Managing and Analyzing Big Traffic Data-An Uncertain Time Series Approach

Hu, Jilin

DOI (link to publication from Publisher):
10.54337/aau306584084

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

MANAGING AND ANALYZING BIG TRAFFIC DATA—AN UNCERTAIN TIME SERIES APPROACH

BY
JILIN HU

DISSERTATION SUBMITTED 2018

AALBORG UNIVERSITY
DENMARK

Managing and Analyzing Big Traffic Data—An Uncertain Time Series Approach

Ph.D. Dissertation

Jilin Hu

Dissertation submitted December 2018

Dissertation submitted: December 2018

PhD supervisor: Prof. Christian S. Jensen
Aalborg University

PhD Co-Supervisor: Prof. Bin Yang
Aalborg University

PhD committee: Associate Professor Christian Thomsen (chairman)
Aalborg University

Professor Matthias Renz
Christian-Albrechts-Universität zu Kiel

Professor Ira Assent
Aarhus University

PhD Series: Technical Faculty of IT and Design, Aalborg University

Department: Department of Computer Science

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-364-8

Published by:
Aalborg University Press
Langagervej 2
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2018

Abstract

We are witnessing a global trend towards urbanization that creates larger and larger cities. This development results in increased demands for mobility. At the same time, it also brings with it increased congestion, increased greenhouse gas emissions, and reduced air quality.

Meanwhile, with the increasingly digitalization of vehicular transportation, notably the deployment of GPS devices, e.g., in smartphones and vehicle navigation devices, and sensors embedded in roads, massive volumes of data are generated that capture the traffic state of a road network. This data foundation and the above problems call for new data analysis techniques that enable high-resolution vehicular transportation services that contribute to addressing the problems.

In this thesis, we adopt an uncertain time series (UTS) approach to capture both uncertainty and temporal dependency. We solve the following three problems: 1) data sparseness in UTSs in a road network; 2) decision making among several UTSs at each time interval; and 3) future forecasting for UTSs with spatio-temporal correlations.

First, traffic data is uncertain and spatio-temporally sparse. It is common that one road segment may be traversed by several vehichles at different speeds at the same time such that an uncertain or stochastic weight is the most appropriate way of capturing the travel time. Yet, it is also normal that some road segments have no vehicle traversals during some time periods, and it is an almost impossible task to collect enough traversal data to cover all edges in a road network with stochastic weights at all times. All time-dependent stochastic weights on edges can be modeled as UTSs, we call this problem *data sparseness in UTSs in a road network*. To address this problem, we provide a general, data-driven Graph Convolutional Weight Completion (GCWC) framework to assign a stochastic weight to each edge. We further improve the performance of GCWC by incorporating contextual information.

Second, UTSs on edges in a road network result in multiple UTSs for a pair of an origin and a destination since multiple candidate paths exist for the same pair. Therefore, we address the problem of selecting the optimal

paths from the set of all candidates, which we call *decision making among several UTSs at each time interval*. We develop techniques to find optimal paths with non-dominated costs that take the risk preferences of users into consideration. We investigate the relationships among risk preferences, categories of utility function, and different kinds of stochastic dominance. To improve the efficiency of comparing multiple UTSs, we propose a grouping strategy and a merge-sort like procedure.

Finally, given a partitioning of a road network into regions, we construct UTSs for Origin-Destination (OD) pairs. To estimate the future stochastic weights of any OD pair, we consider the problem of *future forecasting for UTSs with spatio-temporal correlations.* We propose a stochastic OD matrix forecasting framework that encompasses factorization, forecasting and recovery. We use machine learning techniques to address the data spareness problem in historical OD matrices and temporal dynamics in forecasting; specifically, we utilize Fully-Connected (FC) layers and Gated Recurrent Units (GRU), respectively, in a basic generic framework. Finally, we propose a dual-stage graph convolutional, recurrent neural network to better capture the spatio-temporal correlations in an advanced framework.

We evaluate the proposed methods and frameworks by utilizing of five real data sets, four of which are traffic data sets from three different countries, specifically Denmark, China, and U.S.A., and four different places, specifically Aalborg, Chengdu City, highway toll gate in China, and New York City. Moreover, the data includes both GPS taxi data and loop detector data, which are two important sources of traffic data. The experiments conducted in each paper offer detailed insight into the efficiency and effectiveness of the proposed approaches.

Resumé

Vi oplever en igangværende udvikling mod en urbanisering der skaber større og større byer. Denne udvikling har resulteret i større efterspørgsel til mobilitet, og som konsekvens af dette forekommer en øget overbelastning,og en større mængde drivhusgasser udledes der reducerer luftkvaliteten.

I takt med digitaliseringen af køretøjer skabes store mængder af data der indkapsler tilstande for vejnetværk. Dette er især tilfældet gennem udbredelsen af GPS-enheder i eksempelvis smartphones og navigationsenheder, samt vej-indlejrede sensorer. Dette data-fundament og ovenstående problemstillinger skaber efterspørgsel efter nye analyseteknikker der muliggør køretøjs- og transport-services af høj opløsning, som kan bidrage til at adressere problemerne. I denne tese anvender vi uncertain time series (UTS) til at indfange både usikkerhed og temporal afhængighed. We løser følgende tre problemer: 1) Data sparsomhed i UTS for vejnetværk; 2) Beslutningstagen på tværs af flere UTS for hvert tidsinterval; 3) Fremtidig prognose af UTS med spatio-temporale korrelationer.

Først og fremmest er trafikdata er usikkert og spatio-temporalt tyndt. Det er normalt at et vejsegment bliver traveseret af flere køretøjer med forskellige hastigheder på samme tidspunkt således at en usikker eller stokastisk vægt er den passende måde at indfange rejsetid på. Samtidig er det også typisk at nogle vejsegmenter ikke har opmålte traverseringer over alle tidsintervaller, og det er en næsten umulig opgave at indsamle nok data til at dække alle kanter i et vejnetværk med stokastiske vægte over alle tidsintervaller. Alle tidsafhængige stokastiske vægte forbundet med kanter kan modelleres som UTS, dette problem kaldes data sparsomhed i UTS for et vejnetværk. For at adressere dette problem præsenterer vi et generaliseret, data-drevet Graph Convolutional Weight Completion (GCWC) framework til at tildele en stokastisk vægt til hver kant. Dernæst forbedrer vi GCWC's ydeevne yderligere ved at inkorporere kontekstuelle oplysninger.

For det andet resulterer UTS på kanter i et vejnetværk i flere UTS'er for hver kombination af kilde og destination da der findes mange valg af stier fra A til B. Derfor løser vi problemet med at udvælge den optimale sti fra et sæt af kandidater, som vi kalder beslutningstagen på tværs af flere UTS'er over

hvert tidsinterval. Vi udvikler teknikker til at finde optimale stier med ikke-dominerede omkostninger der tager højde for brugerens risikopræferencer. Vi undersøger forholdet mellem risikopræferencer, kategorier af hjælpefunktionalitet, og forskellige typer af stokastisk dominans. For mere effektisk at kunne sammenligne UTS'er udvikler vi en grupperingsstrategi og en merge-sort-lignende procedure.

Endelig konstruerer vi UTS for Origin-Destination (OD) par ved hjælp af en partitionering af vejnetværket. For at estimerer de fremtidige stokastiske vægte for et givet OD par, undersøger vi hvordan man kan lave prognoser for fremtidige UTS'er ved hjælp af spatio-temporale korrelationer. Vi foreslår et stokastisk OD matrix prognose framework, der omfatter faktorisering, prognoser og genoprettelse. Vi bruger maskinlæringsteknikker til at adressere datasparsomhed i historiske OD matricer og temporale dynamikker i prognoser; mere specifikt udnytter vi Fully-Connected (FC) lag, og Gated Recurrent Units (GRU) i et grundlæggende generisk framework. Til slut foreslår vi et to-stadies Graph Convolutional, Recurrent neuralt netværk for bedre at kunne indfange de spatio-temporale korrelationer i et avanceret framework. Vi evaluerer de foreslåede metoder og frameworks ved at anvende fem autentiske dataset, hvoraf fire af sættene er fra tre forskellige lande, Danmark, Kina og USA, og fire forskellige steder: Aalborg, Chengdu City, en beskatningsport på en motorvej i Kina, og New York City. Derudover inkluderer dette data også både GPS taxi data, og loop detektor data, som er to vigtige kilder til trafikdata. Eksperimenterne der udføres i hver artikel tilbyder detaljeret indsigt i effektiviteten af de foreslåede tilgange og metoder.

Acknowledgments

First, I would like to express my gratitude to my supervisor Chistian S. Jensen for giving me the opportunity to be one of his Ph.D. students. His dedication to academics is admirable. I am quite impressed by his attention to details and pursuit of perfection. All of these teach me how to be an excellent person. I feel very grateful for his instructions on my Ph.D. study.
Next, I would like to express my gratitude to my co-supervisor Bin Yang for recruiting me. I am so proud of being a member of such a great research group. Apart from my own office, Prof. Yang's office is my most frequently visited place in AAU. I am very appreciative of the time he has spent providing suggestions on my work, which are quite instructive. Without his help, I cannot have these publications and thesis.
Then, I would like to express my gratitude to Alexandre Bayen for giving me the opportunity to study at UC Berkeley, which is of great help to my research. I appreciate the funding from Otto Mønsteds Fond for this period of study. Also, I am really grateful to the Obel Family Foundation that funds my whole Ph.D. study.
Moreover, I would like to express my gratitude to all my colleagues at Daisy for providing such a cozy and friendly working environment. The life of a Ph.D. student is full of toughness and loneliness. It is really nice to have their company. I want to thank my co-authors, Chenjuan Guo, Jian Dai, Robert Waury, and Lu Chen, for giving me the chance to work with them and learn from them. Specifically, I want to thank Chenjuan for her dedication to revising my papers and teaching me how to write a good research paper. Moreover, I want to thank Simon Aagaard Pedersen for translating the Danish abstract of this thesis. I also thank the administrative personel in AAU who have made my Ph.D. life easier.
Lastly, this thesis would not become a reality without the support from my family and friends, especially my parents who raised me and my girlfriend who is always there to encourage me. I thank them for their patience. I am lucky to have all of you around me.

Contents

Thesis Details

Thesis Title: Managing and Analyzing Big Traffic Data-An Uncertain Time Series Approach
Ph.D. Student: Jilin Hu
Supervisors: Prof. Christian S. Jensen, Aalborg University
Co-Supervisors: Prof. Bin Yang, Aalborg University
The main body of the thesis consists of the following papers.

[A] **Jilin Hu**, Chenjuan Guo, Bin Yang, Christian S. Jensen, "Stochastic Weight Completion for Road Networks using Graph Convolutional Networks," ICDE 2019, 12 pages (to appear).

[B] **Jilin Hu**, Bin Yang, Chenjuan Guo, Christian S. Jensen, "Risk-aware path selection with time-varying, uncertain travel costs: a time series approach," VLDB J. 27(2): 179-200 (2018).

[C] **Jilin Hu**, Chenjuan Guo, Bin Yang, Christian S. Jensen, Lu Chen, "Stochastic Origin-Destination Matrix Forecasting Using Dual-Stage Graph Convolutional, Recurrent Neural Networks" (under review).

In addtion to the above papers, I have coauthored the following five papers as part of my Ph.D. studies, which are not included in the thesis.

[D] **Jilin Hu**, Bin Yang, Christian S. Jensen, and Yu Ma, "Enabling time-dependent uncertain eco-weights for road networks," GeoInformatica, 21(1): 57–88 (2017).

[E] Bin Yang, Jian Dai, Chenjuan Guo, Christian S. Jensen, **Jilin Hu**, "PACE: a PAth-CEntric paradigm for stochastic path finding," VLDB J. 27(2): 153-178 (2018).

[F] Chenjuan Guo, Bin Yang, **Jilin Hu**, Christian S. Jensen, "Learning to Route with Sparse Trajectory Sets," ICDE 2018: 1073-1084.

[G] Robert Waury, **Jilin Hu**, Bin Yang, Christian S. Jensen, "Assessing the Accuracy Benefits of On-the-Fly Trajectory Selection in Fine-Grained Travel-Time Estimation," MDM 2017: 240-245.

[H] Jian Dai, Bin Yang, Chenjuan Guo, Christian S. Jensen, **Jilin Hu**, "Path Cost Distribution Estimation Using Trajectory Data," PVLDB 10(3): 85-96 (2016).

This thesis has been submitted for assessment in partial fulfillment of the Ph.D. degree. The thesis is based on the submitted or published scientific papers listed above. Parts of the content of the papers in the main body of the thesis are used directly or indirectly in the extended summary part of the thesis. As part of the assessment, co-author statements have been made available to the assessment committee and are also available at the Faculty. The permission for using the published and accepted articles in the thesis have been obtained from the corresponding publishers with the condition that they are cited and copyright are placed prominently in the references. In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of Aalborg University's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to `http://www.ieee.org/publications_standards/publications/rights/rights_link.html` to learn how to obtain a License from RightsLink. If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

Part I

Thesis Summary

Chapter 1

Introduction

1.1 Background and Motivation

Transportation is an integral part of our daily lives. With the rapidly expansion of cities and sharply increasing populations, many more traffic induced problems are likely to occur in cities all around the world, e.g., traffic congestion [1], greenhouse gas (GHG) emissions [2], and degraded air quality [3]. In 2014, traffic congestion has caused citizens of 370 urban regions in U.S.A. to spend a total of 6.9 billion extra hours to travel, which amounts to a 159 billion dollar congestion cost [4].

Meanwhile, massive transportation data is being generated from GPS devices in vehicles and by traffic sensors in roads that capture the travel of people. This data is a very valuable information source: if we are able to successfully analyze this data, this can offer a foundation for more intelligent transportation. Recently, there has been many innovations in transportation, e.g., mobility-on-demand services, autonomous vehicles, and advanced traveler information systems (ATIS) [1], that all seek to address these traffic induced problems.

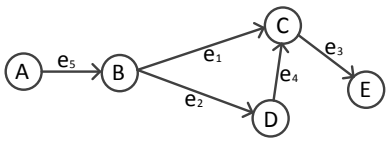
In most previous studies [5–7], a road network is modeled as a graph [23], and the average travel cost of a road segment, e.g., travel time or speed, is specified as a weight of the corresponding graph edge. However, this approach is not accurate since it ignores the presence of uncertainty. The uncertainty comes from the randomness in the underlying data. Notably, different cars on the same road segment have different travel speeds, sensor measurements may be imprecise and map matching methods may be imperfect [8]. Moreover, traffic data also shows strong temporal dynamics, i.e., current data is correlated with the data that is generated before and after.

These two characteristics, uncertainty and temporal dynamics, are fundamental to the research topic of this thesis—uncertain time series (UTS). We

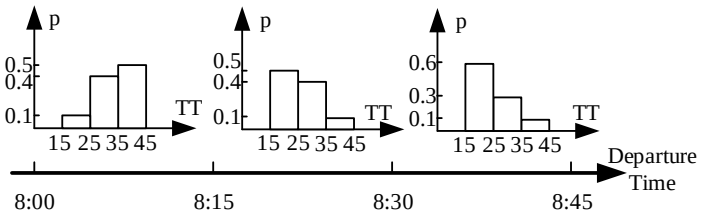
construct an UTS for each road segment such that a travel cost distribution is used to describe the travel cost, e.g., travel time or fuel consumption, during different intervals of time.

Example 1.1.1 (Road Network and UTS)
Figure 1.1(a) shows a road network that consists of 5 vertices and 5 directed edges. We have two paths, $P_1 = \{e_5, e_1, e_3\}$ and $P_2 = \{e_5, e_2, e_4, e_3\}$, from A to E. Based on historical observations, we can have two corresponding travel time distributions for the two paths: Path P_2 has a travel time histogram $\{[40, 50) : 0.3, [50, 60] : 0.7\}$ which means that P_2 offers 0.3 probability of arriving within 40 to 50 minutes and 0.7 probability of arriving within 50 to 60 minutes; thus, the expected travel time is 52 minutes. Path P_1 has histogram $\{[40, 50) : 0.6, [50, 60) : 0.3, [60, 70] : 0.1\}$, and the expected travel time is 50 minutes. Therefore, if we only consider the expected travel time, P_1 is the best path because it has the shorter expected travel time. However, if a user needs to arrive at v_3 within 60 minutes, the user runs a risk of arriving late if choosing P_1, while P_2 can guarantee an on-time arrival.

Figure 1.1(b) illustrates the travel time UTS of edge e_5. We can observe that the probability that the travel time falls into [15, 25) minutes is increasing gradually after 8:15 since the morning peak hour ends. This example also illustrates how traffic data demonstrates temporal dynamics.



(a) Road Network.



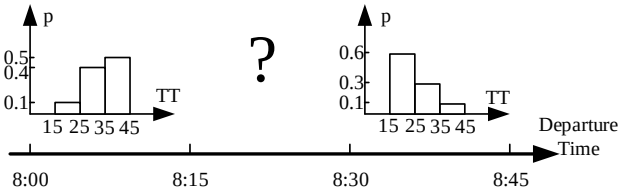
(b) UTS for e_5.

Fig. 1.1: Motivating Example(p denotes probability and TT denotes travel time in minutes).

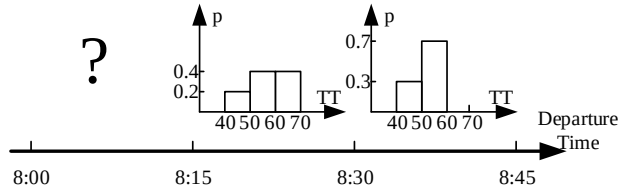
In this thesis, we address the following three problems.

1) **Data Sparseness.** A necessity condition for enabling high-resolution travel costs for all road network paths is that *every* edge in the road network

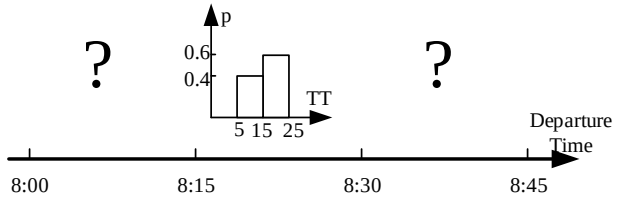
graph has a stochastic weight at each considered time interval such that it can capture uncertain traffic dynamics [2, 9]. To enable such time-dependent and stochastic weights on each edge, different kinds of traffic data, e.g., GPS data, loop detector data, and traffic camera data, can be collected [10, 11]. Yet, it is still an almost impossible task to cover all edges with stochastic weights at each time interval due to two reasons. First, GPS data is skewed [12, 13]. It is quite normal that there always are taxis on downtown roads, while there is barely any taxi appears in a given suburban area; Second, loop detectors are only deployed on some roads. Since a loop detector needs to be embedded into the road, it is expensive to install and maintain loop detectors, so these can only provide data for a small portion of edges [7]. Furthermore, if we split the available data temporally, e.g., split it into 15 minutes intervals, the data spareness problem becomes much more severe [14].



(a) UTS for e_1.



(b) UTS for e_2.



(c) UTS for e_4.

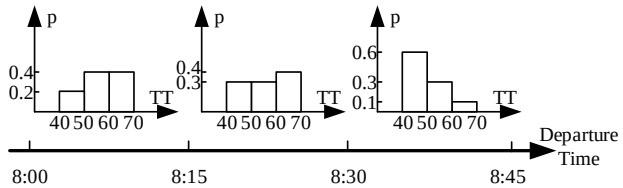
Fig. 1.2: Examples of Data Sparseness in UTS(p denotes probability and TT denotes the travel time in minutes).

Example 1.1.2 (Data Sparseness)
In Figure 1.2, several time intervals are marked with "?" because they do not have any observations. This illustrates the data sparseness problem. Consider the time interval 8:15–8:30. In this interval, the stochastic travel

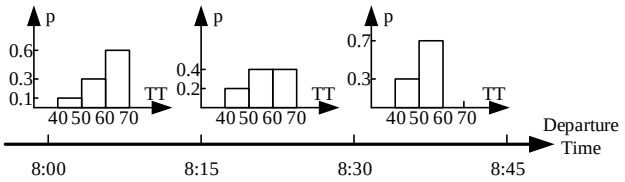
time (in minutes) for edge e_2 can be represented as {[40, 50): 0.2, [50, 60): 0.4, [60, 70]: 0.4}, which means that there is 0.2 probability that the travel time falls into [40, 50), 0.4 probability that it falls into [50, 60), and 0.4 probability that it falls into [60, 70], and the stochastic travel time for edge e_4 is {[5, 15): 0.4, [15, 25]: 0.6}. Yet, there is no stochastic weight for e_1 since no observations are available.

2) **Decision Making under Uncertainty.** When we have filled the gaps in the edge weights in a road network, we can construct an uncertain travel cost for any path. The question is how to select the optimal path under uncertainty, which is different from selecting the optimal path in a setting with deterministic travel costs.

Example 1.1.3 (Path Selections)
Figure 1.3 shows UTSs for paths P_1 and P_2 from A to E. Both UTSs have stochastic travel times for all time intervals. Example 1.1.1 describes the situation shown here for the time interval from 8:30 to 8:45, where the two paths do not dominate each other. So, what is the optimal choice for each of the two time intervals? The answer is P_1 because it stochastically dominates P_2. No matter what deadline is considered, P_1 always has a higher probability of arriving on time.



(a) UTS for p_1.



(b) UTS for p_2.

Fig. 1.3: Example of UTSs for Paths(p denotes probability and TT denotes the travel time in minutes).

3) **Spatio-Temporal Correlations.** Traffic data is spatio-temporally correlated. It is quite intuitive that if an accident happens on one edge, this may

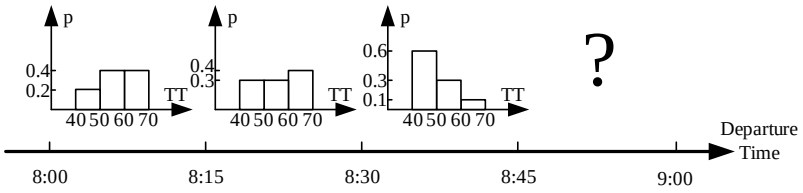


Fig. 1.4: Example of UTSs Forecasting of Path P_1(p denotes probability and TT denotes the travel time in minutes).

affect the future travel costs of the neighbor edges. Similarly, this correlation is also embedded in the traffic data from region to region. However, such spatio-temporal correlations are quite complex and are hard to capture in a well-defined model [15].

Example 1.1.4 (Spatio-Temporal Correlations)
Figure 1.4 shows an example of forecasting the stochastic travel time of path P_1 at 8:45-9:00. It is a safe bet that it is better to consider the historical observations on $e1$, e_2, and e_4 together than considering those of e_1 and e_4 independently, since their observations during 8:00–8:15, 8:15–8:30, and 8:30–8:45 are incomplete. We need to take into account spatio-temporal correlations to enable better forecasting.

1.2 Thesis Structure

The thesis adopts an UTS approach to analyzing big traffic data, the goal being of providing a foundation that can contribute to enabling high-resolution intelligent transportation applications, such as risk aware path selection, stochastic travel time prediction, and eco-routing. As already mentioned in the previous section, we aim to address three challenges in the thesis, whose overall structure is illustrated in Figure 1.5. First, we propose a generic data-driven deep learning stochastic weight completion framework to associate a stochastic weight with each road network edge, which addresses the data sparseness problem. Assuming that the costs of different candidate paths are modeled as UTSs, we conduct a stochastic optimal analysis over the UTSs to identify the optimal paths during different intervals. Finally, we construct an OD matrix to represent the stochastic travel costs, e.g., travel speed or travel flows, between any pair of regions, i.e., a cell in the OD matrix contains stochastic weights that capture travel from an origin region to a destination region. Then we solve the stochastic OD matrix forecasting problem to contend with spatio-temporal correlations.

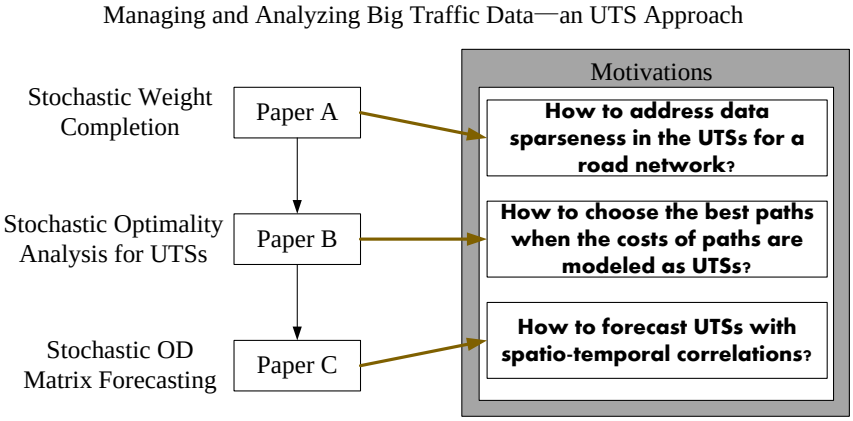


Fig. 1.5: Thesis Structure. Paper A proposes a stochastic weight completion framework to address the data sparseness problem in a road network. To obtain the optimal paths from a set of paths with different UTSs, Paper B proposes a risk-aware path selection method. Further, assuming that we use an OD matrix to store the UTSs of the optimal paths, we investigate the problem of stochastic OD matrix forecasting while taking spatio-temporal correlations into consideration. The recommended order to go through the thesis is *Paper A then Paper B then Paper C*.

