1(3):107–124, 2009.

[6] Climate change: Causes. `http://climate.nasa.gov/causes/`.

[7] Chenjuan Guo, Yu Ma, Bin Yang, Christian S. Jensen, and Manohar Kaul. Ecomark: Evaluating models of vehicular environmental impact. In *GIS*, 2012.

[8] Bolin Ding, Jeffrey Xu Yu, and Lu Qin. Finding time-dependent shortest paths over large graphs. In *Proc. EDBT*, pages 205–216, 2008.

[9] Evangelos Kanoulas, Yang Du, Tian Xia, and Donghui Zhang. Finding fastest paths on a road network with speed patterns. In *Proc. ICDE*, pages 10–21, 2006.

[10] Jarek Gryz and Hu Bin. The review of histogram (the reading report of course 6421).

[11] Yannis E. Ioannidis and Viswanath Poosala. Balancing histogram optimality and practicality for query result size estimation. In *Proc. SIGMOD*, pages 233–244, 1995.

[12] Viswanath Poosala, Yannis E. Ioannidis, Peter J. Haas, and Eugene J. Shekita. Improved histograms for selectivity estimation of range predicates. In *Proc. SIGMOD*, pages 294–305, 1996.

[13] Hamid Mousavi and Carlo Zaniolo. Fast and accurate computation of equi-depth histograms over data streams. In *Proc. EDBT*, pages 69–80, 2011.

[14] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Wavelet-based histograms for selectivity estimation. In *Proc. SIGMOD*, pages 448–459, 1998.

[15] Graham Cormode and Minos N. Garofalakis. Histograms and wavelets on probabilistic data. *CoRR*, abs/0806.1071, 2008.

[16] Yann Disser, Matthias Müller-Hannemann, and Mathias Schnee. Multi-criteria shortest paths in time-dependent train networks. In *Proc. WEA*, pages 347–361, 2008.

[17] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: driving directions based on taxi trajectories. In *ACM SIGSPATIAL*, pages 99–108, 2010.

[18] Yannis Ioannidis. The history of histograms (abridged). In *Proc. VLDB*, pages 19–30, 2003.

[19] Francisco Cmara Pereira, Hugo Costa, and Nuno Martinho Pereira. An off-line map-matching algorithm for incomplete map databases. *European Transport Research Review*, pages 107–124, 2009.

[20] Amara Graps. An introduction to wavelets. *IEEE Computational Science and Engineering*, 2:50–61, 1995.

[21] Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Dynamic maintenance of wavelet-based histograms. In *Proc. VLDB*, pages 101–110, 2000.

[22] Sejoon Lim, Christian Sommer, Evdokia Nikolova, and Daniela Rus. Practical route planning under delay uncertainty: Stochastic shortest path queries. In *Robotics: Science and Systems*, 2012.

[23] Evdokia Nikolova, Matthew Brand, and David R. Karger. Optimal route planning under uncertainty. In *ICAPS*, pages 131–141, 2006.

[24] Anthony Chen and Zhaowang Ji. Path finding under uncertainty. *Journal of advanced transportation*, 39(1):19–37, 2005.

[25] Ming Hua and Jian Pei. Probabilistic path queries in road networks: traffic uncertainty aware path selection. In *Proc. EDBT*, pages 347–358, 2010.

[26] Ajith B Wijeratne, Mark A Turnquist, and Pitu B Mirchandani. Multiobjective routing of hazardous materials in stochastic networks. *European Journal of Operational Research*, 65(1):33–43, 1993.

[27] Kyoungho Ahn, Hesham Rakha, Antonio Trani, and Michel Van Aerde. Estimating Vehicle Fuel Consumption and Emissions based on Instantaneous Speed and Acceleration Levels. *Journal of Transportation Engineering*, 128(2):182–190, 2002.

[28] Hesham Rakha, Kyoungho Ahn, and Antonio Trani. Development of vt-micro model for estimating hot stabilized light duty vehicle and truck emissions. *Transportation Research Part D: Transport and Environment, 9(1):*, pages 49–74, 2004.

[29] Peng Wang, Haixun Wang, and Wei Wang. Finding semantics in time series. In *SIGMOD Conference*, pages 385–396, 2011.

[30] C.M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[31] W. Zheng, D.H. Lee, and Q. Shi. Short-term freeway traffic flow prediction: Bayesian combined neural network approach. *Journal of transportation engineering*, 132(2):114–121, 2006.

[32] Evangelos Kanoulas, Yang Du, Tian Xia, and Donghui Zhang. Finding fastest paths on a road network with speed patterns. In *ICDE*, page 10, 2006.

[33] Nirmesh Malviya, Samuel Madden, and Arnab Bhattacharya. A continuous query system for dynamic route planning. In *ICDE*, pages 792–803, 2011.

[34] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[35] C.D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[36] M. Brand. Coupled hidden markov models for modeling interacting processes. Technical report, MIT Media Lab Perceptual Computing and Common Sense.

[37] J. Kwon and K. Murphy. Modeling freeway traffic with coupled hmms. Technical report, Univ. California, Berkeley, 2000.