Chapter 2

Stochastic Weight Completion

This chapter gives an overall introduction of Paper A [14]. The chapter reuses content from the paper when that was considered most effective.

2.1 Problem Motivation and Statement

We first consider the problem setting. Given a road network where a subset of edges have stochastic weights, our goal is to associate all edges with accurate stochastic weights. We call this the *stochastic weight completion* problem. The following example is reproduced from [14].

> **Example 2.1.1 (Example of Stochastic Weight Completion)**
> Continuing with Example 1.1.2, Figure 2.1 shows the stochastic weight completion process. To ease the process, the prior information is converted into a matrix representation where each row denotes the stochastic weight of an edge such that we get a stochastic weight matrix $W \in R^{6\times 3}$, whose first four rows are denoted as "?" due to data sparseness for $e_1, \cdots, e_4$ and the stochastic weights for e_5 and e_6 are displayed in the last two rows. We aim to estimate the stochastic weights for $e_1, \cdots, e_4$ such that we obtain a new matrix $\widehat{W}$ that contains meaningful stochastic weights for all the edges.

Existing studies have focused on the data sparseness problem in road networks and have converted it into a regression problem [7, 16, 17]. Those studies try to learn the discrepancies of the weights among adjacent edges

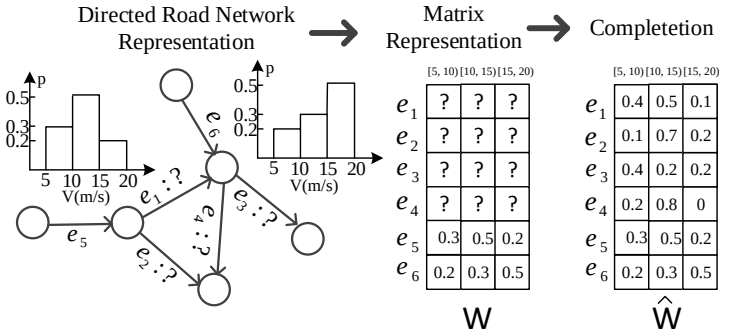


Fig. 2.1: Example of Stochastic Weight Completion [14]. © 2019 IEEE

by using regression, so that the weights of the edges covered by traffic data can be propagated to their adjacent edges that are not covered by traffic data. However, due to the data being sparse, the dependencies among edges cannot be captured accurately by only considering similarities of weights among adjacent edges. Moreover, these existing studies only consider deterministic weights, e.g., average travel speeds, and converting these into stochastic weights, e.g., travel speed distributions, is not a trivial task [14].

To cope with the above mentioned problems, in Paper A, we propose a data-driven framework to address the problem using graph convolutional neural networks (GCNN) [14]. Specifically, we first utilize a GCNN to encode the sparse stochastic weights embedded in a road network in a latent space using spectral graph theory [18]. Then we recover stochastic weights for all edges in this latent space. To satisfy the problem definition, we construct loss functions on the edges with data in the input only such that learning occurs in a unsupervised/semi-supervised manner [14]. When the training data is big enough, the relationships between latent feature and all edges can be learned such that it can be employed to accomplish the task of stochastic weight completion. Moreover, we can consider additional contextual information, e.g., peak/off-peak hour, weekdays, and weather, to improve the performance of this framework [14].

The contributions of this work can be characterized as follows.

- A stochastic weight completion problem is formalized;
- A data-driven framework is proposed to address the problem by using graph convolution neural networks;
- Additional contextual information is included into the framework to further improve accuracy;
- Extensive experiments with different data sources, specifically GPS data

and loop detector data, are conducted to offer insight into the framework and its effectiveness.

2.2 Stochastic Weight Completion

We first need to represent the relationships between edges in the content of graph, so we derive an edge graph from a road network. The following definitions and examples in this section are reproduced from [14].

Definition 2.2.1
A road network is a directed graph $H = (V, E)$, where $V = \{v_1, \cdots, v_{|V|}\}$ is a vertex set contains all road intersections in the road network; $E \subseteq V \times V$ is an edge set, where an edge e_k from vertex v_i to vertex v_j is represented as $e_k = (v_i, v_j)$, where $1 \leq i, j \leq |V|$ and $1 \leq k \leq |E|$.

Definition 2.2.2
An edge graph is an undirected graph $G = (E, A)$, where E is an edge set that has the same definition as above, and $A \in R^{|E|\times|E|}$ is an adjacent matrix that describes the connectivity between edges. Specifically, if e_i and e_j are connected—either from e_i to e_j or the other way around—then $A_{i,j} = 1$; otherwise, $A_{i,j} = 0$. Thus, A is a symmetric matrix, which also indicates that the edge graph is undirected.

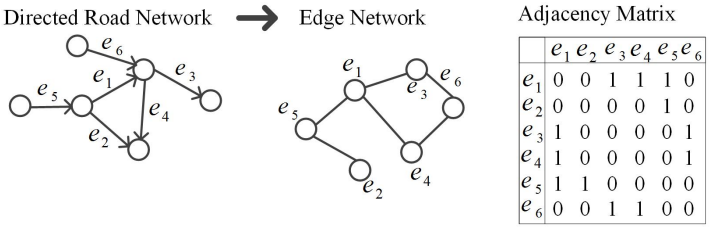


Fig. 2.2: Road Network and Its Edge Graph [14]. © 2019 IEEE

Example 2.2.1 (Constructing Edge Graph)
Figure 2.2 gives an example of how to construct an undirected edge graph and its corresponding adjacent matrix from a directed road network. For instance, after going through edge e_5, e_2 can be a consecutive edge meaning that e_2 and e_5 are connected in the edge graph [14], i.e., $A_{2,5} = A_{5,2} = 1$. However, e_1 and e_2 cannot be consecutive edges for each other, so $A_{2,1} = A_{1,2} = 0$.

To capture the temporal dynamics of traffic, we accumulate the data in every 15 minute interval as stochastic weights and obtain 96 intervals for a

day. Our end goal is to have one stochastic weight function as follows, which is reproduced from [14].

$$\mathcal{F} : E \times TI \rightarrow D,$$

where TI is set of time intervals that we are interested in, and D is the corresponding stochastic weights. Specifically, $\mathcal{F}(e_i, T_j)$ gives the stochastic weight for edge e_i in time interval T_j.

Due to data sparseness, given a time interval T_j, we first partition all edges into two subsets: E_c and E_m, which denote the edges covered by data and the edges not covered ("missing") by data, respectively. They are mutually exclusive and collectively exhaustive, i.e., $E_c \cup E_m = E$ and $E_c \cap E_m = \emptyset$ [14]. In the example in Figure 2.1, $E_c = \{e_5, e_6\}$ and $E_m = \{e_1, e_2, e_3, e_4\}$ at [8:15, 8:30], so $\mathcal{F}(e_5$, [8:15, 8:30]) and $\mathcal{F}(e_6$, [8:15, 8:30]) can be instantiated, while this is not possible for the edges in E_m [14]. Therefore, our main problem can be converted to that of instantiating $\mathcal{F}(e_m, T_j), \forall e_m \in E_m$ [14].

Definition 2.2.3
An equi-width histogram is a set, HIST $= \{(b_1, p_1), \cdots, (b_m, p_m)\}$, where pair (b_i, p_i) denotes a bucket and the corresponding probability, and m is the total number of buckets in the histogram. Bucket $b_i = [l_i, u_i)$ is a range, where l_i and u_i are the lower and upper bounds, respectively. If we fix the buckets, the histogram can be represented as a vector of size m: $[p_1, \cdots, p_m]$.

Example 2.2.2 (Construct Stochastic Weight)
In Figure 2.1, we can observe that the probability that the speed falls into [5, 10) is 0.3, [10, 15) is 0.5, and [15, 20] is 0.2 meaning that the speed histogram of edge e_5 is $\{([5, 10), 0.3), ([10, 15), 0.5), ([15, 20), 0.2)\}$. If we use the same buckets for all edges, the stochastic speed for e_5 can be given by [0.3, 0.5, 0.2].

Paper A uses equi-width histograms and same bucket size to represent the stochastic weights for all edges such that we can use a vector to represent the stochastic weight for an edge. Assume that the number of edges in the road network is n and number of bucket is m. Then we can use a matrix $W \in R^{n \times m}$ to represent the the stochastic weights of all edges in interval T_j, where each row vector $w_{i\cdot}$ corresponds to the stochastic weight for edge e_i [14]. If an edge $e_i \in E_m$ then $w_{i\cdot}$ is an empty vector [14]. Therefore, the first four rows in W in Figure 2.1 that are marked with "?" are empty vectors. Paper A uses the zero-vector, i.e., the vector where all values are zero, to capture missing edges.

Problem Formulation

The *Stochastic Weight Completion* problem can be formulated as follows, which is reproduced from [14]: given a time interval T_j and an incomplete stochastic weight matrix W whose rows have spatial correlations, we aim to produce a full stochastic weight matrix $\widehat{W}$ that contains meaningful stochastic weights.

2.3 Graph Convolutional Weight Completion

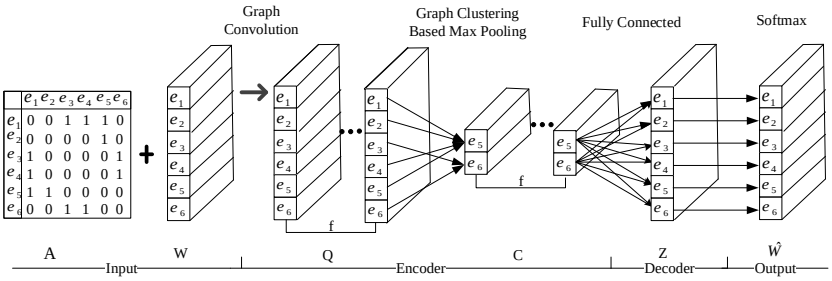


Fig. 2.3: System Architecture for GCWC [14]. © 2019 IEEE

As mentioned above, we construct an edge graph to describe the spatial correlations among edges. The intuition behind Paper A is that the stochastic weights on one edge can be propagated to its connected edges [7, 19] such that a set of latent space features $\mathbb{C} = \{C_i\}_{i=1}^{f}$ can be *encoded* from the incomplete stochastic matrix W [14]. We can then *decode* the latent features $\mathbb{C}$ to obtain a full stochastic weight matrix $\widehat{W}$ [14]. This process is called an *auto-encoder* [20, 21]. Using this approach, we construct a basic model called Graph Convolutional Weight Completion (GCWC) to solve the stochastic weight completion problem [14], whose system architecture is shown in Figure 2.3.

In this architecture, we take as input an incomplete stochastic matrix W and an adjacency matrix A that describes the spatial correlations among rows [14]. The input is processed by the GCNN to map useful features from edges with data to the edges with no data, through which we generate a total number of f feature maps. The feature maps are then encoded via graph clustering based max pooling to generate a set of encoded features $\mathbb{C}$. In the decoding process, we utilize a fully connected layer to determine the relationship between the encoded state $\mathbb{C}$ and the prior-output state Z. We call the latter a prior-output state because it does not meet the requirement of being a stochastic weight matrix: (1) for each $\widehat{w}_{i,j} \in \widehat{W}$, $0 \leq \widehat{w}_{i,j} \leq 1$, with $1 \leq i \leq n$ and $1 \leq j \leq m$; (2) $\sum_j \widehat{w}_{i,j} = 1$ [14]. Therefore, we apply a *softmax* function to Z such that we have an output $\widehat{W}$ that satisfies the above-mentioned property.

During training, we take matrix W as input and output label to make our framework perform unsupervised/semi-supervised learning. We define the loss function used in the framework as the average KL-divergence between $\widehat{W}$ and W on the edges with data [14]. Then, back-propagation is applied to learn the parameters in the framework.

2.4 Context Aware Graph Convolutional Weight Completion

To further improve the performance of the GCWC, we propose an advanced model called A-GCWC that takes into account contextual information [14]; the framework is shown in Figure 2.4.

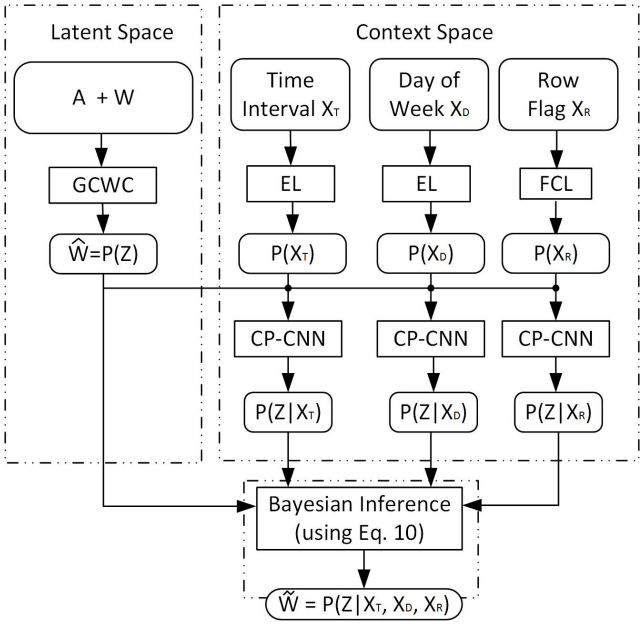


Fig. 2.4: Context Aware Graph Convolution Neural Network [14]. © 2019 IEEE

Since the stochastic weight matrix W is instantiated by giving a time interval T_j, we can derive two kinds of time-related contextual information: time of the day X_T and day of the week X_D [14]. In addition, we can use row flag X_R as another context to indicate whether a row in W is empty or not. The following example shows the construction of these contextual information which is reproduced from [14].

Example 2.4.1 (Construct Contextual Information)
If we select 15 minutes as the interval length, we get 96 intervals in one day, which can be represented as a one-hot vector with 96 bits, $X_T \in R^{96}$. Similarly, we can use another one-hot vector $X_D \in R^7$ with 7 bits to represent the day of the week. For example, we need to instantiate a stochastic weight for [0:15, 0:30), Thursday. Then, $X_T[2] = 1$ and $X_D[4] = 1$, and all other positions in the two vectors are zero. In the example in Figure 2.1, where only e_5 and e_6 are covered with data in W, we have $X_R = [0, 0, 0, 0, 1, 1]$.

In Figure 2.4, the A-GCWC is composed of three modules: a *Latent Space* module, a *Context Space* module, and a *Bayesian Inference* module. The *Latent Space* module is identical to what we already presented in the GCWC framework. The intuition here is to learn initial stochastic weight without contextual information [14]. In the *Context Space* module, we convert the one-hot vectors into probability vectors, e.g., $P(X_T)$, $P(X_D)$ and $P(X_R)$, via an embedding layer or a fully connected layer followed by a softmax layer, which can be regarded as the prior probability for the corresponding contextual information [14]. To ease the process, we make the dimensionality of the probability vectors the same as that of $P(Z)$, the output from the *Latent Space* module. Next, we process $P(Z)$ with $P(X_T)$ via a conditional probability CNN (CP-CNN) operation to generate the posterior probability of stochastic weight Z on the time of the day context, $P(Z|X_T)$ [14]. The same process is applied to $P(X_D)$ and $P(X_R)$ to obtain $P(Z|X_D)$ and $P(Z|X_R)$, respectively.

Finally, all the outputs from the *Latent Space* and *Context Space* modules are taken as input to the *Bayesian Inference* module [14]. The equation we adopt is displayed in Equation 2.1, whose mathematical derivation is given in Paper A; this equation corresponds to Equation 10 in Paper A [14] and Eq. 10 in Figure 2.4.

$$P(Z|X_T, X_D, X_R) = \frac{P(Z|X_T)P(Z|X_D)P(Z|X_R)}{[P(Z)]^2}, \tag{2.1}$$

According to Equation 2.1, we can obtain more accurate stochastic weights given all the types of context from the A-GCWC, denoted as $P(Z|X_T, X_D, X_R)$, which is different from $\widehat{W} = P(Z)$ returned from the GCWC.

2.5 Experimental Evaluation

To study the effectiveness of GCWC and A-GCWC, we use two real traffic data sets: *Highway Tollgate Network (HW)* and *City Road Network (CI)* [14]. In all the experiments, we use eight buckets for histograms, with each bucket

having length 5 m/s, thus covering the range [0, 40]. Then, we compare our proposed frameworks GCWC and A-GCWC with 6 baseline methods: 1) Historical Average (HA): we use all the historical travel speed records on that edge from the training data to construct its corresponding histogram; 2) Gaussian Process (GP): a Gaussian process regression model; 3) Random Forest (RF): a random forest regression model; 4) Latent Space Model (LSM) [7]: the state-of-the-art method to complete the deterministic missing weights in a road network; 5) Convolutional Neural Network (CNN): a classical convolutional neural network that replaces GCNN with CNN in GCWC; 6) Diffusion convolutional Recurrent Neural Network (DR) [15]: the state-of-the-art method in predicting deterministic edge weights in a network [14].

We first evaluate that accuracy of the results from GCWC and A-GCWC for stochastic weight estimation and prediction, respectively. To quantify the differences among different methods, we introduce two measurements: Mean Kullback-Leibler divergence Ratio (MKLR) and Fration of Likelihood Ratio (FLR) [14]. The detailed definitions for the two metrics are found in Paper A [14]. For MKLR, a lower value is prefered, and for FLR, we prefer a higher value. We show the results of estimation on CI data set for both metrics in Tables 2.1 and 2.2, respectively. We can observe that A-GCWC and GCWC outperform other baselines by a clear margin in all the settings with different removal ratios rm.

rm	GP	RF	LSM	CNN	DR	GCWC	A-GCWC
0.5	1.00	0.96	1.08	0.55	0.85	**0.48**	**0.48**
0.6	1.00	0.97	1.17	0.59	0.68	0.50	**0.49**
0.7	1.00	0.98	1.26	0.58	0.55	0.50	**0.49**
0.8	1.00	0.99	1.35	0.66	0.61	**0.49**	**0.49**

Table 2.1: MKLR for the CI Dataset, Estimation [14]. © 2019 IEEE

rm	GP	RF	LSM	CNN	DR	GCWC	A-GCWC
0.5	0.52	0.61	0.10	0.78	0.75	0.84	**0.85**
0.6	0.52	0.61	0.11	0.78	0.75	0.83	**0.84**
0.7	0.51	0.60	0.10	0.81	0.81	0.83	**0.84**
0.8	0.52	0.60	0.11	0.77	0.78	**0.85**	0.83

Table 2.2: FLR for the CI Dataset, Estimation [14]. © 2019 IEEE

We can change the input and output a little bit to adapt to the setting of prediction which is detailed introduced in Paper A. We show the MKLR results when performing prediction using the CI data set in Table 2.3. We observe that A-GCWC and GCWC are the best methods in this setting [14]. However, the performance of DR and CNN are much better than the estimation setting, which are close to those of our methods.

Finally, we modify our framework by replacing the two softmax functions with sigmoid functions to support the estimation of missing deterministic

rm	GP	RF	LSM	CNN	DR	GCWC	A-GCWC
0.5	1.09	0.97	4.18	0.50	0.48	0.45	**0.43**
0.6	1.12	0.97	4.15	0.50	0.49	**0.46**	**0.46**
0.7	1.16	0.98	4.16	0.54	0.54	0.49	**0.48**
0.8	1.24	0.98	4.30	0.59	0.53	0.50	**0.49**

Table 2.3: MKLR for the CI Dataset, Prediction [14]. © 2019 IEEE

values. The result when using the CI data set is shown in Table 2.4, where the mean average percentage error (MAPE) is used to quantify the accuracy—the lower the better [14]. We observe that A-GCWC performs overall the best across all removal ratios.

rm	LSM	CNN	DR	GCWC	A-GCWC
0.5	31.0%	10.9%	11.3%	11.6%	**10.8%**
0.6	37.3%	**11.2%**	12.5%	12.2%	**11.2%**
0.7	44.7%	11.5%	13.6%	12.2%	**11.4%**
0.8	52.1%	13.0%	**11.5%**	12.1%	**11.5%**

Table 2.4: MAPE for the CI Dataset, Average [14]. © 2019 IEEE

Chapter 3

Stochastic Optimality Analysis for UTS

This chapter gives an introduction to Paper B [23] and reuses content from the paper when that was found to be most effective.

3.1 Problem Motivation and Statement

To satisfy the increasing needs of making path selection decisions algorithmically and optimally, it is desirable to be able to take the time varying and uncertain nature of travel costs of candidate paths into account [10, 23]. The underlying applications include autonomous vehicles, mobility-on-demand, efficient logistics, etc.

Origin-destination (OD) matrices [22] are widely used in logistics and transportation companies. To support the use of OD matrices, we partition a road network in a city or country into N zones such that we obtain a matrix with dimensionality $N \times N$ in which an element (i, j) contains the "best" path from zone i to zone j [23].

As explained in the previous chapter, the sparseness of vehicle travel data in a road network can be addressed by the A-GCWC and GCWC frameworks. In particular, given an origin and a destination (i, j), it is possible to derive time-dependent, stochastic weights for several candidate paths from the origin to the destination. We need to find the best paths for each (i, j). It is desirable to model the travel costs of path as being time varying and uncertain [19, 23, 24], which is also the topic of this Ph.D. thesis—uncertain time series (UTS): 1) we select 15 minutes as the time interval length such that we have 96 intervals in one day; 2) the travel costs of a path within a time interval are collected to construct a travel cost distribution that captures the

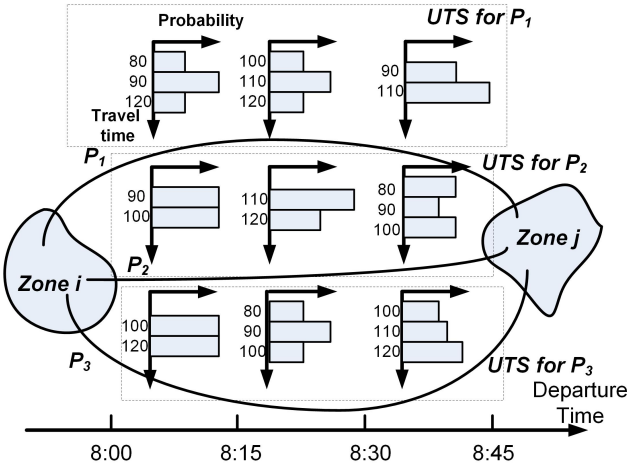


Fig. 3.1: Motivating Example [23]. © Springer 2018

uncertainty in the travel costs.

It is common that logistics companies need to arrange deliveries, and flex transportation companies receive travel requests at any time during a day. Therefore, identifying efficiently the paths with the best stochastic weights at any time during a day is important functionality. For example, find the "best" path among P_1, P_2, and P_3 at all the time intervals is illustrated in Figure 3.1. The following example is reproduced from [23].

Example 3.1.1 (Motivating Example)
Figure 3.1 shows an example of three candidate paths P_1, P_2, and P_3 that go from zone i to zone j. We can see that each path has different travel time histograms for different departure time periods, e.g., [8:00, 8:15), [8:15, 8:30), and [8:30, 8:45) [23]. Of course, the discrete distributions can be replaced with continuous distributions, e.g., Gaussian mixture models [24].

To obtain the "best" paths for different intervals, it is quite intuitive to compare the three paths at each time interval separately. Take time interval [8:00, 8:15) as an example. The stochastic travel times of P_1, P_2, and P_3 are listed in Table 3.1.

Example 3.1.2 (Stochastic Weights at [8:00, 8:15))
In Table 3.1, the distribution for path P_1 indicates that it gives 0.25 probability of arriving after 80 minutes of travel which is the shortest time among the three paths, a probability of 0.5 of arriving after 90 minutes, and 0.25

probability of arriving after 120 minutes, which yields a "wide" distribution. In contrast, the distribution for path P_2 is "narrow" and offers 0.5 probability of arriving after 90 minutes of travel and 0.5 probability of arriving after 100 minutes. Therefore, P_2 offers a more stable travel time but is slower than P_1. Path P_3 needs at least 100 minutes to arrive with 0.5 probability, and arrival after 120 minutes of travel has 0.5 probability.

Travel time (mins)	70	80	90	100	110	120
P_1	0	0.25	0.50	0	0	0.25
P_2	0	0	0.50	0.50	0	0
P_3	0	0	0	0.50	0	0.50

Table 3.1: Uncertain Travel Times, [8:00, 8:15) [23]. © Springer 2018

Different users may have different preferences: *risk-loving* users may prefer an early arrival with a risk of a late arrival, while *risk-averse* users may prefer a more predictable arrival time [23]. For example, in order to catch an early flight, it may be preferable to choose a so-called *risk-loving* path, while some users may choose a *risk-averse* path to be sure to catch the last flight of the day [23].

Moreover, emergency services such as ambulances may choose *risk-loving* paths [23, 25, 26], while *risk-averse* paths may be preferable when transporting perishable goods [27]. Example 3.1.3 shows an example of how to choose paths for different risk preference users, which is reproduced from [23].

Example 3.1.3 (Example of Path Selection)
In the example in Table 3.1, P_1 is for risk-loving users while P_2 is for risk-averse users. A *risk-neutral* user may choose either P_1 or P_2. Path P_3 is not interesting to any user, risk-loving, risk-averse, or risk-neutral, as it offers no benefits over paths P_1 and P_2.

We model the stochastic weights for a path P_i as an UTS that is reproduced from [23]:

$$T_i = \langle X_i^{(1)}, X_i^{(2)}, \cdots, X_i^{(N)} \rangle,$$

where $X_i^{(j)}$ is the stochastic weight of P_i in the j-th time interval, N is the length of, or total number of intervals in, T_i.

Then, UTSs from all candidate paths can be collected as $TS = \{T_1, T_2, \ldots, T_k\}$, where k is the number of candidate paths [23]. Next, a set $RVS^{(j)} = \{X_1^{(j)}, X_2^{(j)}, \ldots, X_k^{(j)}\}, 1 \leq j \leq N$ represents all paths' stochastic weights in

the j-th time interval [23]. Further, $O_x(RVS^{(j)})$ denotes the optimal choices among $RVS^{(j)}$ regarding to a risk preference x.

Example 3.1.4 (Example of Optimal Choice)
Figure 3.1 shows a set of UTS collections $TS = \{T_1, T_2, T_3\}$ for paths P_1, P_2, and P_3. Next, we select 15 minutes as the time interval length to get 3 time intervals from 8:00 to 8:45. Then $RVS^{(1)} = \{X_1^{(1)}, X_2^{(1)}, X_3^{(1)}\}$ is a set of stochastic weights for path P_1, P_2, and P_3 at the first time interval. According to Example 3.1.3, $O_l(RVS^{(j)}) = \{X_1^{(1)}\}$ for *risk-loving* users, $O_a(RVS^{(j)}) = \{X_2^{(1)}\}$ for *risk-averse* users, and $O_n(RVS^{(j)}) = \{X_1^{(1)}, X_2^{(1)}\}$ for *risk-neutral* users.

Paper B first provides a comprehensive analysis of stochastic dominance, user risk preferences, and utility functions [23]. Next, it proposes efficient methods for checking stochastic dominance between two random variables [23]. Further, paper B offers two random variable grouping methods to efficiently compute temporal dominance queries on multiple UTSs [23]. Finally, extensive experimental evaluations with one real world and one synthetic UTS collections are covered that suggest that effectiveness and efficiency of the proposed methods [23].

3.2 Stochastic Optimality

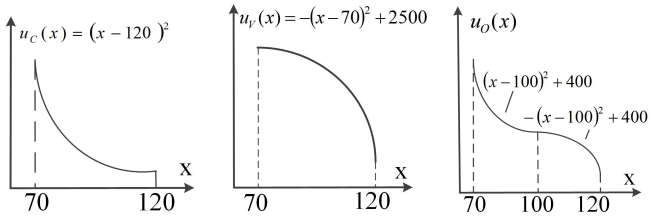
3.2.1 Decision Making under Uncertainty

We first introduce the notion of utility function that we use to capture a user's risk preferences. Since low travel times are preferred in our problem setting, a utility function should be *non-increasing*. We employ the *expected utility principle* [28, 29] to contend with the stochastic weights as follows, which is reproduced from [23]:

$$E_U(X_i) = \int_{X_i.min}^{X_i.max} u(x) \cdot f_{X_i}(x) dx, \tag{3.1}$$

where $u(x)$ is a utility function, $f_{X_i}(x)$ is a probability density function (continuous case) or probability mass function (discrete case) of stochastic weight X_i, and $X_i.min$ and $X_i.max$ are the minimum and maximum values in the range of X_i, respectively.

Here is a reproduced example from [23] of how to compare paths P_1, P_2, and P_3 from Table 3.1 by taking advantage of the utility function, $u(x) = 120 - x$ as follows.



(a) Convex (b) Concave (c) Other

Fig. 3.2: Categorization of Utility Functions [23]. © Springer 2018

Example 3.2.1 (Expected Utility of P_1, P_2, and P_3.)
According to Equation 3.1, the expected utility of P_1, P_2, and P_3 can be calculated as follows: $E_U(P_1) = 0.25 \cdot (120 - 80) + 0.5 \cdot (120 - 90) + 0.25 \cdot (120 - 120) = 25$, $E_U(P_2) = 25$, and $E_U(P_3) = 10$. Therefore, P_1 or P_2 are optimal since their expected utility values are the highest.

As is suggested in Section 3.1, the risk preferences of different users can be classified into three categories: *risk-loving*, *risk-averse*, and *risk-neutral* [23]. Next, we investigate the relationships between risk preferences and different categories of utility functions: convex utility functions are for risk-loving users, concave functions are for risk-averse users, and risk neutral users only care about the *non-increasing* property of utility functions, which we name *other* [23]. Three examples of the different categories of utility functions are shown in Figure 3.2.

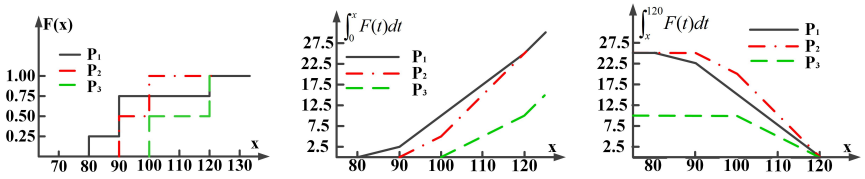
Example 3.2.2 (Expected Utility Values for Different Utility Functions)
To better exemplify the relationships between risk preferences and utility functions, we show the expected utility values of P_1, P_2, and P_3 for different utility functions (in Figure 3.2) in Table 3.2, where the optimal values are highlighted in bold.

Here, we observe that we have conclusions that align with example 3.1.3. P_1 has the largest expected utility value for $u_C(\cdot)$ and is the best option for risk-loving users, P_2 has the largest expected utility value for $u_V(\cdot)$ and is the most favorable option for *risk-averse* users. P_3 is not interesting to any user and does not have the best expected utility value for any of the three utility functions.

Further, we introduce a notion of stochastic dominance to compare stochastic weights. We introduce a stochastic dominance relationship for each of

	P_1	P_2	P_3	Optimal
u_C	**850**	650	200	P_1
u_V	1650	**1850**	800	P_2
u_O	**450**	**450**	200	P_1, P_2

Table 3.2: Expected Utilities for Paths P_1, P_2, and P_3 [23]. © Springer 2018



(a) cdf, $F_{X_i}(x)$ (b) Integral of cdf, $\widehat{F}_{X_i}(x)$ (c) Integral of cdf, $\widetilde{F}_{X_i}(x)$

Fig. 3.3: Distributions of P_1, P_2, and P_3 [23]. © Springer 2018

the three different categories of utility functions and associate their relationships with different risk preferences. Table 3.3 shows the relationships.

Risk Attitudes	**Utility Functions**	**Stochastic Dominance**
Risk-neutral	Non-increasing	First order
Risk-loving	Non-incr., Convex	Second convex order
Risk-averse	Non-incr., Concave	Second concave order

Table 3.3: Risk preferences, Utility Functions, and Stochastic Dominance [23]. © Springer 2018

To ease the understanding of the following definitions and theorems, we declare three probability functions from [23] and show the corresponding example paths P_1, P_2, and P_3 from Figure 3.3:
1) $F_X(x)$, the cumulative distribution function (cdf) of X;
2) $\widehat{F}_X(x) = \int_0^x F_X(t)dt$, the integral of cdf F_X from 0 to x;
3) $\widetilde{F}_X(x) = \int_x^{+\infty} F_X(t)dt$, the integral of cdf F_X from x to $+\infty$.

3.2.2 Stochastic Dominance

We define three kinds of stochastic dominance. The following definitions, theorems, and lemmas are reproduced from [23].

Definition 3.2.1
First Order Stochastic Dominance (FSD). Given two random variables X_1 and X_2, if $\forall a \in \mathbb{R}^+$, $F_{X_1}(a) \geqslant F_{X_2}(a)$, X_1 *first order stochastically dominates* X_2, denoted by $X_1 \succ_{fsd} X_2$.

Theorem 3.2.1
Given a non-increasing utility function $u(a)$ where $a \in R^+$, if $X_1 \succ_{fsd} X_2$ then the expected utility of X_1 is no smaller than that of X_2, i.e., $E_U(X_1) \geqslant E_U(X_2)$.

The proof of Theorem 3.2.1 is provided in paper B. The theorem demonstrates that the expected utility value of X_1 is always larger than or equal to that of X_2 as long as the utlity function is *non-increasing* [23]. We say that X_1 first order stochastic dominates X_2, i.e., $X_1 \succ_{fsd} X_2$. Recall the running example with paths P_1, P_2, and P_3 in Table 3.2. This theorem provides theoretical support for the finding that P_3 is of no interest because P_1 and P_2 first order stochastic dominate P_3 [23].

Definition 3.2.2
Second Convex Order Stochastic Dominance (SSD). Given random variables X_1 and X_2, if $\forall a \in \mathbb{R}^+$, $\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a)$, X_1 *second convex order stochastically dominates* X_2, denoted by $X_1 \succ_{ssd} X_2$.

Definition 3.2.3
Second Concave Order Stochastic Dominance (SCSD). Given two random variables X_1 and X_2, if $\forall a \in \mathbb{R}^+$, $\widetilde{F}_{X_1}(a) \geqslant \widetilde{F}_{X_2}(a)$, X_1 *second concave order stochastically dominates* X_2, denoted by $X_1 \succ_{scsd} X_2$.

Based on the definitions of SSD and SCSD, we are able to derive theorems that guarantee the following: (1) A user with a risk-loving utility function, i.e., a convex function, is not interested in choosing X_2 if $X_1 \succ_{ssd} X_2$, no matter the specific form of the convex function [23]. (2) A user with a risk-averse utility function, i.e., a concave function, is not interested in choosing X_2 if $X_1 \succ_{scsd} X_2$, no matter the specific form of the concave function [23]. In our example, P_1 is the optimal choice for risk-loving users, and P_2 is the optimal choice for risk-averse users [23].

3.2.3 Temporal Dominance Query

On the basis of the above-mentioned concepts, we formulate our problem as a *temporal dominance query* as follows, which is reproduced from [23]:

$$\mathcal{Q} : x \times R \times TS \rightarrow q,$$

where x is a stochastic dominance relationship, which can be mapped to a risk preference of a user, $R = [s, e]$ is the time interval from s to e, and TS is the set of UTS collections. Next, $q = \langle q^{(s)}, q^{(s+1)} \ldots, q^{(e)} \rangle$ is the output of the query, where $q^{(j)} = O_x(RVS^{(j)}), s \leqslant j \leqslant e$, which means that $q^{(j)}$ is a set of optimal (i.e., non-dominated) stochastic weights w.r.t. the given stochastic dominance relationship x among all stochastic weights in $RVS^{(j)}$ [23].

3.3 Checking Stochastic Dominance

We first propose efficient means of checking the three kinds of stochastic dominance between two random variables [23]. Then, we further improve the

performance of checking the stochastic dominance between two UTSs [23]. For brevity, we exemplify the core ideas of checking SSD for both cases. Paper B offers details on checking FSD and SCSD between two random variables [23]. The following lemmas are reproduced from [23] and we omit their proofs, which can be found in Paper B.

3.3.1 Checking SSD between Two Random Variables

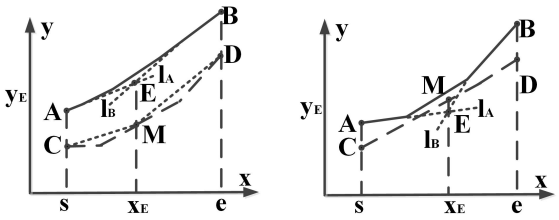
Lemma 3.3.1
If $X_1 \succ_{ssd} X_2$ then $X_1.min \leqslant X_2.min$ and $E(X_1) \leqslant E(X_2)$.

Lemma 3.3.1 offers an initial checking method by exploiting its contrapositive lemma:

Lemma 3.3.2
If $X_1.min > X_2.min$ or $E(X_1) > E(X_2)$ then $X_1 \not\succ_{ssd} X_2$.

Thus, X_1 cannot second convex order stochastic dominate X_2 if it meets one of the two conditions.

Next, we propose a speed-up algorithm to check the SSD relationship between two random variables , which follows the idea of *divide-and-conquer* [23].



(a) Case 1 (b) Case 2

Fig. 3.4: Speedup Algorithm for Checking SSD [23]. © Springer 2018

The speed-up algorithm first takes as input X_1, X_2, and a range $[s, e]$. Then, we graph $\widehat{F}_{X_1}(x)$ and $\widehat{F}_{X_2}(x)$ within $[s, e]$ in one figure. According to Definition 3.2.3, the curve of $\widehat{F}_{X_1}(x)$ should always be above $\widehat{F}_{X_2}(x)$ if X_1 is to dominate X_2 w.r.t. SSD. For ease of describing the intuition, we introduce four endpoints that correspond to $\widehat{F}_{X_1}(x)$ and $\widehat{F}_{X_2}(x)$, respectively: $A = (s, \widehat{F}_{X_1}(s))$, $B = (e, \widehat{F}_{X_1}(e))$, $C = (s, \widehat{F}_{X_2}(s))$ and $D = (e, \widehat{F}_{X_e}(e))$ [23]. Next, we construct a line l_A: $y = \widehat{F}'_{X_1}(s)(x - s)$ through point A and line l_B: $y = \widehat{F}'_{X_1}(e)(x - e)$ through point B. Thus, we obtain an intersection point $E = (x_E, y_E)$ of l_A and l_B. Further, we define a point $M = (x_E, \widehat{F}_{X_2}(x_E))$ on l_B. On the basis of these points, we derive the following lemma that is reproduced from [23].

Lemma 3.3.3
If $y_E \geqslant \widehat{F}_{X_2}(x_E)$ then $\widehat{F}_{X_1}(x) \geqslant \widehat{F}_{X_2}(x), \forall x \in [s, e]$.

Figure 3.4 gives an example of how to use this lemma with two cases:

Case 1: Figure 3.4(a) shows exactly the situation described in Lemma 3.3.3: *Point E is above M*, and thus we can safely say X_1 dominates X_2 w.r.t. SSD.

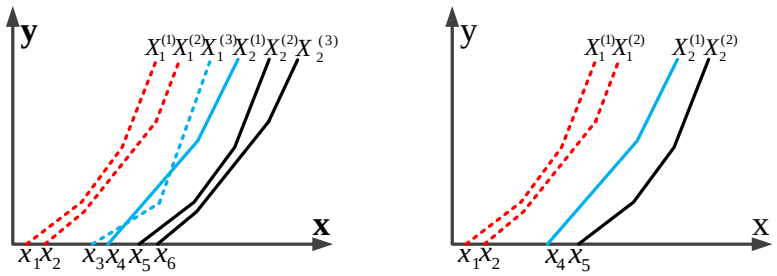
Case 2: Figure 3.4(b) exemplifies the other situation: *Point M is above E*. It does not satisfy Lemma 3.3.3's condition such that we cannot get a conclusion in $[s, e]$. Therefore, we divide $[s, e]$ by x_E into two sub-ranges: $[s, x_E]$ and $[x_E, e]$ [23]. Afterwards, we apply the the speed-up algorithm using $[s, x_E]$ and $[x_E, e]$ as range input.

3.3.2 Checking SSD between Two UTSs

The speed-up algorithm enables checking of SSD between two random variables. An additional step is needed to be able to check SSD between two UTSs $T_1 = \langle X_1^{(1)}, X_1^{(2)}, \cdots, X_1^{(N)} \rangle$ and $T_2 = \langle X_2^{(1)}, X_2^{(2)}, \cdots, X_2^{(N)} \rangle$ [23]. The most intuitive idea is to treat two UTSs as N separate pairs of random variables, $X_1^{(j)}$ and $X_2^{(j)}$, $1 \leq j \leq N$, and then apply the speed-up algorithm to each pair independently. However, such approach is not efficient, especially when N is large.

To better contend with the situation of large N, we propose the idea of grouping: we first split the N pairs of random variables into several groups based on some criteria, and then we compare the groups separately. Specifically, if the lower boundary curve of $\widehat{F}_X(x)$ for one group is always above the upper boundary curve of $\widehat{F}_X(x)$ for another group then all the random variables in former group dominate all the random variables in the latter group, and we do not need to determine the dominance relationships for multiple intervals separately, which improves the efficiency [23].

Example 3.3.1 (Intuition for Grouping.)
We have two UTSs with three time intervals: $T_1 = \langle X_1^{(1)}, X_1^{(2)}, X_1^{(3)} \rangle$ and $T_2 = \langle X_2^{(1)}, X_2^{(2)}, X_2^{(3)} \rangle$. Figure 3.5(a) shows an example of how each $\widehat{F}_{X_i}^{(j)}(x)$, $i = 1, 2$ and $j = 1, 2, 3$, looks. Intuitively, we can pick out $X_1^{(3)}$ and $X_2^{(3)}$ since $\widehat{F}_{X_1}^{(3)}(x)$ and $\widehat{F}_{X_2}^{(1)}(x)$ intersect. Then the remaining two pairs can be assigned to one group: $(X_1^{(1)}, X_2^{(1)})$ and $(X_1^{(2)}, X_2^{(2)})$, which is shown in Figure 3.5(b). In this group, we can clearly observe that the worse interval $X_1^{(2)}$ in T_1 (dashed and red color) dominates the better interval $X_2^{(1)}$ in T_2 (solid and blue color). Therefore, we can conclude that $X_1^{(1)}$ dominates $X_2^{(1)}$ and that $X_1^{(2)}$ dominates $X_2^{(2)}$ w.r.t. SSD, without checking them independently.



(a) Naive (b) Grouping

Fig. 3.5: Intuition for the Grouping Strategy [23]. © Springer 2018

Next, we design a general grouping framework that is capable of identifying groups with appropriate sizes. We then propose two techniques to check the dominance relationships between two groups—boundary checking and checking using lower and upper bounds. Paper B provides more details [23].

3.3.3 Checking SSD among multiple UTSs

Finally, we offer insight into the problem of checking dominance among a collection TS that have multiple UTSs, $|TS| > 2$. Two methods are proposed to address this problem: (1) Checking every pair using the already covered means of checking SSD between two UTSs; (2) Employing a merge-sort-like procedure to compare UTS pairs and use the intermediate comparison results to construct the final result [23]. The following example is reproduced from [23].

Example 3.3.2 (Checking SSD with multiple UTSs)
Considering $TS = \{T_1, T_2, T_3, T_4\}$, the first method checks the dominance relationships for every pair: (T_1, T_2), (T_1, T_3), (T_1, T_4), (T_2, T_3), (T_2, T_4), and (T_3, T_4). The second method first checks (T_1, T_2) and (T_3, T_4), whose results are utilized to construct the final result.

3.4 Experimental Evaluation

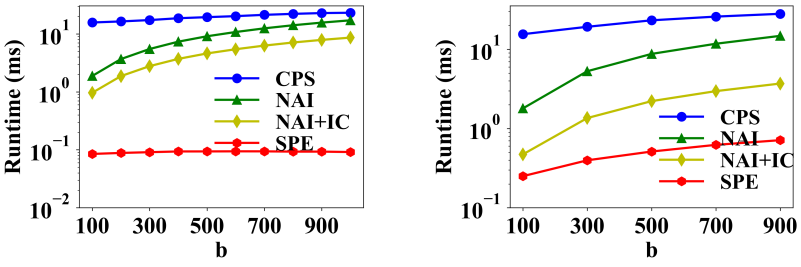
Setup: We consider two different collections of UTSs: real UTSs (RU) and synthetic UTSs (SU) [23]. For RU, we use a large GPS data set in Denmark from January 2007 to December 2008, which contains more than 180 million GPS records [23]. For SU, we construct UTSs from a publicly available deterministic time series data set[1].
Methods: To evaluate the efficiency of our methods on random variables, we consider the following 4 methods: (1) *NAI*: the naive algorithm that follows

[1] `academic.udayton.edu/kissock/http/Weather`

the definition only; (2) *NAI+IC*: the naive algorithm that also takes advantage of Lemma 3.3.1; (3) *SPE*: the speed-up algorithm; (4) *CPS*: a linear programming method using the CPLEX package[2] [23, 30].

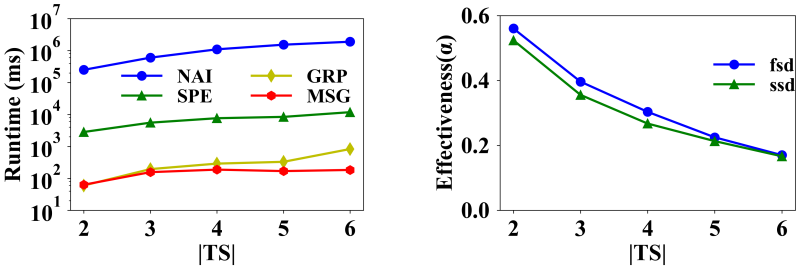
To evaluate our grouping methods, we consider the following four methods: (1) *NAI*: applies the naive method on each interval independently; (2) *SPE*: applies the speed-up method on each interval independently; (3) *GRP*: uses the grouping strategy but does not use the merge-sort-like procedure; (4) *MSG*: uses both the grouping strategy and the merge-sort-like procedure [23].

Results: First, we investigate the checking SSD between two random variables. For brevity, we only report results w.r.t. the number of histogram bins, b; more details can be found in paper B. Figure 3.6 shows that the proposed speed-up algorithm, *SPE*, has the lowest runtime. In addition, *SPE* also has stable performance and is insensitive to b [23].



(a) RU (b) SU

Fig. 3.6: Efficiency, Random Variables [23]. © Springer 2018



(a) Runtime of SSD (b) Effectiveness

Fig. 3.7: Efficiency and Effectiveness, UTS, RU [23]. © Springer 2018

Next, we evaluate the performance of checking SSD among multiple UTSs. Figure 3.7(a) shows the runtime of the different methods w.r.t. the cardinality of UTSs, $|TS|$, on RU [23]. *GRP* and *MSG* that adopt the grouping strategy are over one order of magnitude faster than *SPE* and over three orders of magnitude faster than *NAI*. These findings offer evidence of the efficiency of

[2]https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

our grouping strategy [23]. Further, *MSG* performs better than *GRP* when the cardinality increases, which indicates that the merge-sort-like procedure can further improve performance.

Finally, we use the ratio $\frac{Non-Dominated}{Total}$ to evaluate the effectiveness of the proposed temporal dominance queries, where *Non-Dominated* indicates the number of non-dominated random variables and *Total* indicates the total number of random variables [23]. Thus, the lower the better. Figure 3.7(b) shows that the ratios w.r.t. FSD and SSD go down when the cardinalities of the UTSs increase [23]. We can observe when the cardinality is larger than 2, more than half of the random variables can be pruned safely on average [23].

Chapter 4

Stochastic Origin-Destination Matrix Forecasting

This chapter gives an introduction to Paper C [31]. The chapter reuses content from the paper when that was found to be most effective.

4.1 Problem Motivation and Statement

In the previous chapter, we addressed the problem of finding stochastic optimal UTSs among several candidates from zone i to zone j. Here, we investigate a follow-up question: how to capture the temporal dynamics of the optimal UTSs from any zone i to any zone j in an OD-matrix. We call this problem *stochastic origin-destination matrix forecasting*: we need to forecast future OD-matrices on the basis of historical stochastic OD-matrices. The following example illustrates this problem and is reproduce from [31].

Example 4.1.1 (OD Matrix Forecasting)
Figure 4.1(a) shows an example of using three historical stochastic OD-matrices for intervals $T^{(t-2)}$, $T^{(t-1)}$, and $T^{(t)}$ to predict three future stochastic OD-matrices for intervals $T^{(t+1)}$, $T^{(t+2)}$, and $T^{(t+3)}$, where a three-dimensional tensor is employed to represent one stochastic OD-matrix.

Figure 4.1(b) shows an example of one stochastic OD-matrix being a $\mathcal{R}^{8\times8\times3}$ tensor: 8 origin regions occur in the first dimension, 8 destination regions occur in the second dimension, and 3 stochastic weights occur in the third dimension. In this example, the stochastic weights is constructed with three speed ranges (km/h): [10, 20), [20, 30), and [30, 40]. Therefore, the stochastic speed from region 7 to 8 can be represented as a vector [0.3, 0.5, 0.2] that represents the travel speed histogram $\{([10, 20), 0.3)$,

$([20, 30), 0.5), ([30, 40], 0.2)\}$.