[38] S. Zhong and J. Ghosh. A new formulation of coupled hidden markov models. Technical report, Univ. of Texas at Austin, 2001.

[39] I. Rezek, P. Sykacek, and SJ Roberts. Learning interaction dynamics with coupled hidden markov models. *IEE Proceedings-Science, Measurement and Technology*, 147:345, 2000.

[40] K. Ahn, H. Rakha, A. Trani, and M. Van Aerde. Estimating vehicle fuel consumption and emissions based on instantaneous speed and acceleration levels. *Journal of Transportation Engineering*, 128(2):182–190, 2002.

[41] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. Driving with knowledge from the physical world. In *KDD*, pages 316–324, 2011.

[42] What is the EU doing on climate change? `http://ec.europa.eu/clima/policies/brief/eu/index_en.htm`.

[43] Reducing emissions from transport. `http://ec.europa.eu/clima/policies/transport/index_en.htm`.

[44] C. Guo, Y. Ma, B. Yang, C.S. Jensen, and M. Kaul. EcoMark: Evaluating models of vehicular environmental impact. In *GIS*, pages 269–278, 2012.

[45] T. Kono, T. Fushiki, K. Asada, and K. Nakano. Fuel consumption analysis and prediction model for co?route search. In *15th World Congress on Intelligent Transport Systems and ITS America's 2008 Annual Meeting*, 2008.

[46] E. Ericsson, H. Larsson, and K. Brundell-Freij. Optimizing route choice for lowest fuel consumption-potential effects of a new driver support tool. *Transportation Research Part C: Emerging Technologies*, 14(6):369–383, 2006.

[47] G. Tavares, Z. Zsigraiova, V. Semiao, and M.G. Carvalho. Optimisation of MSW collection routes for minimum fuel consumption using 3D GIS modelling. *Waste Management*, 29(3):1176–1185, 2009.

[48] Tsuyoshi Idé and Masashi Sugiyama. Trajectory regression on road networks. In *AAAI*, pages 203–208, 2011.

[49] T. Idé and S. Kato. Travel-time prediction using gaussian process regression: A trajectory-based approach. In *SDM*, pages 1183–1194, 2009.

[50] S. Clark. Traffic prediction using multivariate nonparametric regression. *Journal of Transportation Engineering*, 129(2):161–168, 2003.

[51] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: driving directions based on taxi trajectories. In *GIS*, pages 99–108, 2010.

[52] J. Ygnace, C. Drane, Y. B. Yim, and L. Renaud. Travel time estimation on the san francisco bay area network using cellular phones as probes. Technical report, Institute of Transportation Studies, UC Berkeley, 2000.

[53] J. C. Herrera and A. M. Bayen. Incorporation of lagrangian measurements in freeway traffic state estimation. *Transportation Research Part B: Methodological*, 44(4):460–481, 2010.

[54] G. Song, L. Yu, Z. Wang, et al. Aggregate fuel consumption model of light-duty vehicles for evaluating effectiveness of traffic management strategies on fuels. *Journal of Transportation Engineering*, 135:611, 2009.

[55] E. Köhler, K. Langkau, and M. Skutella. Time-expanded graphs for flow-dependent transit times. In *ESA*, pages 599–611, 2002.

[56] B. George and S. Shekhar. Time-aggregated graphs for modeling spatio-temporal networks. In *ER (Workshops)*, pages 85–99, 2006.

[57] B. Jiang. Ranking spaces for predicting human movement in an urban environment. *International Journal of Geographical Information Science*, 23(7):823–837, 2009.

[58] E. Crisostomi and et al. A google-like model of road network dynamics and its application to regulation and control. *International Journal of Control*, 84(3):633–651, 2011.

[59] A. N. Langville and C. D. Meyer. Survey: Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2003.

[60] C. F. Daganzo and Y. Sheffi. On stochastic models of traffic assignment. *Transportation Science*, 11(3):253–274, 1977.

[61] B. Jiang, S. Zhao, and J. Yin. Self-organized natural roads for predicting traffic flow: a sensitivity study. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:7008–7035, 2008.

[62] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[63] N.J. Yuan, Y. Zheng, L. Zhang, and X. Xie. T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Transactions on Knowledge and Data Engineering*, 2012.

[64] S. Seo, E.J. Yoon, J. Kim, S. Jin, J.S. Kim, and S. Maeng. Hama: An efficient matrix computation with the mapreduce framework. In *CloudCom*, pages 721–726, 2010.

[65] J. Lin and C. Dyer. Data-intensive text processing with mapreduce. *Synthesis Lectures on Human Language Technologies*, 3(1):1–177, 2010.

[66] P. Cantos-Sanchez, R. Moner-Colonques, J.J. Sempere-Monerris, and A. Alvarez-SanJaime. Viability of new road infrastructure with heterogeneous users. *Transportation Research, Part A*, 45(5):435–450, 2011.

[67] igraph library. `http://igraph.sourceforge.net/`.

[68] matrix-toolkits-java package. `http://code.google.com/p/matrix-toolkits-java`.