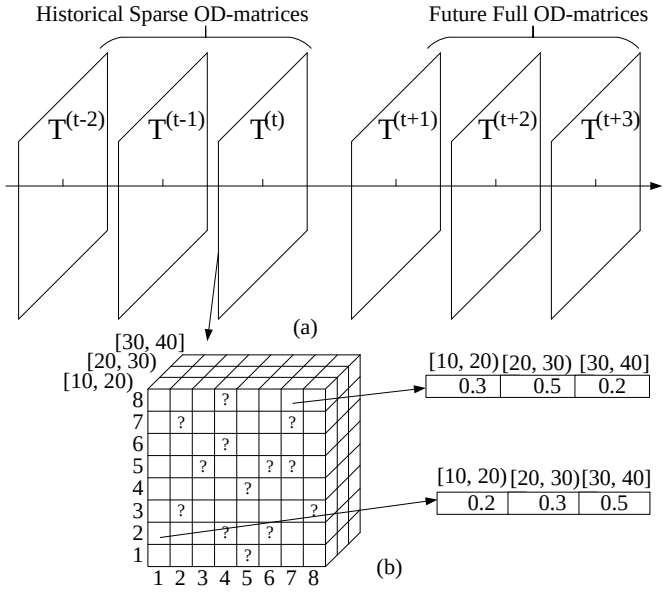


Fig. 4.1: Stochastic Origin-Destination Matrix Forecasting [31].

Therefore, our problem can be stated as follows. Given s sparse OD stochastic speed tensors $\mathbf{M}^{(t-s+1)}, \ldots, \mathbf{M}^{(t)}$ during s historical time intervals $T^{(t-s+1)}, \ldots, T^{(t)}$, we aim to predict the stochastic speeds for the next h time intervals $T^{(t+1)}, \ldots, T^{(t+h)}$ in the form of h full OD stochastic speed tensors $\mathbf{M}^{(t+1)}, \ldots, \mathbf{M}^{(t+h)}$ by learning the following function f [31].

$$f : [\mathbf{M}^{(t-s+1)}, \ldots, \mathbf{M}^{(t)}] \rightarrow [\mathbf{M}^{(t+1)}, \ldots, \mathbf{M}^{(t+h)}]$$

It is not trivial to solve this problem, due to the following two reasons.

(1) **Data Sparseness.** We need substantial trajectory data for each region pair during a time interval to construct a stochastic OD-matrix [31]. As mentioned in Section 1.1, however, data sparseness is prevalent in traffic data since GPS data is skewed and loop detectors are deployed only on some roads. Given even massive trajectory data, as this data is invariably spatially and temporally skewed [1, 5, 6, 13, 32], it is an almost impossible task to collect enough data to cover all region pairs for all intervals [31]. Here is a reproduced example from [31] as follows.

Example 4.1.2 (Sparseness Problem in NYC data.)
New York City taxi data set[1] that we use in paper C contains more than 14 million trips occurred on Manhattan during November and December

2013. Yet, this large trip set is unable to cover all "taxizone" pairs that are obtained by splitting Manhattan into 67 subregions, resulting in a total of $67 \times 67 = 4489$ pairs. Specifically, only 65% of these pairs can be covered with data. If we distribute this data to different time intervals, e.g., 15 minute interval, the data spareness will be far more severe.

Figure 4.1(b) shows an example of data sparseness, with missing elements marked with "?". Our task is to forecast future full OD-matrices by taking advantage of historical sparse OD-matrices [31].
(2) **Spatio-temporal Correlations.** It is easy to understand that traffic data is spatio-temporally correlated [31]. For example, if one region has congestion during a time interval then regions next to it have a high probability of being congested in the previous and subsequent time intervals. So we need to take both spatial and temporal correlations into consideration to achieve accurate stochastic OD forecasting. Figure 4.2 shows two ways, grid-based partitioning and road-based partitioning, of partitioning a region into sub-regions. Regardless of which partitioning method is utilized, we cannot guarantee that the region identifiers of two adjacent regions are consecutive, e.g., regions 1 and 4 are adjacent in Figure 4.2(a), and regions 4 and 7 are adjacent in Figure 4.2(b) [31]. Therefore, we need to invent a mechanism that allows spatial correlations to be taken into account for both the origin and the destination dimensions in the OD matrix. Furthermore, we need to capture the temporal dynamics to achieve better forecasting.

(a) Grid-based Partition (b) Road-based Partition

Fig. 4.2: Partition a City into Regions [31].

To forecast future full OD-matrices, in paper C, we propose a data-driven, end-to-end deep learning framework that addresses the problems of data sparseness and spatio-temporal correlations [31].

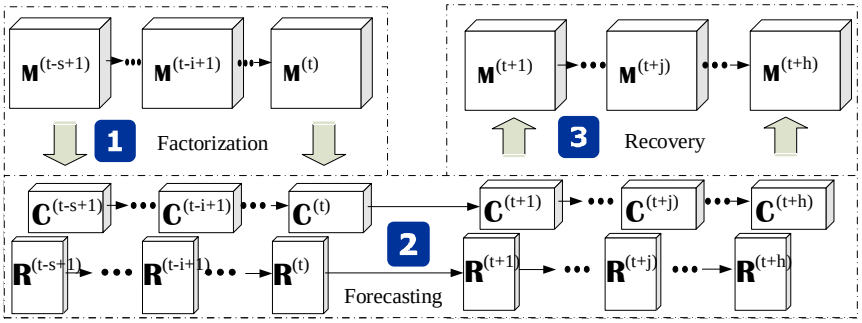


Fig. 4.3: Framework Overview [31].

4.2 Stochastic Speed Forecasting

The basic framework for stochastic origin-destination matrix forecasting is shown in Figure 4.3, which is composed of three steps: *Factorization*, *Forecasting*, and *Recovery*. We take as input s sparse OD matrices $\mathbf{M}^{(t-s+1)}, \ldots, \mathbf{M}^{(t)}$ that are instantiated from historical time intervals $T^{(t-s+1)}, \ldots, T^{(t)}$ [31]. For each input tensor $\mathbf{M}^{(t-i+1)}$, we first utilize a fully connected layer to *factorize* it into two smaller tensors $\mathbf{R}^{(t-i+1)} \in \mathbb{R}^{N \times \beta \times K}$ and $\mathbf{C}^{(t-i+1)} \in \mathbb{R}^{\beta \times N' \times K}$, respectively, where $1 \leq i \leq s$ and $\beta \ll N, N'$ [31]. Next, the resulting smaller tensors are fed into two gated recurrent unit neural networks (GRUs) to *forecast* corresponding tensors $\widehat{\mathbf{R}}^{(t+j)} \in \mathbb{R}^{N \times \beta \times K}$ and $\widehat{\mathbf{C}}^{(t+j)} \in \mathbb{R}^{\beta \times N' \times K}$, $1 \leq j \leq h$ [31]. Then, we employ matrix multiplication to *recover* $\mathbf{M}^{(t+j)} = \widehat{\mathbf{R}}^{(t+j)} \times \widehat{\mathbf{C}}^{(t+j)}$. To make each cell $\mathbf{M}^{(t+j)}_{o,d,:} \in \mathbb{R}^{1 \times K}$ meaningful, it should meet two requirements: 1) $\mathbf{M}^{(t+j)}_{o,d,k} \in [0,1]$, $\forall k \in [1,K]$; 2) $\sum_{k=1}^{K} \mathbf{M}^{(t+j)}_{o,d,k} = 1$, we apply a softmax function to $\mathbf{M}^{(t+j)}_{o,d,:}$, defined as follows, which is reproduced from [31].

$$\widehat{\mathbf{M}}^{(t+j)}_{o,d,:} = softmax(\mathbf{M}^{(t+j)}_{o,d,:}), \forall o \in [1,N], \forall d \in [1,N'], \forall j \in [1,h]. \tag{4.1}$$

The loss function $\ell(\cdot)$ is defined as the sum of the errors between the recovered future tensor and the ground-truth future tensor plus the regularization errors of the predicted factors $\widehat{\mathbf{R}}^{(t+j)}$ and $\widehat{\mathbf{C}}^{(t+j)}$ as follows, which is reproduced from [31].

$$\begin{aligned} \ell(\boldsymbol{F}, \boldsymbol{b}) = \sum_{j=1}^{h} [\lambda ||\widehat{\mathbf{R}}^{(t+j)}||_F^2 + \lambda ||\widehat{\mathbf{C}}^{(t+j)}||_F^2 + \\ ||\Omega^{(t+j)} \circ (\mathbf{M}^{(t+j)} - \widehat{\mathbf{M}}^{(t+j)})||_F^2], \end{aligned} \tag{4.2}$$

where $\boldsymbol{F}$ and $\boldsymbol{b}$ are training parameters in the framework, λ is a regularization parameter, $\Omega^{(t+j)} \in \mathbb{R}^{N \times N' \times K}$ is an indication tensor, where $\Omega^{(t+j)}_{o,d,k} = 1$ if $\mathbf{M}^{(t+j)}_{o,d,:}$ is not empty, which means it contains the ground truth [31]. Further, $\circ$ denotes element-wise multiplication, and $||\cdot||_F$ is the Frobenius-norm.

4.3 Forecast with Spatial Dependency

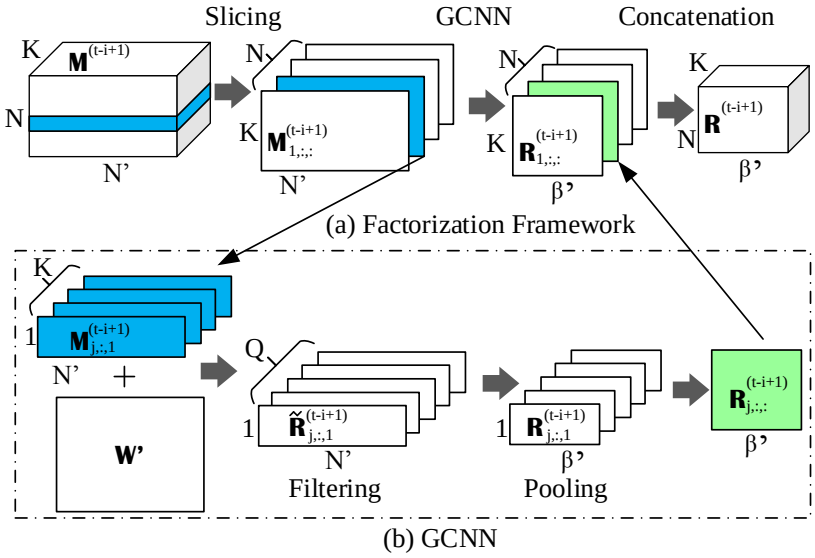


Fig. 4.4: Spatial Factorization for $\boldsymbol{R}$ [31].

We proceed to explain how to account for spatio-temporal correlations in the framework.

4.3.1 Spatial Correlation

To capture the spatial correlations among origin regions and destination regions, we employ the notion of a proximity matrix [15, 31]. For brevity, we only demonstrate this idea on the origin regions; the same applies to the destination regions.

Since we have N origin regions, we can construct an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ to capture connectivity, i.e., $A_{u,v} = 1$ if origin region i and j share a boundary; otherwise, $A_{u,v} = 0$ [31]. Next, we can derive a weighted proximity matrix $\boldsymbol{W}^{(\alpha,\sigma)} \in \mathbb{R}^{N \times N}$, where α is a parameter that captures adjacency hops and σ denotes standard deviation. In particular, $\boldsymbol{W}^{(\alpha,\sigma)}_{u,v} = e^{-x^2/\sigma^2}$ if region u can reach v in α hops, where x is the centroid distance between

u and v; otherwise, $\boldsymbol{W}_{u,v}^{(\alpha,\sigma)} = 0$. It is obvious that $\boldsymbol{W}^{(\alpha,\sigma)}$ is symmetric and non-negative [31].

4.3.2 Spatial Factorization

After obtaining the weighted proximity matrix, we perform factorization via the GCNN such that spatial correlation can be taken into consideration. Figure 4.4 shows an example of the operations for obtaining row factorization $\mathbf{R}^{(t-i+1)}$ from $\mathbf{M}^{(t-i+1)}$ [31]. In Figure 4.4(a), we first slice $\mathbf{M}^{(t-i+1)} \in \mathbb{R}^{N\times N'\times K}$ into N matrices along the origin dimension, i.e., $slice(\mathbf{M}^{(t-i+1)}) = [\mathbf{M}_{1,:,:}^{(t-i+1)}, \cdots, \mathbf{M}_{N,:,:}^{(t-i+1)}]$ [31]. Next, we apply the same GCNNs to each sliced matrix to obtain the result $[\mathbf{R}_{1,:,:}^{(t-i+1)}, \cdots, \mathbf{R}_{N,:,:}^{(t-i+1)}]$ [31]. Then, we obtain $\mathbf{R}^{(t-i+1)} \in \mathbb{R}^{N\times\beta'\times K}$ through concatenation. Figure 4.4(b) exemplifies the GCNNs operation on a sliced matrix $\mathbf{M}_{j,:,:}^{(t-i+1)}$, $j \in [1, N]$, which transforms $\mathbf{M}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{K\times N'}$ into $\mathbf{R}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{K\times\beta'}$ via *Filtering* and *Pooling* [31].

4.3.3 Spatial Forecasting

To better contend with spatio-temporal correlations, we construct a spatial forecasting that employs three GCNNs within each GRU, yielding CNRNNs. Since the signals encoded in $\mathbf{R}^{(t)}$ and $\mathbf{C}^{(t)}$ are different, we utilize different CNRNNs to process each factorization. For example, the processing of $\mathbf{R}^{(t)}$ at time interval $T^{(t)}$ can be formulated as follows, which are reproduced from [31].

$$\mathbf{S}^{(t+1)} = \sigma(\mathbf{G}_{\mathbf{S}} \otimes [\mathbf{H}^{(t)} : \mathbf{R}^{(t)}] + \boldsymbol{b}_{\mathbf{S}}) \tag{4.3}$$

$$\mathbf{U}^{(t+1)} = \sigma(\mathbf{G}_{\mathbf{U}} \otimes [\mathbf{H}^{(t)} : \mathbf{R}^{(t)}] + \boldsymbol{b}_{\mathbf{U}}) \tag{4.4}$$

$$\mathbf{H}^{(t+1)} = \tanh(\mathbf{G}_{\mathbf{H}} \otimes [\mathbf{R}^{(t)} : (\mathbf{S}^{(t+1)} \circ \mathbf{H}^{(t)})] + \boldsymbol{b}_{\mathbf{H}}) \tag{4.5}$$

$$\widehat{\mathbf{R}}^{(t+1)} = \mathbf{U}^{(t+1)} \circ \mathbf{R}^{(t)} + (1 - \mathbf{U}^{(t+1)}) \circ \mathbf{H}^{(t+1)}, \tag{4.6}$$

where $\mathbf{G}_{\mathbf{S}}$, $\mathbf{G}_{\mathbf{U}}$, and $\mathbf{G}_{\mathbf{H}}$ are graph convolution filters; $\mathbf{R}^{(t)}$ and $\widehat{\mathbf{R}}^{(t+1)}$ are the input and output of a CNRNN cell at time interval $T^{(t)}$, respectively; $\mathbf{S}^{(t)}$ and $\mathbf{U}^{(t)}$ are reset and update gates, respectively; symbol $\otimes$ denotes graph convolution whose definition can be found to be Definition 8 in [31]; symbol $\circ$ denotes the Hadamard product between two tensors; $\sigma(\cdot)$ is a *sigmoid* function and $\tanh(\cdot)$ is a *tanh* function [31].

Similarly, we apply the other CNRNN to $\mathbf{C}^{(t)}$. Therefore, we have the following spatial forecasting results: $[\widehat{\mathbf{R}}^{(t+1)}, \ldots, \widehat{\mathbf{R}}^{(t+h)}]$ and $[\widehat{\mathbf{C}}^{(t+1)}, \ldots, \widehat{\mathbf{C}}^{(t+h)}]$, which have the same dimensionality as introduced in previous section [31].

Finally, we use the same recovery operation as in previous section and obtain h future full OD stochastic speed tensors: $\widehat{\mathbf{M}}^{(t+1)}, \ldots, \widehat{\mathbf{M}}^{(t+h)}$ [31].

4.3.4 Loss Function

Similar to Equation 4.2, we define the loss function as follows, which is reproduced from [31].

$$\ell(\mathbf{G}, \boldsymbol{b}) = \sum_{i=1}^{h} [\lambda ||\widehat{\mathbf{R}}^{(t+j)}||_{W}^{2} + \lambda ||\widehat{\mathbf{C}}^{(t+j)}||_{W'}^{2} + ||\Omega^{(t+j)} \circ (\mathbf{M}^{(t+j)} - \widehat{\mathbf{M}}^{(t+j)})||_{F}^{2}], \tag{4.7}$$

where $\mathbf{G}$ and $\boldsymbol{b}$ denote the GCNN weights parameters and $||\cdot||_{W}^{2}$ is the Dirichlet norm under proximity matrix $\boldsymbol{W}$. Other symbols are explained in the context of Equation 4.2.

4.4 Experimental Evaluation

Setup: We consider two taxi trip data sets to study the proposed framework: the New York City Data Set (NYC) and the Chengdu Data Set (CD). NYC contains 14 million taxi trips collected from 2013-11-01 to 2013-12-31 in Manhattan, New York City [31]. CD contains 1.4 billion GPS records from 14,864 taxis collected from 2014-08-03 to 2014-08-30 in Chengdu, China[2] [31].
Methods: We evaluate the effectiveness of our proposed basis framework (BF) and advanced framework (AF), while considering also five baselines: (1) Naive Histograms (NH): we construct a histogram for each OD pair using all the travel speed records on the corresponding OD pair in the training set, which is then utilized to be the future forecasting results [31]. Next, we consider three time series forecasting methods: (2) Support Vector Regression (SVR) [33], (3) Vector Auto-regression (VAR) [34], and (4) Gaussian Process regression (GP) [35], (5) Fully Connected (FC): we apply a fully connected layer directly to obtain a single dense tensor to replace the factorization step in BF [31].
Results: Table 4.1 shows forecasting accuracy results when varying h for $s = 3$. To quantify the differences among different methods, we employ three measurements: Kullback-Leibler divergence (KL), Jensen-Shannon divergence (JS) and earth mover's distance (EMD). The detailed definitions of these are given in Paper C [31], and lower values are preferred for all three.

We first observe that the deep learning based methods FC, BF, and AF have better performance than the other baselines in most cases [31]. Next, BF

[2] `https://goo.gl/3VsEym`

Data	Metric	h	NH	SVR	VAR	GP	FC	BF	AF
NYC	KL	1	0.592	0.704	0.554	0.522	0.446	0.427	**0.311**
		2	0.592	0.713	0.562	0.535	0.438	0.417	**0.313**
		3	0.592	0.720	0.577	0.545	0.438	0.415	**0.314**
	JS	1	0.439	0.530	0.440	0.391	0.332	0.322	**0.299**
		2	0.439	0.537	0.452	0.400	0.327	0.317	**0.302**
		3	0.439	0.543	0.471	0.408	0.327	0.316	**0.305**
	EMD	1	0.293	0.452	0.305	0.266	0.250	0.246	**0.214**
		2	0.293	0.455	0.313	0.270	0.247	0.243	**0.216**
		3	0.293	0.458	0.317	0.274	0.247	0.243	**0.217**
CD	KL	1	0.697	0.818	0.699	0.674	0.694	0.582	**0.549**
		2	0.709	0.836	0.715	0.687	0.689	0.586	**0.555**
		3	0.700	0.822	0.780	0.684	0.807	0.792	**0.626**
	JS	1	0.580	0.672	0.814	0.597	0.517	0.438	**0.435**
		2	0.584	0.677	0.869	0.599	0.513	0.442	**0.444**
		3	0.592	0.692	0.875	0.619	0.590	0.571	**0.500**
	EMD	1	0.441	0.543	0.799	0.439	0.360	0.307	**0.289**
		2	0.443	0.539	0.871	0.434	0.361	0.313	**0.295**
		3	0.464	0.574	0.787	0.471	0.423	0.378	**0.311**

Table 4.1: Forecast Accuracy with Varying h, $s = 3$ [31].

beats the other baselines in most settings, which means that using factorization and forecasting are effective for OD matrix forecasting when having to contend with data sparseness [31]. Further, AF improves performance and outperforms the other methods significantly, which offers evidence of the effectiveness of our proposed dual-stage graph convolutional, recurrent neural network capturing spatio-temporal correlations and improving forecasting accuracy. Moreover, we also observe that the results on NYC are better than those on CD since the regions in Manhattan are more homogeneous than those in CD that has diverse regions that make forecasting much more complex and challenging [31]. Finally, we notice that the accuracy of AF becomes worse when increasing h. This is reasonable in that it is harder to forecast accurately in the far future.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis offers an uncertain time series (UTS) approach to analyzing big traffic data, which addresses the data sparseness problem in road networks, enables risk-aware path selection with UTSs, and enables forecasting in the context of UTSs with spatio-temporal correlations. Each problem is formulated and addressed in one paper, and three papers are included in the thesis. A summary of each paper is given below.

- Paper A [14] defines and studies the problem of stochastic weight completion such that each edge in a road network is assigned a UTS to describe its time varying stochastic travel cost weights. It first proposes a basic graph convolutional weight completion (GCWC) framework to assign stochastic weights to edges with missing weights. Then the framework is extended to take into account contextual information, yielding context aware graph convolutional weight completion (A-GCWC). Extensive experiments with two real data sets, highway loop detector data and city GPS taxi data, are conducted to demonstrate that the proposed models are able to outperform baselines and the state-of-the-art methods in all considered settings.
- Paper B [23] studies the retrieval of stochastically optimal paths in a UTS setting. The paper first conducts extensive investigations of the relationships among stochastic dominance, risk preferences, and utility functions. Next, it proposes speedup algorithms to check three different kinds of stochastic dominance relationships, each of which corresponds to a particular kind of risk preference. Then, a grouping strategy is put forward to improve the performance on two UTSs for different kinds

of stochastic dominance. Finally, it employs a merge-sort-like procedure to further improve the performance of finding optimal paths from multiple UTSs. Extensive experiments are performed to verify the efficiency of checking stochastic dominance between two uncertain objects, two UTSs, and multiple UTSs, respectively. The results show that the efficiency of the proposed algorithms is better than that of baselines in all considered settings.

- Paper C [31] formulates the problem of instantiating future full OD matrices from historical sparse OD matrices. To address the data sparseness problem while taking into consideration of spatio-temporal correlations, a data-driven, end-to-end deep learning framework is proposed. The paper first proposes a basic framework that is composed of three steps: factorization, forecasting, and recovery. In this framework, spatial correlations during factorization and forecasting cannot be captured well. Then, the framework is extended to empoly graph convolution on factorization and forecasting to contend better with spatial correlations and improve performance. Finally, the paper reports on extensive experiments with two real data sets, NYC and CD, to offer insight its performance , showing that its performance excels over different baselines and the state-of-the-art methods in all considered settings.

5.2 Future Work

In future work, it is of interest to extend the frameworks proposed in this thesis to support continuous distribution models, especially Gaussian mixture models. Next, it is also of interest to explore distributed and parallel computing frameworks [36, 37]. Then, these frameworks can be utilized by current real time routing systems while providing high-resolution travel cost information. Further, these frameworks can be integrated with reinforcement learning, the goal being to make autonomous vehicles more intelligent [38].

References

[1] I. Jindal, X. Chen, M. Nokleby, J. Ye *et al.*, "A unified neural network approach for estimating travel time and distance for a taxi trip," *arXiv preprint arXiv:1710.04350*, 2017.

[2] B. Yang, J. Dai, C. Guo, C. S. Jensen, and J. Hu, "PACE: a path-centric paradigm for stochastic path finding," *The VLDB Journal–The International Journal on Very Large Data Bases*, vol. 27, no. 2, pp. 153–178, 2018.

[3] N. Künzli, R. Kaiser, S. Medina, M. Studnicka, O. Chanel, P. Filliger, M. Herry, F. Horak Jr, V. Puybonnieux-Texier, P. Quénel *et al.*, "Public-health impact of outdoor and traffic-related air pollution: a european assessment," *The Lancet*, vol. 356, no. 9232, pp. 795–801, 2000.

[4] Texa A&M Transportation Institute, "2015 urban mobility scorecard and appendices," *https://mobility.tamu.edu/ums/congestion-data/other-urban-areas/*, 2015.

[5] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proceedings of the 20th ACM International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 25–34.

[6] H. Wang, Y.-H. Kuo, D. Kifer, and Z. Li, "A simple baseline for travel time estimation using large-scale trip data," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2016, pp. 61:1–61:4.

[7] D. Deng, C. Shahabi, U. Demiryurek, L. Zhu, R. Yu, and Y. Liu, "Latent space model for road networks to predict time-varying traffic," in *Proceedings of the 22th ACM International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1525–1534.

[8] J. Chen, "Modeling route choice behavior using smartphone data," Ph.D. Thesis, 2013, Ecole Polytechnique Fédérale de Lausanne, Switzerland.

[9] J. Hu, B. Yang, C. S. Jensen, and Y. Ma, "Enabling time-dependent uncertain eco-weights for road networks," *GeoInformatica*, vol. 21, no. 1, pp. 57–88, 2017.

[10] C. Guo, C. S. Jensen, and B. Yang, "Towards total traffic awareness," *SIGMOD Record*, vol. 43, no. 3, pp. 18–23, 2014.

[11] Z. Ding, B. Yang, Y. Chi, and L. Guo, "Enabling smart transportation systems: A parallel spatio-temporal database approach," *IEEE Trans. Computers*, vol. 65, no. 5, pp. 1377–1391, 2016.

[12] J. Dai, B. Yang, C. Guo, C. S. Jensen, and J. Hu, "Path cost distribution estimation using trajectory data," *PVLDB*, vol. 10, no. 3, pp. 85–96, 2016.

[13] C. Guo, B. Yang, J. Hu, and C. S. Jensen, "Learning to route with sparse trajectory sets," in *Proceedings of the 34th IEEE International Conference of Data Engineering*, 2018, pp. 1073–1084.

[14] J. Hu, C. Guo, B. Yang, and C. S. Jensen, "Stochastic weight completion for road networks using graph convolutional networks," in *Proceedings of the 35th IEEE International Conference of Data Engineering*, 2019, 12 pages (to appear).

[15] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proceedings of the 6th International Conference on Learning Representations*, 2018, *https://openreview.net/forum?id=SJiHXGWAZ*.

[16] T. Idé and M. Sugiyama, "Trajectory regression on road networks," in *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, 2011, pp. 203–208.

[17] J. Zheng and L. M. Ni, "Time-dependent trajectory regression on road networks via multi-task learning," in *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, 2013, pp. 1048–1055.

[18] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," *arXiv:1606.09375 [cs, stat]*, 2016.

[19] B. Yang, M. Kaul, and C. S. Jensen, "Using incomplete information for complete weight annotation of road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1267–1279, 2014.

[20] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Unsupervised and Transfer Learning - Workshop held at the 28th International Conference on Machine Learning, 2011*, pp. 37–50.

[21] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *Proceedings of the 19th IEEE International Conference on Mobile Data Management*, 2018, pp. 125–134.

[22] J. Walpen, E. M. Mancinelli, and P. A. Lotito, "A heuristic for the od matrix adjustment problem in a congested transport network," *European Journal of Operational Research*, vol. 242, no. 3, pp. 807–819, 2015.

[23] J. Hu, B. Yang, C. Guo, and C. S. Jensen, "Risk-aware path selection with time-varying, uncertain travel costs: a time series approach," *The VLDB Journal–The International Journal on Very Large Data Bases*, vol. 27, no. 2, pp. 179–200, 2018.

[24] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, "Stochastic skyline route planning under time-varying uncertainty," in *Proceedings of the 30th IEEE International Conference of Data Engineering*, 2014, pp. 136–147.

[25] E. Erkut, A. Ingolfsson, and G. Erdoğan, "Ambulance location for maximum survival," *Naval Research Logistics (NRL)*, vol. 55, no. 1, pp. 42–58, 2008.

[26] D. Woodard, G. Nogin, P. Koch, D. Racz, M. Goldszmidt, and E. Horvitz, "Predicting travel time reliability using mobile phone GPS data," *Transportation Research Part C: Emerging Technologies*, vol. 75, pp. 30 – 44, 2017.

[27] E. Jenelius, "The value of travel time variability with trip chains, flexible scheduling and correlated travel times," *Transportation Research Part B: Methodological*, vol. 46, no. 6, pp. 762–780, 2012.

[28] M. Kijima and M. Ohnishi, "Stochastic orders and their applications in financial optimization," *Mathematical Methods of Operations Research*, vol. 50, no. 2, pp. 351–372, 1999.

[29] H. Levy, "Stochastic dominance and expected utility: survey and analysis," *Management Science*, vol. 38, no. 4, pp. 555–593, 1992.

[30] Q. Li, Y. M. Nie, S. Vallamsundar, J. Lin, and T. Homem-de Mello, "Finding efficient and environmentally friendly paths for risk-averse freight carriers," *Networks and Spatial Economics*, vol. 16, no. 1, pp. 255–275, 2016.

[31] J. Hu, C. Guo, B. Yang, C. S. Jensen, and L. Chen, "Recurrent multi-graph neural networks for travel cost prediction," *arXiv preprint arXiv:1811.05157*, 2018.

[32] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *Proceedings of the 24th ACM International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 858–866.

[33] D. Basak, S. Pal, and D. C. Patranabis, "Support vector regression," *Neural Information Processing-Letters and Reviews*, vol. 11, no. 10, pp. 203–224, 2007.

[34] C. A. Sims, "Macroeconomics and reality," *Econometrica: Journal of the Econometric Society*, pp. 1–48, 1980.

[35] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced Lectures on Machine Learning: ML Summer Schools, Canberra, Australia*, 2003, pp. 63–71.

[36] P. Yuan, C. Sha, X. Wang, B. Yang, A. Zhou, and S. Yang, "XML structural similarity search using mapreduce," in *Proceedings of the 11th International Conference on Web-Age Information Management*, 2010, pp. 169–181.

[37] B. Yang, Q. Ma, W. Qian, and A. Zhou, "TRUSTER: trajectory data processing on clusters," in *Proceedings of 14th International Conference on Database Systems for Advanced Applications*, 2009, pp. 768–771.

[38] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: Architecture and benchmarking for reinforcement learning in traffic control," *arXiv preprint arXiv:1710.05465*, 2017.

Part II

Papers

Paper A

Stochastic Weight Completion for Road Networks using Graph Convolutional Networks

Jilin Hu, Chenjuan Guo, Bin Yang, Christian S. Jensen

The paper has been accepted for presentation at the
34th IEEE International Conference on Data Engineering, 2019.

Abstract

Innovations in transportation, such as mobility-on-demand services and autonomous driving, call for high-resolution routing that relies on an accurate representation of travel time throughout the underlying road network. Specifically, the travel time of a road-network edge is modeled as a time-varying distribution that captures the variability of traffic over time and the fact that different drivers may traverse the same edge at the same time at different speeds. Such stochastic weights may be extracted from data sources such as GPS and loop detector data. However, even very large data sources are incapable of covering all edges of a road network at all times. Yet, high-resolution routing needs stochastic weights for all edges.

We solve the problem of filling in the missing weights. To achieve that, we provide techniques capable of estimating stochastic edge weights for all edges from traffic data that covers only a fraction of all edges. We propose a generic learning framework called Graph Convolutional Weight Completion (GCWC) that exploits the topology of a road network graph and the correlations of weights among adjacent edges to estimate stochastic weights for all edges. Next, we incorporate contextual information into GCWC to further improve accuracy. Empirical studies using loop detector data from a highway toll gate network and GPS data from a large city offer insight into the design properties of GCWC and its effectiveness.

A.1 Introduction

We are witnessing increasing needs for high-resolution routing that takes into account the dynamics and uncertainty of traffic [1, 2]. For instance, consider a person taking an autonomous taxi to catch a flight. If the taxi takes into account the travel speed distributions of different candidate paths, rather than just average speeds, it is able to choose the path with the highest probability of arriving on time [3]. Using only average speeds often leads to unreliable path choices [4]. Consider an example where two paths P_1 and P_2 lead to the airport. Based on the speed distributions of the edges in the paths, we are able to derive the paths' travel time distributions: P_1 has travel time distribution $\{(30, 0.2), (40, 0.8)\}$, meaning that traversing P_1 may take 30 or 40 mins with probabilities 0.2 and 0.8, respectively; and P_2 has distribution $\{(30, 0.5), (40, 0.3), (50, 0.2)\}$. If the passenger needs to arrive in the airport within 40 mins, taking P_1 is the best since it guarantees an on-time arrival. In contrast, taking P_2 has a 0.2 probability of arriving late. However, if considering only average travel times, P_2 is recommended since its average 37 mins is smaller than that of P_1, i.e., 38 mins.

Such high-resolution routing calls for a road network graph where *every* edge has a time-dependent, stochastic edge weight that captures uncertain traffic dynamics [5, 6]. Various types of traffic data, ranging from GPS data to loop detector data [7, 8], can be used to obtain time-dependent and stochastic edge weights. However, such traffic data often lacks the coverage needed to assign weights to all edges. Loop detectors are typically deployed only on some edges due to high deployment costs; and some loop detectors may be malfunctioning during some periods [9]. Next, a recent study shows that GPS data is often skewed, making it almost impossible to collect sufficient GPS data to cover all edges, during all time intervals [10, 11]. We call this the *data sparseness* problem.

In this paper, we formalize a *stochastic weight completion* problem. Given a traffic data set that only covers a subset of the edges in a road network, the objective is to associate each edge with accurate stochastic weights. Consider the example in Figure A.1 where the road network has 6 directed edges. Assume that during [8:15, 8:30], only edges e_5 and e_6 are covered by GPS data and can be associated with stochastic weights in this interval. These weights are represented as travel speed histograms, as shown in the figure. For instance, when traversing edge e_5 during [8:15, 8:30], it may take 5 m/s to 10 m/s with probability 0.3, 10 m/s to 15 m/s with probability 0.5, and 15 m/s to 20 m/s with probability 0.2. The weights for the remaining edges, i.e., e_1, e_2, e_3, and e_4, are missing. During other intervals, different edges may have GPS data and thus can be assigned weights, while the remaining edges cannot.

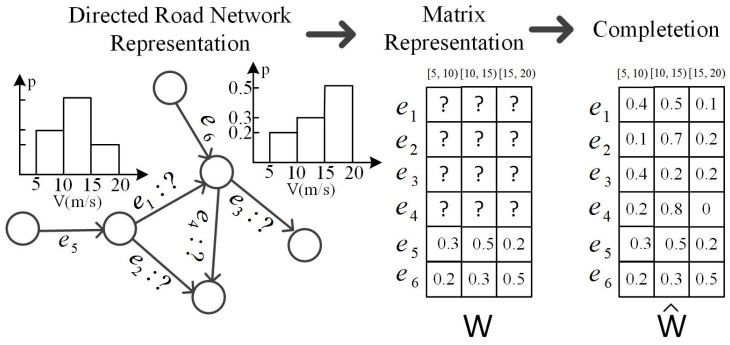


Fig. A.1: Example of Stochastic Weight Completion.

We convert this information into a matrix representation to facilitate processing, where each row represents the stochastic weight of an edge. For example, the first four rows in W are empty since weights for $e_1, \cdots, e_4$ are missing. The 5-th and 6-th rows represent the stochastic weights of e_5 and e_6. The goal of the stochastic weight completion is to estimate the stochastic weights for the edges that are not covered by traffic data, i.e., $e_1, \cdots, e_4$. The final result is a new matrix $\hat{W}$, where the empty rows in W are filled with values so that the stochastic weights are available for all edges.

Travel speeds of different edges in a road network exhibit high dependencies. In particular, studies addressing the data sparseness problem often assume that adjacent edges [9, 12, 13] tend to have similar (travel speed based) weights. Thus, the weights of the edges covered by traffic data can be propagated to their adjacent edges that are not covered by traffic data, using regression with judiciously designed loss functions that consider the discrepancies of the weights between the adjacent edges.

However, similarity assumptions may not always be true, since the correlations among the travel speeds of different edges can be very complex. Considering only the weight similarities between adjacent edges is unable to model complex correlations accurately. Further, existing studies only consider deterministic weights (e.g., average travel speeds). It is non-trivial to extend them to support stochastic weights such as travel speed distributions. A data driven approach that is able to capture complex correlations and to support stochastic weights is desirable.

We propose a data-driven, deep learning based framework, with the goal of capturing complex correlations among edge weights in a road network which in turn helps us estimate stochastic weights for edges without data. In particular, we first encode the topology of a road network using spectral graph theory [14] into a graph convolutional neural network (GCNN). Then, we feed available traffic data into the GCNN as both the input and the labeled output to let the GCNN learn complex correlations of edge weights in an "un-

supervised" manner, i.e., without requiring additional labeled data as output. The learned GCNN is then employed to complete stochastic weights for the edges without data. Further, we propose an advanced model that takes as input additional context information, e.g., time intervals, day of week, etc., which is able to further improve accuracy of the completed weights. The proposed framework is generic in the sense that it is able to support both stochastic and deterministic edge weights, and it also outperforms the state-of-the-art method when completing deterministic edge weights.

To the best of our knowledge, this is the first study of stochastic weight completion. In particular, we make four contributions. First, we formalize the stochastic weight completion problem. Second, we propose a data-driven framework using a graph convolution neural network to solve this problem. Third, we extend the framework by taking into account additional contextual information, which further improves accuracy. Fourth, we conduct extensive experiments using both GPS and loop detector data sets to provide insight into the effectiveness of the framework.

The remainder of the paper is organized as follows. Section A.2 covers related work. Section A.3 defines the setting and formalizes the problem. Section A.4 and Section A.5 detail the framework of graph convolutional weight completion (GCWC) and context aware graph convolutional weight completion (A-GCWC), respectively. Section A.6 reports experiments and results. Section A.7 concludes.

A.2 Related Work

Data Spareness in Road Networks: Although traffic prediction has been studied extensively [15, 16], only a few studies [9, 12, 13, 17] consider the data sparseness problem in road networks. The basic ideas of the *trajectory regression* problem [12, 13, 17] have been covered in Section B.1. More recently, a latent space model (LSM) is proposed to estimate the weights of edges that are not covered by loop detector data [9]. Non-negative matrix factorization is used as an encoder to learn the latent space features, which helps estimate the weights of edges without data. LSM is the state-of-the-art method.

All existing studies that address the data sparseness problem only consider deterministic weights and cannot be extended to support stochastic weights in a straightforward manner. In addition, they all employ linear models to cope with correlations among edge weights. However, such correlations can be highly non-linear [18]. We propose a graph convolutional weight completion framework that enables stochastic weight annotation while considering non-linear weight correlations.

Deep Learning in Transportation: RNNs with auto-encoders are proposed to enable traffic forecasts using traffic sensor data [18]. However, this pro-

posal ignores spatial correlations among the sensors. To address this problem, another method uses diffusion convolutional networks, which are able to model spatial correlations, together with RNNs to enable traffic forecasts [19]. Alternatively, classic convolutional networks can also model sensor correlations [20]. However, these methods are restricted to deterministic traffic values and do not support stochastic values. In addition, it is assumed that sufficient traffic data is available to cover all edges in a road network while our proposal considers the case when data is sparse. A more recent study focuses on travel time estimation for origin-destination pairs, but not for edges [21], using multi-task learning. Multi-task learning is also applied to distinguish trajectories from different drivers [22]. To the best of our knowledge, this paper proposes the first deep learning framework for stochastic weight completion in road network graphs.

A.3 Preliminaries and Problem Formulation

A.3.1 Road Network

A road network is often represented as a directed graph $H = (V, E)$, where vertex set V represents road intersections and edge set $E \subseteq V \times V$ represents directed edges. We model a road network as an edge graph $G = (E, A)$, where E is the edge set, and A is an $|E| \times |E|$ adjacency matrix that captures how the directed edges are connected. In particular, $A_{i,j} = 1$ if travel is possible from edge e_i to edge e_j or from edge e_j to edge e_i via a single vertex; otherwise, $A_{i,j} = 0$. This makes matrix A symmetric and the edge graph undirected.

Figure A.2 shows a road network and its corresponding edge graph and adjacency matrix. For example, $A_{5,2} = 1$ because a vehicle can travel from e_5 to e_2 by traversing one vertex in the road network. However, since travel from e_2 to e_1 and travel from e_1 to e_2 via a single vertex are not possible, we have $A_{2,1} = 0$.

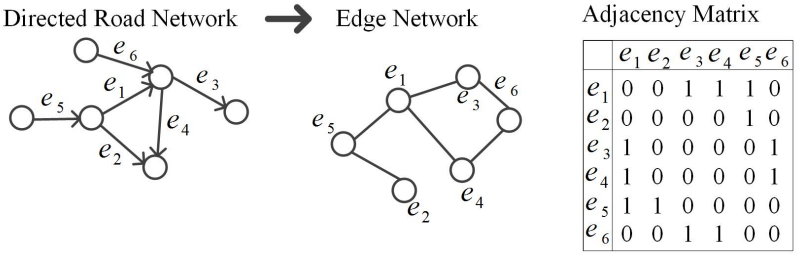


Fig. A.2: Road Network and Its Edge Graph.

A.3.2 Stochastic Weights

To capture time-dependent traffic, we partition a day into a number of intervals, e.g., 96 15-min intervals. Based on this, we introduce a stochastic weight function $\mathcal{F} : E \times TI \rightarrow D$, where TI is the whole time domain of interest and D is a set of all possible speed distributions. Given a specific edge $e_i \in E$ and a time interval $T_j \in TI$, function $\mathcal{F}(e_i, T_j)$ returns a stochastic weight that represents the speed distribution on edge e_i during interval T_j.

To instantiate the weight function $\mathcal{F}$, we need to assign stochastic weights to all edges for each interval $T_j \in TI$. In the following discussion, we focus on instantiating $\mathcal{F}$ for a specific interval T_j.

We first identify the traffic data available in T_j. Next, we partition all edges into subsets E_c and E_m, the edges with and without traffic data, respectively. This means that $E_c \cup E_m = E$ and $E_c \cap E_m = \emptyset$. In Figure A.1, we have $E_c = \{e_5, e_6\}$ and $E_m = \{e_1, e_2, e_3, e_4\}$ for interval [8:15, 8:30].

For each edge $e_c \in E_c$, which is covered by traffic data, we are able to derive a stochastic weight and thus able to instantiate $\mathcal{F}(e_c, T_j)$. However, we are unable to instantiate $\mathcal{F}(e_m, T_j)$ if edge $e_m \in E_m$. For example, during [8:15, 8:30], GPS trajectories exist for edges e_5 and e_6, and thus we are able to use them to build speed distributions, i.e., stochastic weights for e_5 and e_6, during [8:15, 8:30], thus instantiating $\mathcal{F}(e_5$, [8:15, 8:30]) and $\mathcal{F}(e_6$, [8:15, 8:30]).

We use equi-width histograms to represent speed distributions. In particular, an equi-width histogram is a set of bucket-probability pairs $\{(b_i, p_i)\}$. A bucket $b_i = [l_i, u_i)$ represents the speed range from l_i to u_i, and all buckets have the same range size. Next, p_i is the probability that the speed falls into range b_i. For example, the speed histogram $\{([0, 20), 0.5), ([20, 40), 0.3), ([40, 60), 0.2)\}$ for edge e_5 means that the probability that the speed (m/s) on e_5 falls into $[0, 20)$, $[20, 40)$, and $[40, 60)$ is 0.5, 0.3, and 0.2, respectively.

We use the same finest bucket range size for all edges' speed histograms. Thus, we can ignore the buckets and represent the speed histogram of each edge as a vector. For example, when choosing 20 m/s as the bucket range size, e_5's speed histogram can be represented as $\langle 0.5, 0.3, 0.2 \rangle$.

Assume that we have $n = |E|$ edges in the road network and we use a histogram with m buckets to present a stochastic weight. Then, the stochastic weights of all edges in interval T_j can be represented as an $n \times m$ matrix W. A row vector $w_{i\cdot}$ in W corresponds to the vector representation of edge e_i's stochastic weight, i.e., its speed histogram. If an edge $e_i \in E_m$, $w_{i\cdot}$ is an empty vector. For example, the first four rows in W in Figure A.1 are empty vectors.

A.3.3 Problem Formulation

Consider a time interval T_j of a day. Given the instantiated stochastic weight matrix W, the *stochastic weight completion* problem is that of completing the empty rows in W to produce a new stochastic weight matrix $\widehat{W}$ without empty rows. This is equivalent to instantiating $\mathcal{F}(e_m, T_j)$ for each edge $e_m \in E_m$ that is not covered by traffic data.

To ease the presentation, Table B.2 lists important notation that we use throughout this paper.

Notations	Definition
G	Edge graph of a road network
E, V	Edge set, Vertex set
A	Adjacency matrix of G
n	Total number of edges
m	Total number of buckets in a histogram
W	Input Weight Matrix
W_G	Ground Truth Weight Matrix
$\widehat{W}$, $\widetilde{W}$	Estimated, complete weight matrix
X_i	A context variable

Table A.1: Notation.

A.3.4 Solution Overview

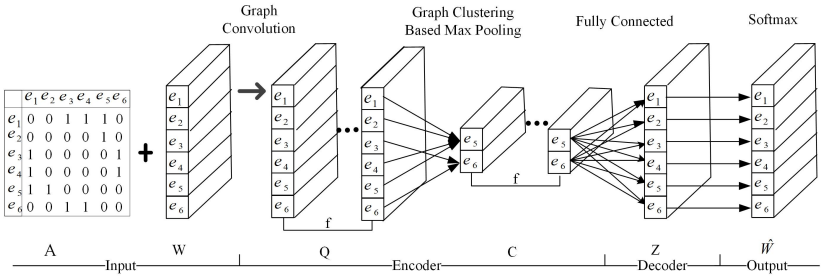


Fig. A.3: System Architecture for GCWC.

We propose a basic model and an advanced model to solve the problem. The basic model takes as input an instantiated, *incomplete* stochastic weight matrix W and the edge adjacency matrix A. The basic model is able to learn correlated edge features from W and A using graph convolution filters and thus derives a *complete* stochastic weight matrix $\widehat{W}$.

However, the basic model does not make use of important *contextual* information, e.g., time intervals and the day of the week. For example, traffic

in peak vs. off-peak intervals may be different and traffic on weekdays vs. weekends may also be different. To better utilize such contexts that are missing in the basic model, but may be useful for completing weights, we propose the advanced model. This model also takes as input the available contexts and applies a Bayesian inference model to construct dependency relationships between contextual features and the output of the basic model with the goal of improving the accuracy of the complete $\widehat{W}$.

We present the basic and the advanced models in Sections A.4 and A.5, respectively.

A.4 Graph Convolutional Weight Completion

A.4.1 Intuitions and Framework Overview

Since traffic on one edge may influence traffic on many other edges [9, 17], it is intuitive to assume that stochastic weights of different edges share correlated features. We model such features by transform stochastic weight matrix W into a set $\mathbb{C} = \{C_i\}_{i=1}^{f}$ of latent variables that captures correlations among the weights of edges. Based on the latent variables in $\mathbb{C}$, we construct a new stochastic weight matrix $\widehat{W}$ without empty rows. The whole process can be regarded as an *auto-encoder* [23, 24], where we first *encode* incomplete weight matrix W into a set of features $\mathbb{C}$ in a latent space and then *decode* $\mathbb{C}$ back to a complete weight matrix $\widehat{W}$.

Figure A.3 shows an overview of the basic model for Graph Convolutional Weight Completion, denoted as GCWC, where we adopt the intuition of the auto-encoder.

We provide stochastic weight matrix W and adjacency matrix A to GCWC as input. Next, we use convolutional and max pooling layers to encode W into a set of features $\mathbb{C}$, which can be regarded as the encoding process. Finally, we map the encoded features $\mathbb{C}$ to the final output layer with the help of a fully connected layer and thus obtain an estimated weight matrix $\widehat{W}$, where each edge has a stochastic weight. This corresponds to the decoding process.

In the training phase, input matrix W is also used as a source of labels for conducting back-propagation. We learn the parameters of our framework with the objective of minimizing a loss function that is defined based on the KL-divergence between the estimated stochastic weights and the ground truth stochastic weights, i.e., the instantiated stochastic weights of the edges that are covered by traffic data (details to be provided in Section A.4.5).

A.4.2 Convolutional Layer

In classical convolutional neural networks (CNNs), 2D convolutional filters are applied in convolutional layers based on the assumption that nearby elements in the input matrix share local features [14]. For example, when representing an image as a matrix, nearby elements, e.g., pixels, share local features, e.g., represent parts of the same object. However, in our setting, the input stochastic weight matrix may not always satisfy this assumption—two adjacent rows in matrix W may represent two geometrically distant road network edges and may not share any features. In Figure A.1, although the rows for e_5 and e_6 are adjacent in W, e_5 and e_6 are not adjacent in the road network. This renders classical 2D convolutional filters ineffective in our setting and calls instead for new filters that take into account the topology of the road network, e.g., through the use of adjacency matrix A. Our solution utilizes recently invented graph convolutional neural networks (GCNNs) [14, 25].

Background on GCNNs: In GCNNs, *graph convolutional filters* [14, 25, 26], which take into account the topology of a road network based on spectral graph theory, are employed to replace the classic 2D convolutional filters in the convolutional layers. Graph convolutional filters consider that topologically adjacent elements in a graph share local features. By using graph convolutional filters, we can "propagate" the input stochastic weights to adjacent, correlated edges during convolutions via the road network topology. In the literature, different variations of graph convolutional filters exist. We use *Simplified ChebNet* [26], due to its efficiency and effectiveness.

Simplified ChebNet: Since graph convolutional filters are based on spectral graph theory, the central component of a graph filter is the graph Laplacian [14, 26]. To construct the graph Laplacian L, we utilize adjacency matrix A that captures the topology of the edge graph of a road network (see Section A.3). In particular, $L = D - A$, where D is the diagonal degree matrix with $D_{i,i} = \Sigma_j A_{i,j}$. Next, we derive the scaled Laplacian $\tilde{L} = 2L/\lambda_{max} - I$, where λ_{max} is the maximum eigenvalue of L and I is an identify matrix.

In the convolution layer, we apply graph convolutional filters to the input stochastic weight matrix W by using scaled Laplacian $\tilde{L}$. Recall that column vector $w_{\cdot j} \in \mathbb{R}^{n\times 1}$ of the input stochastic weight matrix W represents the weights of all n edges in the j-th bucket (where $j \in [1, m]$). Based on $w_{\cdot j}$ and $\tilde{L}$, we first generate a matrix $Y_j \in \mathbb{R}^{n\times k}$ and $Y_j = [\hat{x}_0, \hat{x}_1, \cdots, \hat{x}_{k-1}]$, where $\hat{x}_i \in R^{n\times 1}$ is a column vector and k is a hyper-parameter. Specifically, we have $\hat{x}_0 = w_{\cdot j}$, i.e., the original input column vector. Next, $\hat{x}_1 = \tilde{L}\hat{x}_0$. When $k \geq 2$, $\hat{x}_k$ is defined recursively: $\hat{x}_k = 2\tilde{L}\hat{x}_{k-1} - \hat{x}_{k-2}$. Further, we define a filter τ as an $\mathbb{R}^{k\times 1}$ matrix, and we use a total of f filters.

Based on the above, the convolution using graph convolutional filters on

an input vector $w_{\cdot j}$ is defined based on τ and Y_j using Equation A.1.

$$H_j = \tanh(Y_j \cdot \tau + b) = \tanh([\hat{x}_0, \hat{x}_1, \cdots, \hat{x}_{k-1}] \cdot \tau + b) \tag{A.1}$$

Here, $\cdot$ denotes matrix multiplication, and thus we have $H_j \in \mathbb{R}^{n \times 1}$.

The example in Figure A.1 features 6 edges and 3 buckets, which yields the first column vector $w_{\cdot 1} = [0, 0, 0, 0, 0.3, 0.2]^T$ for the $[5, 10)$ bucket (i.e., the first bucket). Assuming $k = 2$, we construct the corresponding Y_1 from adjacency matrix A as $\begin{pmatrix} 0 & 0 & 0 & 0 & 0.3 & 0.2 \\ -0.3 & -0.3 & -0.2 & -0.2 & 0.3 & 0.2 \end{pmatrix}^T$. Next, we apply a filter $\tau = [1, -1]^T$ to Y_1 and get $H_1 = [0.3, 0.3, 0.2, 0.2, 0, 0]^T$. Thus, data for edges e_5 and e_6 are propagated to edges $e_1, \cdots, e_4$ by means of graph convolution.

Given an input stochastic weight matrix W, we apply all f filters, $\tau_1, \ldots, \tau_f$, to each column vector $w_{\cdot j}$ in W, $j \in [1, m]$. Thus, for each column vector $w_{\cdot j}$, we obtain its f corresponding matrices $H_j^1, \ldots, H_j^f$. Next, for each filter τ_l, $1 \leq l \leq f$, we concatenate the convoluted matrices of all column vectors as an $\mathbb{R}^{n \times m}$ matrix $Q_l = [H_1^l, \ldots, H_m^l]$. Thus, we obtain a total of f matrices, $Q = [Q_1, \ldots, Q_f]$, as the final result from the convolutional layer. Referring back to the auto-encoder model, here the f filters will eventually construct a set of f features in the latent space.

A.4.3 Pooling Layer

Although the convolutional layer tries to propagate stochastic weights to the edges that are not covered by traffic data, some edges may still have zero values. Therefore, we further compress the convoluted results, i.e., Q_1, Q_2, …, Q_f, via max-pooling layers, which follows the design principles of traditional CNNs. Specifically, we employ a multi-level graph-based pooling algorithm [14] to first identify clusters of edges using the graph topology and distributions of available stochastic weights, and then the identified clusters are used as pools to perform max-pooling operations. For example, Figure A.3 shows an example where e_1, e_2, e_4, e_5 are clustered into one pool and e_3 and e_6 are clustered into another pool. Based on the pools, each convoluted matrix Q_l is further compressed into a more compact matrix C_l, where $1 \leq l \leq f$. Now, we regard such compact matrices as the feature set $\mathbb{C} = \{C_1, C_2, \ldots, C_f\}$ in the latent space.

A.4.4 Output Layer

After pooling, we obtain compact matrices that capture representative features of input matrix W. Next, we perform decoding that uses the compact matrices to produce a new stochastic weight matrix $\widehat{W}$. We first utilize a fully-connected (FC) layer to obtain a matrix Z, representing stochastic weights of all edges decoded from feature set $\mathbb{C}$. In particular, $Z = [z_{1\cdot}, \cdots, z_{n\cdot}]^T$, where

n is the total number of edges, and $z_{i\cdot} \in \mathbb{R}^m$, with m being the number of buckets in histograms, is used to estimate the stochastic weight vector for the i-th edge.

The final output $\widehat{W} \in \mathbb{R}^{n \times m}$ must meet two requirements to be a meaningful histogram: (1) for each value $\widehat{w}_{i,j}$ of $\widehat{W}$, we have $0 \leq \widehat{w}_{i,j} \leq 1$, with $1 \leq i \leq n$ and $1 \leq j \leq m$; (2) $\sum_j \widehat{w}_{i,j} = 1$, meaning that the sum of values in a histogram for the i-th edge equals 1, thus aligning with our definition of stochastic weights (see Section A.3.2).

To this end, a softmax function is applied to every $z_{i\cdot}$:

$$\widehat{w}_{i\cdot} = softmax(z_{i\cdot}), 1 \leq i \leq n, \tag{A.2}$$

where $\widehat{w}_{i\cdot}$ is the estimated histogram for the i-th edge. Thus, we have $\widehat{W} = [\widehat{w}_{1\cdot}, \cdots, \widehat{w}_{n\cdot}]^T$.

A.4.5 Loss Function

The loss function $L(W, \widehat{W})$ of GCWC measures the discrepancy between the input stochastic weight matrix $W \in \mathbb{R}^{n \times m}$ and the estimated stochastic weight matrix $\widehat{W} \in \mathbb{R}^{n \times m}$ using KL-divergence, as shown in Equation A.3.

$$\begin{aligned} L(W, \widehat{W}) &= \sum_{i=1}^{n} I_i \cdot KL(w_{i\cdot} || \widehat{w}_{i\cdot}), \text{where} \\ KL(w_{i\cdot} || \widehat{w}_{i\cdot}) &= \sum_{j=1}^{m} \widehat{w}_{ij} \cdot \log \frac{\widehat{w}_{ij} + \epsilon}{w_{ij} + \epsilon}. \end{aligned} \tag{A.3}$$

Specifically, we have n edges in total and the function focuses on the edges that are covered by traffic data, i.e., the non-empty rows in W, since we use the loss function to measure whether the estimated weights are similar to the original weights. Thus, we apply an indicator I_i, and we set $I_i = 1$ if the i-th edge is covered by traffic data; otherwise, we set $I_i = 0$.

Then, the KL-divergence measures the divergence of the estimated weight $\widehat{w}_{i\cdot}$ from the actual stochastic weight $w_{i\cdot}$ on the i-th edge, where w_{ij} and $\widehat{w}_{ij}$ are the actual and estimated weights for the i-th edge at the j-th bucket. In total, we have m buckets. We apply a positive small value ϵ to prevent having a zero when using the *log* function.

A.5 Context Aware Graph Convolutional Weight Completion

A.5.1 Context Aware Graph Convolution Neural Network

We proceed to present how contexts can be integrated into GCWC to enable an advanced model A-GCWC, as presented in Figure A.4.

When we consider stochastic weight matrix W that is instantiated during time interval T_j of a day, we can consider a set $\mathbb{X}$ of contexts. Specifically, we consider three contexts—time interval X_T, day of the week X_D, and row flag X_R.

Assume that a day is partitioned into 96 15-min intervals. We are able to use a one-hot vector with 96 bits to represent a specific time interval. For example, if a weight matrix W is instantiated during [0:15, 0:30], the 2-nd bit in the vector is set to 1 and all remaining bits are set to 0. This vector is used as the time interval context X_T. Next, we use a 7-bit vector for X_D to represent the week days. Finally, the row flag X_R of a weight matrix W indicates the non-empty rows. In particular, X_R is a vector with n bits, where n is the number of edges. If the k-th edge is not empty, the k-th bit in X_R is set to 1; otherwise, it is set to 0.

Figure A.4 gives an overview of the A-GCWC framework, which consists of a basic *GCWC*, a *context embedding* module, and a *Bayesian inference* module.

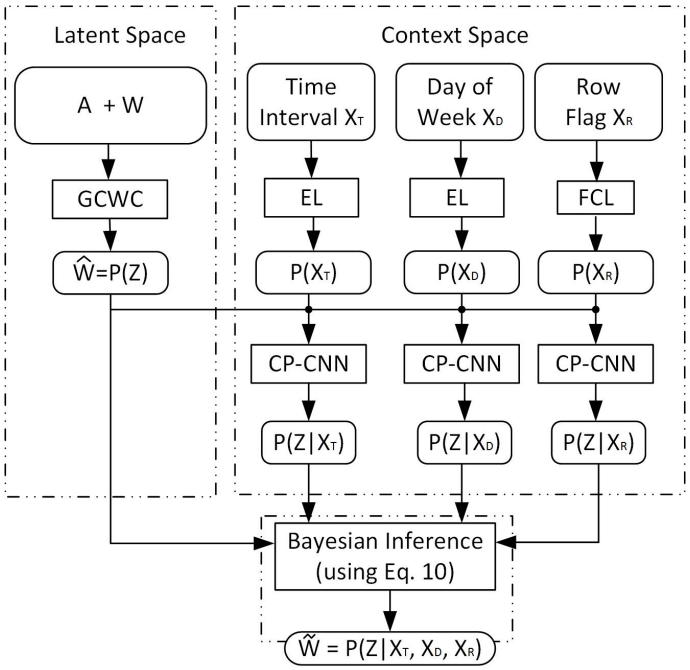


Fig. A.4: Context Aware Graph Convolution Neural Network.

The basic GCWC takes as input an incomplete weight matrix W and an adjacency matrix A, and it outputs an estimated complete weight matrix. The intuition is to learn stochastic weights of all edges without considering the contexts. Recall that in GCWC, we apply a softmax function on the output

of the fully connected layer, i.e., Z, to produce a valid weight matrix $\widehat{W}$. Here, we use $P(Z)$ to denote the output of GCWC $\widehat{W}$ where $P(\cdot)$ denotes the softmax function (see Section A.4.4), which can also be interpreted as a probability function.

The context embedding module first embeds the provided contexts X_T, X_D, and X_R into a space with the same dimensionality. Then, together with the output from the GCWC, i.e., $P(Z)$, it represents the conditional probability of having weight matrix Z conditioned on each context, denoted as $P(Z|X_T)$, $P(Z|X_D)$, and $P(Z|X_R)$, respectively.

The Bayesian inference module takes as input $P(Z)$ from the GCWC, and $P(Z|X_T)$, $P(Z|X_D)$, and $P(Z|X_R)$ from the context embedding module, and it infers the conditional probability for all contexts. Thus, more accurate stochastic weights given all types of context can be learned as $P(Z|X_T, X_D, X_R)$. We denote the estimated weight matrix from A-GCWC as $\widetilde{W} = P(Z|X_T, X_D, X_R)$, which is different from the estimated matrix from the basic GCWC, $\widehat{W} = P(Z)$.

A.5.2 Context Embedding Module

The proposed context embedding module is generic—although we only use three types of context, the module is extendable to include, e.g., weather conditions, wind speeds and traffic flows, that may further improve the overall accuracy, if they are available.

Due to different representations of X_T, X_D, and X_R, we apply two different models, i.e., an embedding layer and a fully connected layer, to incorporate the different types of contexts into A-GCWC.

Embedding Layer

We apply an embedding layer (EL) for time interval context X_T and day of the week context X_D, since both are represented as one-hot vectors. The embedding method [27] was initially proposed in order to effectively transform a categorical value represented by a high-dimensional, one-hot vector into a low-dimensional vector. As a result, neural network can process the categorized value more efficiently.

For generality, we assume that a total of α time intervals are considered. Thus, we have vector $X_T \in \mathbb{R}^{\alpha \times 1}$ as the one-hot representation for the time interval. The embedding layer is able to transform X_T into $\widehat{X}_T \in \mathbb{R}^{\beta \times 1}$ where $\beta \ll \alpha$. Next, we apply the softmax activation function to compute a distribution for X_T using $\widehat{X}_T$, denoted as $P(X_T)$. Specifically, $P(X_T)$ represents the probability of having each embedded context value in $\widehat{X}_T$. Similarly, we are able to derive $P(X_D)$.

Fully Connected Layer

A row flag vector X_R may have more than one occurrence of "1". Since some embedding methods, especially those based on lookup tables [28], require inputs represented as one-hot representations, we instead apply a fully connected layer (FCL) to embed $X_R \in \mathbb{R}^{n\times 1}$ into a smaller space $\widehat{X}_R \in \mathbb{R}^{\beta\times 1}$, where $\beta \ll n$, as follows.

$$P(X_R) = \sigma(\widehat{X}_R) \text{ where } \widehat{X}_R = M \times X_R + b, \tag{A.4}$$

where $M \in \mathbb{R}^{\beta\times n}$ is a weight matrix, $b \in \mathbb{R}^{\beta\times 1}$ is a bias vector, and σ is a softmax function. Based on the above, $P(X_R)$ represents the probability of each value occurring in $\widehat{X}_R$.

Computing Conditional Probabilities

We utilize a conditional probability convolutional neural network (CP-CNN) to capture the dependency between the stochastic weight of an edge $z_{j\cdot}$ and each type of context, as shown in Figure A.5. To simplify our discussion, we use X_i to denote a context, where X_i can be X_T, X_D, or X_R, in the following.

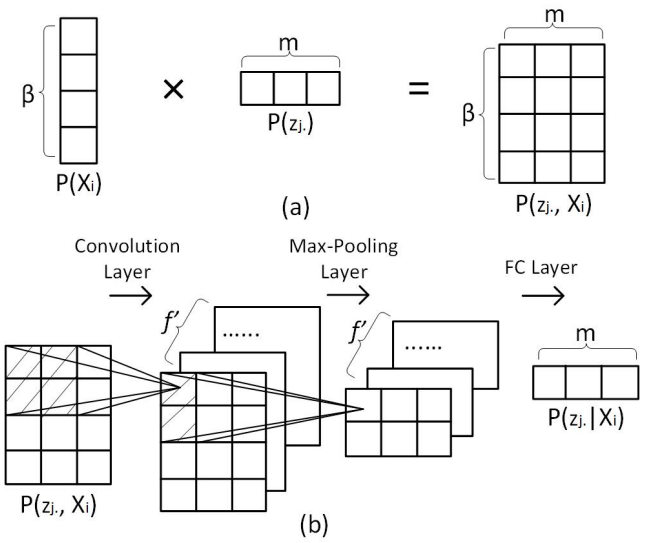


Fig. A.5: Conditional Probability CNN (CP-CNN).

As shown in Figure A.5(a), we multiply $P(X_i)$ with $P(z_{j\cdot})$, where $P(X_i) \in \mathbb{R}^{\beta\times 1}$ is the probability distribution of the embedded context values for context X_i and $P(z_{j\cdot}) \in \mathbb{R}^{1\times m}$ is the estimated stochastic weight for the j-th edge obtained by GCWC, with m being the bucket size. As a result, we obtain a matrix, denoted as $P(z_{j\cdot}, X_i) \in \mathbb{R}^{\beta\times m}$, that associates each bucket of the

j-th edge's weight with each embedded context value in context X_i, based on which we learn whether a bucket and an embedded context value exhibit a dependency.

Next, we capture such possible dependencies using classical convolutional filters. Following the running example in Figure A.1, we have $m = 3$ buckets as $[0, 20)$, $[20, 40)$, and $[40, 60)$. If we apply a filter of size 2×2 to the 4 shadowed squares in the leftmost matrix of Figure A.5(b), we are able to capture the dependency between 2 buckets, e.g., $[0, 20)$, $[20, 40)$, and 2 values of context X_i. This is intuitive, since the probabilities of speeds falling into $[0, 20)$ and $[20, 40)$ influence each other, and they may also be influenced by similar contexts, e.g., intervals [8:00, 8:15] and [8:15, 8:30].

As shown in Figure A.5(b), we utilize f' filters and obtain a total of f' matrices, each of the same sizes as $\mathbb{R}^{\beta \times m}$. Next, a classical max-pooling layer with window size 2 is applied to learn more representative dependencies, giving rise to a total of f' matrices, each of the same size as $\mathbb{R}^{\frac{\beta}{2} \times m}$. We apply a fully connected layer to concatenate f' matrices into an $\mathbb{R}^{1 \times m}$ matrix as the conditional probability between the stochastic weight of the j-th edge and context X_i, denoted as $P(z_{j\cdot}|X_i)$.

After we conduct the same procedure for all n edges, we obtain $P(Z|X_i)$, i.e., the weight matrix when considering context X_i.

A.5.3 Bayesian Inference

We utilize the Bayesian Inference module to derive a probability distribution for weight matrix Z given all types of contexts X_T, X_D, and X_R.

For generality, we assume that we have N types of contexts $X_1, \ldots, X_N$ and that we have obtained $P(Z|X_1), \ldots, P(Z|X_N)$ as the conditional probability of Z given each context X_i, where $i \in [1, N]$ (cf. Section A.5.2). Further, we have obtained $P(Z)$ from the basic GCWC. We aim to infer $P(Z|X_1, \ldots, X_N)$ as the stochastic weight $\widetilde{W}$, i.e., the conditional probability of Z given all contexts $X_1, \ldots, X_N$, from $P(Z|X_1)$, ..., $P(Z|X_N)$ and $P(Z)$.

To this end, we make the assumption that different contexts are independent, and thus we have $P(X_1, \ldots, X_N) = \prod_{i=1}^{N} P(X_i)$. This is a reasonable assumption because, for example, time intervals, day of week, and row flags

do not have obvious correlations. Then, we have

$$P(Z|X_1,\ldots,X_N) \tag{A.5}$$

$$= \frac{P(Z,X_1,\ldots,X_N)}{P(X_1,\ldots,X_N)} = \frac{P(Z,X_1,\ldots,X_N)}{\prod_{i=1}^{N} P(X_i)} \tag{A.6}$$

$$= \frac{P(Z)P(X_1,\ldots,X_N|Z)}{\prod_{i=1}^{N} P(X_i)} = \frac{P(Z)\prod_{i=1}^{N} P(X_i|Z)}{\prod_{i=1}^{N} P(X_i)} \tag{A.7}$$

$$= \frac{\prod_{i=1}^{N}(P(X_i|Z)P(Z))}{[P(Z)]^{(N-1)}\prod_{i=1}^{N} P(X_i)} = \frac{\prod_{i=1}^{N}\frac{P(X_i,Z)}{P(X_i)}}{[P(Z)]^{(N-1)}} \tag{A.8}$$

$$= \frac{\prod_{i=1}^{N} P(Z|X_i)}{[P(Z)]^{(N-1)}} \tag{A.9}$$

Here we keep using Bayesian rule and the independence assumption in Equations A.6, A.7, and A.8.

According to Equation (A.9), we compute $\widetilde{W} = P(Z|X_T, X_D, X_R)$ using Equation (A.10).

$$P(Z|X_T, X_D, X_R) = \frac{P(Z|X_T)P(Z|X_D)P(Z|X_R)}{[P(Z)]^2}, \tag{A.10}$$

where $P(Z|X_T, X_D, X_R)$ is the estimated stochastic weight given context X_T, X_D, and X_R, which is further normalized to get $\widetilde{W}$. The normalization makes sure that: (1) for each $\widetilde{w}_{i,j} \in \widetilde{W}$, $0 \leq \widetilde{w}_{i,j} \leq 1$, with $1 \leq i \leq n$ and $1 \leq j \leq m$; (2) $\sum_j \widetilde{w}_{i,j} = 1$.

A.5.4 Loss Function

The loss function for A-GCWC is based on KL-divergence—we compute $L(W, \widetilde{W})$ using Equation A.3.

A.6 Experiments

A.6.1 Experimental Setup

Data sets

We use two traffic data sets for studying the effectiveness of the proposed models. In both data sets, we report results on the setting of HIST-8, where a histogram has 8 buckets. In particular, the bucket length is 5 m/s, and the range is from 0 to 40 m/s, yielding a total of 8 buckets. Due to the space limitation, we do not report results for the setting of HIST-4 where 4 buckets are used. This setting yields similar results as the HIST-8 setting.

Highway Tollgates Network (HW): *HW* consists of 1.07 million travel time records from loop detectors located in $n = 24$ links in a highway tollgate network. The records cover the period from 19/07/2016 to 31/10/2016. This data is obtained from a big data challenge. We partition a day into 96 15-min intervals. Given the $m = 8$ buckets used in HIST-8, each input matrix $W \in \mathbb{R}^{n \times m}$, i.e., the stochastic weights, is an $\mathbb{R}^{24 \times 8}$ matrix, and we have a total of 96 matrices per day. For each matrix W, we use the corresponding time interval, day of the week, and row flag vector of W as the contexts of the matrix.

City Road Network (CI): *CI* consists of 1.4 billion GPS records from 14,864 taxis obtained in the period from 03/08/2014 to 30/08/2014 in Chengdu, China. This data is obtained from a big data competition. To get reliable results, we use $n = 172$ connected road segments with sufficient GPS data in the experiments. In particular, we choose a dense subgraph with 172 edges where almost all edges have GPS data in most time intervals. This design decision is made because we need to ensure that each edge has sufficient data to derive ground truth weights, such that we can evaluate the accuracy of the estimated weights. Specifically, 1) we select edges with the top-5000 largest amounts of GPS records; 2) we derive the connected subgraphs using the 5000 edges; 3) we use the largest connected subgraph with 172 edges in the experiments.

Similarly, we have 96 time intervals per day in *CI*. Each interval has a stochastic weight matrices $W \in \mathbb{R}^{172 \times 8}$, which is associated with three types of context.

Ground Truth and Input Data

We first introduce ground truth weight matrix W_G and then show how we generate input matrix W from W_G.

Given a time interval, we are able to instantiate a ground truth matrix W_G from the available traffic data. We only instantiate weights for edges with at least 5 speed records. Next, we randomly select $n \times rm$ edges from a total of n edges, where rm is a removal ratio. We set the stochastic weights of the chosen edges in W_G to be zeros, giving rise to the incomplete input matrix W. We use four values for rm: 0.5, 0.6, 0.7, and 0.8. Taking W as input, we use a method to estimate a complete weight matrix $\widehat{W}$ that covers all edges. Then we evaluate the accuracy of $\widehat{W}$ by comparing $\widehat{W}$ with the ground truth matrix W_G.

Note that the matrix W_G may already have empty rows for some edges since the available traffic data in an interval may not cover these edges. Although there are edges with empty rows, we do not estimate a complete weight matrix based on W_G directly. If we simply did so, although we would be able to fill in weights for the edges without data, we are not able to eval-

uate the accuracy of the estimated weights. This is why we instead remove $n \times rm$ edges' weights from W_G to create input matrices.

For each data set, we order all input matrices in ascending time order, and we partition these into 5 equal-sized portions. Then we use 5-fold cross validation in all the experiments—in each run, 4 folds of matrices are used for training and validating, and the remaining 1 fold of matrices are used for testing. We run this workload 5 times in total, so that each fold of matrices is used for testing once.

Model Functionalities

The proposed models, GCWC and A-GCWC, are generic and extendable, and they are able to support different functionalities when changing the configurations slightly. Here, we consider three different functionalities: *estimation* and *prediction* of stochastic weights, in the form of histograms, and estimation of *average* speeds, in the form of deterministic values. We use $W@T$ to denote weight matrix W during interval T.

Estimation: Given input matrix $W@T_i$ that represents stochastic weights at time interval T_i, where some edges do not have weights, we estimate $\widehat{W}@T_i$ that has the estimated stochastic weights at T_i for all edges.

During training, we have a set of training matrices $\{W@T_k\}$. For each training matrix in $\{W@T_k\}$, we use the matrix itself as a label to train the two models. During testing, given input matrix $W@T_i$, the estimated $\widehat{W}@T_i$ is compared with the ground truth matrix at T_i, i.e., $\widehat{W}_G@T_i$, to evaluate the accuracy.

Prediction: Given input matrix $W@T_i$ that represents stochastic weights at time interval T_i, where some edges do not have weights, we predict $\widehat{W}@T_{i+1}$ that contains the predicted stochastic weights in the next time interval T_{i+1} for all edges.

During training, we have a set of training matrices $\{W@T_k\}$. For each training matrix $W@T_k$, we use the ground truth matrix in the next interval, i.e., $W_G@T_{k+1}$, as a label to train the two models. We make sure that $W_G@T_{k+1}$ has the same rm as the input matrix $W@T_k$. For example, when rm is 0.6, both the input matrix $W@T_k$ and label matrix $W_G@T_{k+1}$ have 60% empty edges. During testing, given input matrix $W@T_i$, the estimated $\widehat{W}@T_{i+1}$ is compared with the ground truth weight matrix at T_{i+1}, i.e., $\widehat{W}_G@T_{i+1}$, to evaluate the accuracy. Table A.2 summarizes the settings of estimation and prediction.

	Training		Testing	
	Input	Label	Input	Ground Truth
Estimation	$W@T_k$	$W@T_k$	$W@T_i$	$W_G@T_i$
Prediction	$W@T_k$	$W_G@T_{k+1}$	$W@T_i$	$W_G@T_{i+1}$

Table A.2: Settings, Estimation vs. Prediction.

Average: This setting is similar to *Estimation*. Given input matrix $W@T_j$, we estimate a deterministic average speed value for each edge during the same interval T_j, rather than a speed histogram. To achieve this, we replace the softmax function in Equation (A.2) of Section A.4.4 with the sigmoid function and thus obtain $P(Z) \in \mathbb{R}^{n\times 1}$ in the latent space (see Figure A.4). We also replace the normalization on $P(Z|X_T, X_D, X_R)$ in Equation (A.10) of Section A.5.3 with the sigmoid function so that the output $\widehat{W} \in \mathbb{R}^{n\times 1}$ represents the estimated average speeds for all edges in time interval T_j. The ground truth matrix $W_G@T_j$ of average speeds during T_j is derived by averaging all speed records for each edge.

Model Settings

Model Construction: We present hyperparameters of all models in Table C.2. We refer to the models constructed for the estimation and prediction of speed histograms as being of type HIST, and for the average as being of type AVG. For both types, we describe the hyperparameters used for GCWC and A-GCWC, including the learning rate (LR), learning rate decay (Decay), Dropout, and regularization (Regul). The column with header "#Para" indicates the total number of parameters used in a deep learning model (e.g., parameters for convolution filters, full-connected layers, biases used in activation functions, etc.). This reflects the complexity of different neural networks. The higher the value is, the more complex the corresponding neural network is. The #Para column in Table C.2 shows that the total number of parameters in CNN, GCWC, and A-GCWC are similar, meaning that, compared to classical CNN, the proposed GCWC and A-GCWC do not significantly increase the model complexity.

We use the following notation to describe the model construction: $C_f^{k_1\times k_2}$ denotes a convolution layer that has f filters, each of which is a $k_1 \times k_2$ matrix; P_k denotes a pooling layer of size and stride k; FC_k denotes a fully connected layer with k hidden units. For example, GCWC for HW, HIST is constructed as $C_{16}^{8\times 1}$-P_4-$C_{16}^{8\times 1}$-P_2-FC_n, where n is the number of edges, which varies across data sets.

Further, β, the dimension of the embedded context space, is set to 4 for A-GCWC for all data sets.

We obtain an optimal set of hyperparameters with Bayesian optimization using Gaussian Processes search.

Model Complexity: The time complexity of the training of GCWC and A-GCWC is $O((F^2K + n^2F) \times m \times S)$ and $O((F^2K + n^2F + nKF) \times m \times S)$, respectively, where F is the maximum number of convolutional filters, K is the maximum size of the convolutional filters, n is the number of edges, S is the batch size, and m is the number of histogram buckets. The average running time on CI for training is 36 ms (16 ms) per training batch for A-GCWC

Type	Data	Model	Configuration	#Para
HIST	HW	CNN	$C_{16}^{8\times1}$-P_4-$C_{16}^{8\times1}$-P_2-FC_{24}	18,840
		DR	$C_{16}^{8\times1}$-$C_{16}^{8\times1}$-$C_{16}^{8\times1}$-$C_8^{8\times1}$	39,680
		GCWC	$C_{16}^{8\times1}$-P_4-$C_{16}^{8\times1}$-P_2-FC_{24}	19,224
		A-GCWC	$C_{16}^{8\times1}$-P_4-$C_{16}^{8\times1}$-P_2-FC_{24} + $C_4^{2\times2}$-P_2-$C_8^{2\times2}$-P_2-FC_1	20,184
	CI	CNN	$C_8^{8\times1}$-P_2-$C_8^{4\times1}$-P_2-FC_{172}	32,412
		DR	$C_{16}^{8\times1}$-$C_{16}^{8\times1}$-$C_{16}^{8\times1}$-$C_8^{8\times1}$	39,680
		GCWC	$C_8^{8\times1}$-P_2-$C_8^{4\times1}$-P_2-FC_{172}	46,860
		A-GCWC	$C_8^{8\times1}$-P_2-$C_8^{4\times1}$-P_2-FC_{172} + $C_4^{2\times2}$-P_2-$C_8^{2\times2}$-P_2-FC_1	48,296
AVG	HW	CNN	$C_{16}^{8\times1}$-P_4-$C_{16}^{8\times1}$-P_2-FC_{24}	3,384
		DR	$C_{16}^{8\times1}$-$C_{16}^{8\times1}$-$C_{16}^{8\times1}$-$C_1^{8\times1}$	33,520
		GCWC	$C_{16}^{8\times1}$-P_4-$C_{16}^{8\times1}$-P_2-FC_{24}	3,768
		A-GCWC	$C_{16}^{8\times1}$-P_4-$C_{16}^{8\times1}$-P_2-FC_{24} + $C_4^{2\times2}$-P_2-$C_8^{2\times2}$-P_2-FC_1	4,728
	CI	CNN	$C_8^{8\times1}$-P_2-$C_8^{4\times1}$-P_2-FC_{172}	30,088
		DR	$C_{16}^{8\times1}$-$C_{16}^{8\times1}$-$C_{16}^{8\times1}$-$C_1^{8\times1}$	33,520
		GCWC	$C_8^{8\times1}$-P_2-$C_8^{4\times1}$-P_2-FC_{172}	44,536
		A-GCWC	$C_8^{8\times1}$-P_2-$C_8^{4\times1}$-P_2-FC_{172} + $C_4^{2\times2}$-P_2-$C_8^{2\times2}$-P_2-FC_1	45,972

Type	Data	Model	LR	Decay	Dropout	Regul
HIST	HW	CNN	3.5e-3	0.95	0.3	0.001
		DR	6.4e-3	0.92	0.0	0.004
		GCWC	0.999	0.17	0.001	
		A-GCWC	5.0e-4	0.98	0.2	0.045
	CI	CNN	0.97	0.12	0.001	
		DR	0.01	1.0	0.6	0.1
		GCWC	1.5e-3	0.99	0.13	0.002
		A-GCWC	1.0e-3	0.99	0.19	0.004
AVG	HW	CNN	3.5e-3	0.95	0.3	0.001
		DR	0.1	0.9	0.0	0.1
		GCWC	2.0e-4	0.999	0.17	0.001
		A-GCWC	5.0e-4	0.98	0.2	0.045
	CI	CNN	2.2e-3	0.97	0.12	0.001
		DR	0.013	0.9	0.6	0.1
		GCWC	1.5e-3	0.99	0.13	0.002
		A-GCWC	1.0e-3	0.99	0.19	0.004

Table A.3: Model Construction and Hyperparamter Selection.

(GCWC), where each training batch has 20 training instances, i.e., 20 input matrices. The average running time on HW for training is 19 ms (14 ms) per training batch for A-GCWC (GCWC), where each training batch has 20 training instances.

Baselines

We compare GCWC and A-GCWC with six baselines: (1) Historical Average (HA): for each edge, we use all the travel speed records on the edge in the training data to derive a histogram. The histogram is used as the estimated histogram in the testing phase. (2) GP: a Gaussian process regression model. (3) RF: a random forest regression model. (4) LSM [9]: the state-of-the-art for filling in missing weights in road networks using a latent space model. Note that GP, RF, and LSM are originally only able to fill in deterministic weights. To support filling in stochastic weights, we fill in values for each bucket separately. For example, in the setting of HIST-8, we consider 8 individual regression problems, where each regression is able to fill in values for each bucket. (5) CNN: classical convolutional neural network, whose hyper-parameters are set by following the same notations as for GCWC and A-GCWC, as shown in Table C.2. (6) DR: diffusion convolutional recurrent neural network [19], which is the state-of-the-art for predicting deterministic edge weights based on historical data.

Evaluation Metrics

We describe the metrics for evaluating the *estimation*, *prediction*, and *average* functions.

Estimation and Prediction: To assess the effectiveness of the models for estimation and prediction, we use the Mean Kullback-Leibler divergence Ratio (MKLR) and the Fraction of Likelihood Ratio (FLR) to measure the accuracy of the estimated (or predicted) stochastic weights $\widehat{W}$.

Specifically, MKLR is defined as follows.

$$MKLR = \frac{\sum_{i=1}^{T}\sum_{j=1}^{n} I_{ij} KL(w_{G_{j.}^{(i)}} || \widehat{w}_{j.}^{(i)})}{\sum_{i=1}^{T}\sum_{j=1}^{n} I_{ij} KL(w_{G_{j.}^{(i)}} || HA_{j.})}, \tag{A.11}$$

where T is the total number of testing time intervals, n is the total number of edges, and $I_{ij} \in [0,1]$, with $i \in [1,T]$ and $j \in [1,n]$, is an indicator of whether the stochastic weight of the j-th edge at the i-th time interval needs to be evaluated. In particular, we set set $I_{ij} = 0$, if in the i-th interval, edge e_j is not covered by traffic data; otherwise, we set $I_{ij} = 1$. Next $w_{G_{j.}^{(i)}}$ and $\widehat{w}_{j.}^{(i)}$ are the ground truth and estimated (or predicted) stochastic weights for the j-th edge at the i-th time interval, respectively.

Function $KL(\cdot||\cdot)$ computes the KL-divergence between two distributions, i.e., two stochastic weights represented as histograms. The lower a KL-divergence value is, the more similar the two stochastic weights are, indicating more accurate estimation or prediction. However, since KL-divergences range from 0 to ∞, it is hard to judge how small a KL-divergence value must be for an estimation or prediction to be considered accurate.

To solve this problem, we use $HA_{j\cdot}$ as a reference stochastic weight for the j-th edge, which is derived from all speed records in the training data that traversed the j-th edge, i.e., using the HA baseline. Here, we interpret $HA_{j\cdot}$ as the worst estimation or prediction of the stochastic weight for the j-th edge. Next, we derive a ratio *MKLR* between the KL-divergence of another method and the KL-divergence of HA. This ratio suggests how much a method can improve over HA. Lower *MKLR* values indicate higher improvements over HA.

We also measure the accuracy of estimated or predicted stochastic weights using FLR, defined as follows.

$$FLR = \frac{\sum_{i=1}^{T}\sum_{j=1}^{n} I_{ij}|LR_{ij} > 1|}{\sum_{i=1}^{T}\sum_{j=1}^{n} I_{ij}}, \tag{A.12}$$

where the meanings of T, n, and I_{ij} are the same as those in Equation (A.11), and LR_{ij} denotes the *likelihood ratio* [29] on the j-th edge in the i-th testing time interval. Specifically, $LR_{ij} = \frac{\sum_{k=1}^{N_{ij}} log(P_{\hat{w}}(o_k)+\epsilon)}{\sum_{k=1}^{N_{ij}} log(P_{HA}(o_k)+\epsilon)}$, where N_{ij} is the total number of ground truth speed records on the j-th edge in the i-th test time interval, o_k is the k-th ground truth speed record, and ϵ is a small value introduced to avoid zeros in *log* functions.

Given the j-th edge in the i-th testing time interval, we have an estimated or predicted stochastic weight $\hat{w}_{j\cdot}^{(i)}$ and the reference stochastic weight $HA_{j\cdot}$. We compute $P_{\hat{w}(o_k)}$ and $P_{HA}(o_k)$ as the likelihoods of observing o_k from the two distributions, i.e., $\hat{w}_{j\cdot}^{(i)}$ and $HA_{j\cdot}$. Here, if $LR_{ij} > 1$, the estimated (or predicted) weight $\hat{w}_{j\cdot}^{(i)}$ has a higher likelihood of observing the ground speed records than the reference weight $HA_{j\cdot}$, thus indicating a more accurate prediction of $\hat{w}_{j\cdot}^{(i)}$ than when using $HA_{j\cdot}$. We set $|LR_{ij} > 1|$ as 1 if the LR_{ij} value exceeds 1; otherwise, it is set to 0.

Average: We use Mean Absolute Percentage Error (MAPE) to measure the accuracy of the estimated average speeds.

$$MAPE = \frac{\sum_{i=1}^{T}\sum_{j=1}^{n} I_{ij}\frac{|y_{ij}-\hat{y}_{ij}|}{y_{ij}}}{\sum_{i=1}^{T}\sum_{j=1}^{n} I_{ij}} \times 100\%, \tag{A.13}$$

where T, n, and I_{ij} have the same meaning as before. Further, y_{ij} and $\hat{y}_{ij}$

represent the ground truth and estimated average speeds for the j-th edge in the i-th time interval, respectively.

A.6.2 Experimental Results

We proceed to cover experimental results using data sets CI and HW with different removal ratios *rm*. We compare our models for *estimation* and *prediction* with baseline approaches and report MKLR and FLR values for both data sets. We also compare our models for the *average* function with the baseline approaches and report MAPE values. Finally, we report on a study of the scalability of the proposed models.

Estimation

Tables A.4 and A.5 show MKLR values on HW and CI, respectively. As lower MKLR values indicate higher accuracy, we highlight the least MKLR values in each *rm* setting. A-GCWC has the best accuracy under all settings.

rm	GP	RF	LSM	CNN	DR	GCWC	A-GCWC
0.5	1.00	0.91	1.54	0.45	0.48	0.43	**0.43**
0.6	1.00	0.93	1.57	0.47	0.52	0.44	**0.43**
0.7	1.00	0.95	1.58	0.49	0.52	0.45	**0.44**
0.8	1.00	0.98	1.61	0.56	0.57	0.52	**0.46**

Table A.4: MKLR for the HW Dataset, Estimation.

rm	GP	RF	LSM	CNN	DR	GCWC	A-GCWC
0.5	1.00	0.96	1.08	0.55	0.85	**0.48**	**0.48**
0.6	1.00	0.97	1.17	0.59	0.68	0.50	**0.49**
0.7	1.00	0.98	1.26	0.58	0.55	0.50	**0.49**
0.8	1.00	0.99	1.35	0.66	0.61	**0.49**	**0.49**

Table A.5: MKLR for the CI Dataset, Estimation.

For both data sets, the MKLR values of all methods increase as the removal ratio *rm* increase. The reason is that as the *rm* value increase, fewer edges are covered with traffic data, and therefore more edges need to be assigned estimated stochastic weights, which is more challenging. We observe that LSM, the state-of-the-art method in weight completion, fails in the considered settings, since all MKLR values exceed 1, meaning that HA is better than LSM. This suggest that LSM cannot be extended to support stochastic weights and LSM cannot deal with the case where many edges lack traffic data.

The MKLR values reported for CNN vary significantly as *rm* increases, while those reported by GCWC and A-GCWC, especially A-GCWC, are rather

stable. This is because CNN is unable to capture the spatial correlations of a road network well, while GCWC and A-GCWC are.

DR achieves much better accuracy on HW than on CI—in particular, DR has MKLR as large as 0.85 at $rm = 0.5$ on CI. This indicates that although DR has good propagation abilities on small graphs, this ability is weakened on large graphs.

Finally, GCWC and A-GCWC show stable accuracy, and they double the accuracy of HA, i.e., the MKLR values of below 0.5 for both data sets. A-GCWC reports more stable MKLR values when *rm* values increase and higher accuracy, i.e., lower MKLR values, than GCWC.

Next, we report FLR values for each *rm* setting, as show in Tables A.6 and A.7. We highlight the highest FLR values for each *rm* setting since higher FLR values indicate higher accuracy.

For both data sets, we observe that the FLR values of all methods decrease as *rm* increases, which is because the tasks become more challenging. LSM also fails and reports FLR values less than 0.5, meaning that HA behaves better than LSM in most cases. In most settings, GCWC and A-GCWC can achieve the highest FLR values, and A-GCWC outperforms GCWC, which is consistent with the results reported by the MKLR values. CNN slightly outperforms our models given $rm = 0.5$ in HW data set. This is because for a small road network, e.g., for HW, and when many edges are covered with traffic data, e.g., $rm = 0.5$, CNN may be able to capture some latent correlations among the stochastic weights of edges. DR shows consistent trends—it achieves good accuracy on HW and performs significantly worse on CI.Overall, A-GCWC achieves the best FLR.

rm	GP	RF	LSM	CNN	DR	GCWC	A-GCWC
0.5	0.52	0.78	0.21	**0.90**	0.86	0.88	0.89
0.6	0.50	0.76	0.20	0.88	0.88	0.88	**0.92**
0.7	0.48	0.74	0.20	0.88	0.89	0.89	**0.91**
0.8	0.47	0.67	0.22	0.85	0.87	0.84	**0.88**

Table A.6: FLR for the HW Dataset, Estimation.

rm	GP	RF	LSM	CNN	DR	GCWC	A-GCWC
0.5	0.52	0.61	0.10	0.78	0.75	0.84	**0.85**
0.6	0.52	0.61	0.11	0.78	0.75	0.83	**0.84**
0.7	0.51	0.60	0.10	0.81	0.81	0.83	**0.84**
0.8	0.52	0.60	0.11	0.77	0.78	**0.85**	0.83

Table A.7: FLR for the CI Dataset, Estimation.

Prediction

Tables A.8 and A.9 show MKLR values for HW and CI, respectively. We highlight the least MKLR values given each *rm* setting.

Most observations for *estimation* also hold for *prediction*. On HW, as *rm* increases, the MKLR values of all methods, except GP and RF, follow an increasing trend. In contrast, on CI, the MKLR values do not follow an increase trend when *rm* increases. This is because CI is a larger, city network with more uncertainty and less dependency among different time intervals than in the case of HW, which is a highway toll gate network. Nevertheless, GCWC and A-GCWC outperform the other methods in almost all settings for both data sets, and A-GCWC behaves more stable and has lower MKLR values than does GCWC.

rm	GP	RF	LSM	CNN	DR	GCWC	A-GCWC
0.5	2.61	1.00	1.56	0.45	0.47	0.46	**0.43**
0.6	2.59	1.00	1.60	0.45	0.46	0.46	**0.44**
0.7	2.52	0.99	1.62	0.46	0.46	0.47	**0.43**
0.8	2.60	0.98	1.67	0.46	0.49	0.47	**0.45**

Table A.8: MKLR for the HW Dataset, Prediction.

rm	GP	RF	LSM	CNN	DR	GCWC	A-GCWC
0.5	1.09	0.97	4.18	0.50	0.48	0.45	**0.43**
0.6	1.12	0.97	4.15	0.50	0.49	**0.46**	**0.46**
0.7	1.16	0.98	4.16	0.54	0.54	0.49	**0.48**
0.8	1.24	0.98	4.30	0.59	0.53	0.50	**0.49**

Table A.9: MKLR for the CI Dataset, Prediction.

Next, we report FLR in Tables A.10 and A.11. In this setting, we use real speed observations during the testing intervals to compute the likelihood ratios. Recall that higher FLR values indicate higher accuracy.

rm	GP	RF	LSM	CNN	DR	GCWC	A-GCWC
0.5	0.29	0.55	0.23	0.93	0.93	**0.96**	0.92
0.6	0.31	0.55	0.22	0.93	0.93	**0.96**	0.92
0.7	0.32	0.58	0.23	0.93	**0.94**	0.93	0.93
0.8	0.35	0.59	0.24	0.93	**0.93**	**0.93**	0.92

Table A.10: FLR for the HW Dataset, Prediction.

We observe that LSM also fails, with all values being below 0.5. However, we cannot observe a clear trend with the increase of *rm* for both data sets. Further, we observe that GCWC and A-GCWC achieve the best results in most settings, the exception being when $rm = 0.7$ (Table A.10). Overall, the performance improvements of GCWC and A-GCWC over DR are less obvious

rm	GP	RF	LSM	CNN	DR	GCWC	A-GCWC
0.5	0.50	0.58	0.12	0.84	0.82	**0.86**	0.85
0.6	0.49	0.58	0.12	0.83	0.83	0.84	**0.85**
0.7	0.49	0.58	0.13	0.84	0.82	**0.84**	**0.84**
0.8	0.47	0.57	0.14	0.82	0.82	**0.84**	0.83

Table A.11: FLR for the CI Dataset, Prediction.

for prediction, because DR models temporal dependency explicitly by using RNNs, yielding more accurate predictions the edges with data. However, the overall accuracies of GCWC and A-GCWC are still better due to ability of these to propagate weights to edges without data.

We also observe that the FLR values for CNN, DR, GCWC, and A-GCWC are closer and higher for HW than those for CI, because speed observations for CI have more uncertainty than for HW, making predictions for CI more challenging.

Average

Tables A.12 and A.13 show MAPE when estimating average speeds for CI and HW with different *rm* values, respectively. In this setting, LSM is the state-of-the-art linear solution [9] and DR [19] can be regarded as the state-of-the-art non-linear solution.

We have the following observations. (1) A-GCWC performs overall best on both data sets; (2) LSM seems to not work on the CI dataset. The reason may be that the citywide road network is much more complex than the highway road network, meaning that linear modeling is unable to capture the latent attributes of this system. On the HW dataset, LSM does a much better job when the *rm* value is 0.5, compared to on the CI dataset. However, the performance falls markedly when the *rm* value increases, meaning that LSM falls short when many edges are not covered by traffic data. Note that in the original paper [9], LSM shows good accuracy when up to 30% of the edges lack average speeds, i.e., when $rm \leq 0.3$. (3) CNN and DR also performs better than LSM, suggesting that it the linear modeling correlations is the key problem. (4) While DR is the state-of-the-art when the data is not sparse, it performs worse than the proposed GCWC and A-GCWC when the data is sparse.

rm	LSM	CNN	DR	GCWC	A-GCWC
0.5	25.5%	12.9%	14.5%	13.0%	**12.8%**
0.6	28.5%	13.0%	13.5%	13.0%	**12.9%**
0.7	33.5%	13.0%	13.3%	**12.1%**	12.9%
0.8	48.3%	13.0%	13.5%	**12.4%**	12.9%

Table A.12: MAPE for the HW Dataset, Average.

rm	LSM	CNN	DR	GCWC	A-GCWC
0.5	31.0%	10.9%	11.3%	11.6%	**10.8%**
0.6	37.3%	**11.2%**	12.5%	12.2%	**11.2%**
0.7	44.7%	11.5%	13.6%	12.2%	**11.4%**
0.8	52.1%	13.0%	**11.5%**	12.1%	**11.5%**

Table A.13: MAPE for the CI Dataset, Average.

Scalability

We conduct this experiment to investigate the scalability of GCWC and A-GCWC on large road networks. Due to the unavailability of large road networks with sufficient amount of traffic data that cover most edges during most intervals, we manually enlarge the road network of CI by scales of 10, 20, 30, 40, and 50 such that the largest road network has a total number of $172 \times 50 = 8,600$ edges. Of course, the road network can be enlarged further, e.g., 60 or higher. However, we found that this is the largest road network that one single K80 GPU can deal with in our setting.

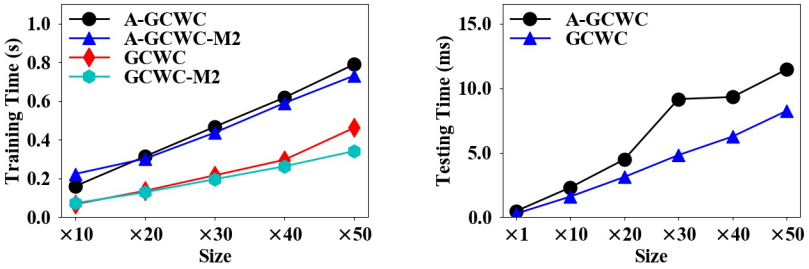
If the road network is too large to fit into a single machine, we can divide the network into smaller sub-networks and process them either in parallel on multiple computers [30, 31] or in sequence on a single computer. To simulate the case of a very large road network, we consider two settings: (1) processing a single road network using GCWC and A-GCWC; and (2) partitioning the road network into two small road networks and processing the two small road networks in sequence, denoted as GCWC-M2 and A-GCWC-M2.

Figure B.7(a) shows the average training time for one batch with batch size 20. This is the time it takes to finish an entire back propagation using 20 training instances, i.e., 20 input matrices. We observe that training A-GCWC takes more time than GCWC, which is reasonable since A-GCWC needs to train an extra CP-CNN. Further, if we partition a large road network into two small sub-networks and train them in sequence, this takes less time. However, this normally results in lower accuracy since the partitioning destroys some adjacency relationships in the original road network.

Figure B.7(b) shows that the average testing time for one instance, e.g., estimating or predicting a weight matrix $\widehat{W}$, is very fast (less than 15 ms), and there is little difference between A-GCWC and GCWC.

A.7 Conclusions and Outlook

High-resolution vehicle routing calls for road network models where each edge has a time-varying travel-time distribution. Even with massive volumes of vehicle data, it is generally impossible to construct distributions, also called stochastic weights, for all edges and times. We define and study the problem



(a) Training Time (b) Testing Time

Fig. A.6: Scalability.

of stochastic weight completion in this setting. A data-driven deep learning model, graph convolutional weight completion (GCWC), is proposed to fill in missing stochastic weights. In addition, an advanced GCWC model that takes into account contextual information is proposed to further improve accuracy. Empirical studies with two different, real datasets—highway loop detector data and citywide taxi GPS data—suggest that the proposed models are capable of outperforming other methods in all the experimental settings considered.

In future work, it is of interest to exploit temporal correlations to further improving the accuracy of the stochastic weights and to support continuous distribution models such as Gaussian mixture models. It is also of interest to integrate GCWC with existing routing algorithms [32, 33] to enhance routing quality.

Acknowledgments: This research was supported in part by a grant from the Obel Family Foundation, a grant from the Independent Research Fund Denmark, and by the DiCyPS center funded by Innovation Fund Denmark.

References

[1] S. Aljubayrin, B. Yang, C. S. Jensen, and R. Zhang, "Finding non-dominated paths in uncertain road networks," in *Proceedings of 24th ACM International Conference on Advances in Geographic Information Systems*, pp. 15:1–15:10, 2016.

[2] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, "Stochastic skyline route planning under time-varying uncertainty," in *Proceedings of the 30th IEEE International Conference of Data Engineering*, pp. 136–147, 2014.

[3] G. Andonov and B. Yang, "Stochastic shortest path finding in path-centric uncertain road networks," in *Proceedings of the 19th IEEE International Conference on Mobile Data Management*, pp. 40–45, 2018.

[4] J. Hu, B. Yang, C. Guo, and C. S. Jensen, "Risk-aware path selection with time-varying, uncertain travel costs: a time series approach," *The VLDB Journal–The International Journal on Very Large Data Bases J.*, vol. 27, no. 2, pp. 179–200, 2018.

[5] J. Hu, B. Yang, C. S. Jensen, and Y. Ma, "Enabling time-dependent uncertain eco-weights for road networks," *GeoInformatica*, vol. 21, no. 1, pp. 57–88, 2017.

[6] B. Yang, J. Dai, C. Guo, C. S. Jensen, and J. Hu, "PACE: a path-centric paradigm for stochastic path finding," *The VLDB Journal–The International Journal on Very Large Data Bases J.*, vol. 27, no. 2, pp. 153–178, 2018.

[7] C. Guo, C. S. Jensen, and B. Yang, "Towards total traffic awareness," *SIGMOD Record*, vol. 43, no. 3, pp. 18–23, 2014.

[8] Z. Ding, B. Yang, Y. Chi, and L. Guo, "Enabling smart transportation systems: A parallel spatio-temporal database approach," *IEEE Trans. Computers*, vol. 65, no. 5, pp. 1377–1391, 2016.

[9] D. Deng, C. Shahabi, U. Demiryurek, L. Zhu, R. Yu, and Y. Liu, "Latent space model for road networks to predict time-varying traffic," in *Proceedings of the 22th ACM International Conference on Knowledge Discovery and Data Mining*, pp. 1525–1534, 2016.

[10] J. Dai, B. Yang, C. Guo, C. S. Jensen, and J. Hu, "Path cost distribution estimation using trajectory data," *PVLDB*, vol. 10, no. 3, pp. 85–96, 2016.

[11] C. Guo, B. Yang, J. Hu, and C. S. Jensen, "Learning to route with sparse trajectory sets," in *Proceedings of the 34th IEEE International Conference of Data Engineering*, pp. 1073–1084, 2018.

[12] T. Idé and M. Sugiyama, "Trajectory regression on road networks," in *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pp. 203–208, 2011.

[13] J. Zheng and L. M. Ni, "Time-dependent trajectory regression on road networks via multi-task learning," in *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, pp. 1048–1055, 2013.

[14] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," *arXiv:1606.09375 [cs, stat]*, 2016. arXiv: 1606.09375.

[15] B. Yang, C. Guo, and C. S. Jensen, "Travel cost inference from sparse, spatio-temporally correlated time series using markov models," *PVLDB*, vol. 6, no. 9, pp. 769–780, 2013.

[16] Z. Ding, B. Yang, R. H. Güting, and Y. Li, "Network-matched trajectory-based moving-object database: Models and applications," *IEEE Trans. Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1918–1928, 2015.

[17] B. Yang, M. Kaul, and C. S. Jensen, "Using incomplete information for complete weight annotation of road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1267–1279, 2014.

[18] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, "Deep learning: A generic approach for extreme condition traffic forecasting," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 777–785, 2017.

[19] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proceedings of the 6th International Conference on Learning Representations*, *https://openreview.net/forum?id=SJiHXGWAZ*, 2018.

[20] R. Cirstea, D. Micu, G. Muresan, C. Guo, and B. Yang, "Correlated time series forecasting using multi-task deep neural networks," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 1527–1530, 2018.

[21] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu, "Multi-task representation learning for travel time estimation," in *Proceedings of the 24th ACM International Conference on Knowledge Discovery and Data Mining*, pp. 1695–1704, 2018.

[22] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Distinguishing trajectories from different drivers using incompletely labeled trajectories," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018.

[23] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Unsupervised and Transfer Learning@the 28th International Conference on Machine Learning*, pp. 37–50, 2012.

[24] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *19th IEEE International Conference on Mobile Data Management*, pp. 125–134, 2018.

[25] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral Networks and Locally Connected Networks on Graphs," *arXiv:1312.6203*, 2013.

[26] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv:1609.02907*, Sept. 2016.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of 27th Conference on Neural Information Processing Systems*, pp. 3111–3119, 2013.

[28] "Tensor flow api." `https://www.tensorflow.org/api_docs/python/tf/nn/embedding_lookup`.

[29] R. Waury, J. Hu, B. Yang, and C. S. Jensen, "Assessing the accuracy benefits of on-the-fly trajectory selection in fine-grained travel-time estimation," in *18th IEEE International Conference on Mobile Data Management*, pp. 240–245, 2017.

[30] B. Yang, Q. Ma, W. Qian, and A. Zhou, "TRUSTER: trajectory data processing on clusters," in *Proceedings of the 14th International Conference on Database Systems for Advanced Applications*, pp. 768–771, 2009.

[31] P. Yuan, C. Sha, X. Wang, B. Yang, A. Zhou, and S. Yang, "XML structural similarity search using mapreduce," in *Proceedings of the 11th International Conference on Web-Age Information Management*, pp. 169–181, 2010.

[32] H. Liu, C. Jin, B. Yang, and A. Zhou, "Finding top-k shortest paths with diversity," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 3, pp. 488–502, 2018.

[33] H. Liu, C. Jin, B. Yang, and A. Zhou, "Finding top-k optimal sequenced routes," in *Proceedings of the 34th IEEE International Conference of Data Engineering*, pp. 569–580, 2018.

Paper B

Risk-Aware Path Selection with Time-Varying, Uncertain Travel Costs—A Time Series Approach

Jilin Hu, Bin Yang, Chenjuan Guo, Christian S. Jensen

The paper has been published in the
The VLDB Journal—The International Journal on Very Large Data Bases,
27(2), pp. 179-200, 2018.

Abstract

We address the problem of choosing the best paths among a set of candidate paths between the same origin-destination pair. This functionality is used extensively when constructing origin-destination matrices in logistics and flex transportation. Because the cost of a path, e.g., travel time, varies over time and is uncertain, there is generally no single best path. We partition time into intervals and represent the cost of a path during an interval as a random variable, resulting in an uncertain time series for each path. When facing uncertainties, users generally have different risk preferences, e.g., risk-loving or risk-averse, and thus prefer different paths.

We develop techniques that, for each time interval, are able to find paths with non-dominated lowest costs while taking the users' risk preferences into account. We represent risk by means of utility function categories and show how the use of first-order and two kinds of second-order stochastic dominance relationships among random variables makes it possible to find all paths with non-dominated lowest costs. We report on empirical studies with large uncertain time series collections derived from a 2-year GPS data set. The study offers insight into the performance of the proposed techniques, and it indicates that the best techniques combine to offer an efficient and robust solution.

B.1 Introduction

Due to a combination of factors, including developments in autonomous vehicles, the emergence of mobility-on-demand, flex transportation, and increasing needs for more efficient logistics, it is a safe bet that path selection decisions will increasingly be made algorithmically while taking into account time-varying and uncertain travel costs of candidate paths [1].

For example, in logistics, exemplified by PostNord[1], and in flex transportation, exemplified by FlexDanmark[2], origin-destination matrices (*OD-matrices*) [2, 3] are often used to schedule trips. In particular, a city or a country is partitioned into N zones, e.g., according to administrative districts or simply using a uniform grid, yielding an $N \times N$ matrix where element (i, j) records the "best" path, along with its travel time, from zone i to zone j [4, 5].

To obtain an OD-matrix, we need to select the best path for each origin-destination pair. Given such a pair, the best path is often chosen from among a few candidate paths connecting the origin zone and the destination zone. As an example, we may choose among three paths, P_1, P_2, and P_3, that connect zone i to zone j, as shown in Figure B.1.

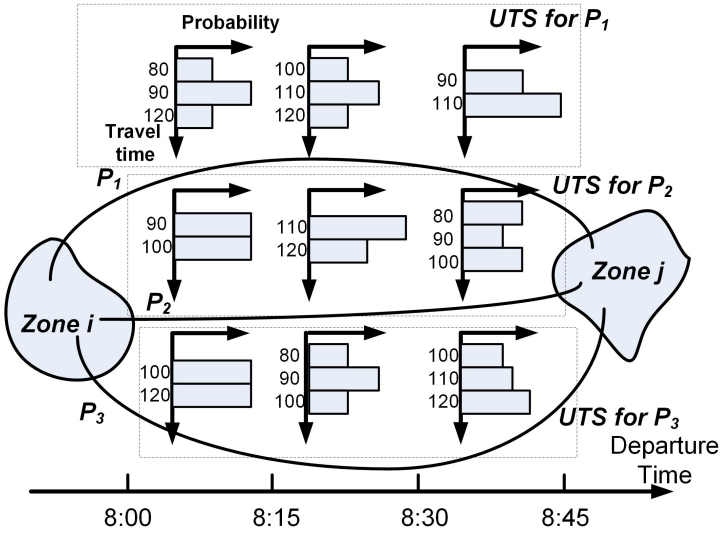


Fig. B.1: Motivating Example.

An intuitive strategy is to choose a path with the minimum average travel time. However, the use of average costs has shortcomings. Specifically, the travel time of a path varies across time, e.g., due to traffic and the weather; and even at a single point in time, different drivers may travel at different speeds, e.g., due to personal preferences and traffic lights. In other words, travel time is time varying [6] and

[1] `http://www.postnord.dk/`
[2] `https://www.flexdanmark.dk/`

uncertain [7]. Therefore, it is commonplace to model the travel time of a path as an uncertain time series (UTS) [8]: (1) time is first partitioned into fixed-length intervals, e.g., 15-min intervals; (2) a random variable is then recorded for each interval to capture the travel time distribution of using the path when the departure time falls into the interval.

Figure B.1 illustrates the UTSs of our three paths. The UTS of a path contains a random variable that represent the path's travel time distribution for each departure time interval, e.g., [8:00, 8:15), [8:15, 8:30), and [8:30, 8:45). A random variable is represented as a discrete distribution using a histogram in Figure B.1, but it can also be represented as a continuous distribution using, e.g., Gaussian mixture models [7]. In particular, the histogram modeling the travel time during [8:00, 8:15) for path P_1 indicates the total travel time of path P_1 may be 80, 90, and 120 minutes with probabilities 0.25, 0.5, and 0.25, respectively.

Since a logistics company generally needs to make deliveries at any time during a day and a flex transportation company generally receives requests for travel at any time during a day, it is useful to compute the best paths for all departure intervals. Thus, the problem becomes one of identifying, for each interval, the "best" path in the interval according to the travel-time distributions of the candidate paths, e.g., P_1, P_2, and P_3 for the example in Figure B.1.

When travel time is uncertain, there may be no unique best path. Table B.1 shows the travel times of the three paths for the departure time falls in the interval [8:00, 8:15). The first row states the above-mentioned probabilities of different travel times for path P_1.

Travel time (mins)	70	80	90	100	110	120
P_1	0	0.25	0.50	0	0	0.25
P_2	0	0	0.50	0.50	0	0
P_3	0	0	0	0.50	0	0.50

Table B.1: Uncertain Travel Times for P_1, P_2, and P_3, [8:00, 8:15).

Path P_1 has a "wide" distribution, implying that a delivery may arrive at the destination very early, but may also arrive very late. Path P_2 has a "narrow" distribution, thus offering a more predictable delivery time, but also no chance of an early delivery.

A key question is then which path to choose. Some users may prefer to run the risk of making a late delivery in order to have the possibility of making an early delivery, while other users may prefer a more predictable delivery time.

For example, one airport delivery may choose a so-called *risk-loving* path in order to connect with an early flight among several options, while another delivery may choose a *risk-averse* path in order to make sure to catch the last flight of the day. Further, there is evidence that emergency services such as ambulances may choose risk-loving paths [9, 10] and that transports involving perishables may prefer risk-averse paths [11]. It is important to provide integrated support for different risk preferences in a framework such as the one proposed in this paper.

In the example in Table B.1, path P_1 is risk-loving while path P_2 is risk-averse. A *risk-neutral* user may choose either P_1 or P_2. Path P_3 is not interesting to any user,

risk-loving, risk-averse, or risk-neutral, as it offers no benefits over paths P_1 and P_2.

The problem now becomes the following: given a set of candidate paths, a time horizon of interest, e.g., a day, a week, or a month, and a risk preference, e.g., risk-loving, risk-averse, or risk-neutral, identify, for each interval in the time horizon, the "best" path w.r.t. the given risk preference.

We show how different risk preferences can be modeled by means of utility function categories when assuming that the paths with the highest utility are preferred. Further, we provide a connection between stochastic dominance relationships and utility function categories. In short, having chosen a risk preference, and thus a utility function category, if one random variable dominates another random variable, this means that the former random variable must have at least the same expected utility as the latter, no matter which specific utility function in the category is considered. Thus, the path represented by the dominated random variable is uninteresting to any user whose risk preference is captured by the utility function category.

Based on the above, we provide a generic framework that aims to maximally help users with different risk preferences choose paths. In short, the framework compares the random variables of paths and returns exactly the paths with non-dominated random variables. When a user has a risk-loving or a risk-averse preference, thus having a convex or a concave utility function, we compare the random variables of paths using the notions of *second convex order* and *second concave order* stochastic dominance to compare random variables. Further, if a user has no clear preference, i.e., has a risk-neural preference, the user may use any monotonous utility function. In this case, we use *first order* stochastic dominance to compare the random variables of paths.

We provide all necessary techniques for checking first order, second convex order, and second concave order stochastic dominance between two random variables. These are then used as building blocks for techniques that enable the checking of dominance for each interval among different uncertain time series. Further, with the goal of improving efficiency, we present a grouping strategy that groups similar random variables from different intervals, and we provide an optimized group dominance checking technique. If one group dominates another group, all random variables in the former group dominate all random variables in the latter group. When domination between groups occurs, fewer random variables have to be compared.

To the best of our knowledge, this is the first study that provides a comprehensive analysis of the relationship between decision making under uncertainty with utility functions that represents different risk preferences, on the one hand, and different-order stochastic dominance, on the other hand; and it is the first study that enables temporal dominance checking among multiple uncertain time series. In particular, we make four contributions. First, we provide a comprehensive analysis of the relationships among risk preferences, utility functions, and stochastic dominance, and we formulate a novel temporal dominance query on uncertain time series. Second, we present a complete set of techniques for checking first-order, second convex order, and second concave order dominance between two random variables. Third, we present a general grouping strategy and techniques for checking dominance between random variable groups. Fourth, we report on empirical studies based on two large uncertain time series collections that indicate that the proposed techniques are efficient.

The remainder of the paper is organized as follows. Section 2 defines the setting

and formalizes the problem. Section 3 covers stochastic dominance relations. Section 4 and Section 5 detail the algorithms for checking dominance between random variables and between random variable groups, respectively. Section 6 reports on the empirical study. Related work is coved in Section 7, and Section 8 concludes.

B.2 Preliminaries

B.2.1 Uncertain Time Series

A *road network* is modeled as a graph where *vertices* represent road intersections or road ends and *edges* represent road segments. A *path* is a sequence of adjacent edges that represent connected road segments.

We model the travel time of a path as a *uncertain time series (UTS)*: the time horizon is partitioned into intervals, and each interval is associated with a random variable that represents the path's total travel time when the departure time belongs to the interval. We denote path P_i's UTS as

$$T_i = \langle X_i^{(1)}, X_i^{(2)}, \cdots, X_i^{(N)} \rangle,$$

where $X_i^{(j)}$ is the random variable associated with the j-th interval. The length of UTS T_i, denote as $|T_i|$, is the number of random variables, or equivalently, the number of intervals in the UTS, i.e., $|T_i| = N$.

Consider the example shown in Figure B.1, where three candidate paths exist that connect a source zone and a destination zone. Thus, we consider three UTSs, T_1, T_2, and T_3, representing the time-varying uncertain travel times of paths P_1, P_2, and P_3, respectively. Assuming that the time horizon is partitioned into 15-min intervals and we consider a time horizon of a week, each UTS $T_i = \langle X_i^{(1)}, X_i^{(2)}, \cdots, X_i^{(672)} \rangle$, $1 \leqslant i \leqslant 3$, has length 7*24*4 = 672, or equivalently, has 672 different departure time intervals. In particular, random variable $X_i^{(j)}$, $1 \leqslant j \leqslant 672$, represents the travel time distribution of path P_i when the departure time falls into the j-th interval.

The random variables in an UTS may be *dependent* and may also be *independent* of each other. To achieve a generic solution that cover both cases, the paper's solution do not make any assumptions on the dependency among the random variables in an UTS, e.g., the independence or Markov assumption. This has the effect that the proposed solution works both when the random variables are independent or dependent of each other.

B.2.2 Constructing UTSs from Trajectory Data

We use map-matched [12] GPS trajectories to derive UTSs for paths, as sketched below.

After map-matching, a GPS trajectory, $\langle r_1, r_2, \ldots, r_n \rangle$, is represented as a sequence of *trajectory records* that capture a trip made by a vehicle. A trajectory record $r_i = (t_i, e_i, m_i)$ indicates a trip on edge e_i that started at time t_i and took travel time m_i.

Given an interval I and an edge e, we are able to obtain a multiset of travel-time measurements $M_{I,e} = \{r_i.m_i | r_i.e_i = e \wedge r_i.t_i \in I\}$ from the trajectory records that

occurred on edge e during interval I. Next, a random variable that represents the travel-time distribution on edge e during interval I can be learned from the travel-time measurements in $M_{I,e}$ [7, 13]. For example, if $M_{[8:00,8:15),e_i} = \{80, 90, 120, 90, 120, 90, 80, 90\}$, we are able to learn a discrete random variable $\{(80, 0.25), (90, 0.50), (120, 0.25)\}$ that then represents the travel-time distribution for edge e_i when the departure time falls into interval [8:00, 8:15). It is also possible to learn a continuous random variable [7]. The techniques we propose later accommodate both discrete and continuous random variables.

Based on the above, for each edge and each interval, we are able to derive a random variable, thus obtaining a UTS for each edge. Next, given a path, we are able to construct a random variable for each interval by "summing up" the random variables of the edges in the path, thus obtaining a UTS for the path. Specifically, the travel time distribution of path $P_i = \langle e_1, e_2, \ldots, e_M \rangle$ during departure time interval I_j is given by $\odot_{k=1}^{M} RV(e_k, I_{e_k})$, where $\odot$ denotes convolution of two distributions and $RV(e_k, I_{e_k})$ denotes the travel time distribution of edge e_k during interval I_{e_k}, where I_{e_k} is the departure time interval on edge e_k, which may differ from the departure time interval of the path I_j and needs to be progressively updated according to the travel times of e_k's predecessor edges. The details of constructing random variables for paths are covered elsewhere [7, 14, 15].

B.2.3 Problem Definition

Recall the problem exemplified in the introduction. Given a set of candidate paths, a time horizon of interest, e.g., a day, a week, or a month, and a risk preference, e.g., risk-loving, risk-averse, or risk-neutral, we aim at identifying, for each interval in the time horizon of interest, the "best" path w.r.t. the given risk preference.

Since the travel time of a path is represented by UTS, we consider a UTS collection $TS = \{T_1, T_2, \ldots, T_k\}$, where each UTS $T_i = \langle X_i^{(1)}, X_i^{(2)}, \ldots, X_i^{(N)} \rangle$, $1 \leqslant i \leqslant k$, corresponds to a path. The time horizon of interest can be mapped to an interval range $R = [s, e]$ from the s-th interval to the e-th interval.

Next, let $RVS^{(j)} = \{X_1^{(j)}, X_2^{(j)}, \ldots, X_k^{(j)}\}$ denote the set of the random variables during the j-th interval in TS. Given a user risk preference, we aim at identifying, for each interval j, where $s \leqslant j \leqslant e$, the "optimal" random variables among the random variables in $RVS^{(j)}$ for the given risk preference. Thus, the path that corresponds to the optimal random variable is chosen as the best path for the j-th interval. For example, if X_2^5 is the optimal random variable, then P_2 is chosen as the best path in the 5-th interval in the OD-matrix.

Shortly, in Section B.3, we explain (1) how different user risk preferences can be connected to different-order stochastic dominance relationships and (2) how optimal random variables can be defined under a specific stochastic dominance relationship.

Based on the above, we study the *temporal dominance query* $Q(x, R, TS)$ that takes as input a stochastic dominance relationship x that corresponds to a user risk preference, an interval range $R = [s, e]$, and a UTS collection TS. The query returns a sequence of optimal random variable sets $q = \langle q^{(s)}, q^{(s+1)}, \ldots, q^{(e)} \rangle$, where $q^{(j)}$ is a set of optimal random variables w.r.t. the given stochastic dominance relationship x among all random variables in $RVS^{(j)}$, where $s \leqslant j \leqslant e$. A formal definition of the temporal

dominance query is given in Section B.3.3 after we define different orders of stochastic dominance.

Frequently used notation is listed in Table B.2.

Notation	Definition
X_i, X_j	Random variables
$E(X_i)$	The expectation of X_i
$X_i.min$	The minimum value in the range of X_i
$X_i.max$	The maximum value in the range of X_i
$u(a)$	A non-increasing utility function
$u'(a)$	The first derivative of function $u(a)$
$u''(a)$	The second derivative of function $u(a)$
$E_U(X_i)$	The expected utility of X_i
$f_{X_i}(x)$	The probability density function (pdf) of X_i
$F_{X_i}(x)$	The cumulative distribution function (cdf) of X_i
$\widehat{F}_{X_i}(x)$	$\int_0^x F_{X_i}(t)dt$, the integral of F_{X_i} from 0 to x
$\widetilde{F}_{X_i}(x)$	$\int_x^{+\infty} F_{X_i}(t)dt$, the integral of F_{X_i} from x to $+\infty$
$X_i \succ_{fsd} X_j$	X_i first order dominates X_j
$X_i \succ_{ssd} X_j$	X_i second convex order dominates X_j
$X_i \succ_{scsd} X_j$	X_i second concave order dominates X_j
TS	A collection of uncertain time series
T_i	An uncertain time series
RVS	A set of random variables
$RVS^{(j)}$	A set of the random variables in the j-th interval in TS
$O_{fsd}(RVS)$	The first order optimal random variable set of RVS
$O_{ssd}(RVS)$	The second convex order optimal random variable set of RVS
$O_{scsd}(RVS)$	The second concave order optimal random variable set of RVS

Table B.2: Notation.

B.3 Stochastic Optimality

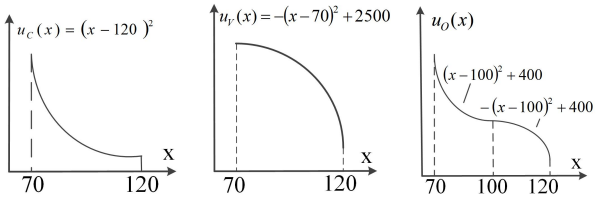
We explain how user risk preferences, utility functions, and stochastic dominance are connected and define the notion of optimal random variable.

B.3.1 Decision Making Under Uncertainty

Utility Functions

We model decision making as a utility maximization problem. Utility is computed by a utility function that takes *measurements* as input. Without loss of generality, we assume that smaller measurements (e.g., smaller travel times) are preferred. Thus, we consider *non-increasing* utility functions.

Following the example in Section B.1, we first consider a simple, deterministic case where the average travel time of paths P_1, P_2, and P_3 are 100, 110, and 120 minutes.



(a) Convex (b) Concave (c) Other

Fig. B.2: Categorization of Utility Functions.

Next we consider the utility function $u(x) = 120 - x$, where x is a path's travel time. Thus, the utilities of the three paths are 20, 10, and 0, respectively. Therefore, P_1 has the highest utility and is the optimal choice.

Expected Utility Principle

Next, to accommodate uncertain measurements, we apply the *expected utility principle* [16, 17]. An uncertain measurement is represented as a random variable X_i, and the expected utility of random variable X_i is employed as the utility of the uncertain measurement: $E_U(X_i) = \int_{X_i.min}^{X_i.max} u(x) \cdot f_{X_i}(x)dx$.

Given the utility function $u(x) = 120 - x$ and the uncertain travel times of P_1, P_2, and P_3 as shown in Table B.1, the expected utility of P_1, P_2, and P_3 can be calculated as follows: $E_U(P_1) = 0.25 \cdot (120 - 80) + 0.5 \cdot (120 - 90) + 0.25 \cdot (120 - 120) = 25$, $E_U(P_2) = 25$, and $E_U(P_3) = 10$. Therefore, P_1 or P_2 are optimal as both have the highest expected utility.

Risk Preferences and Utility Functions

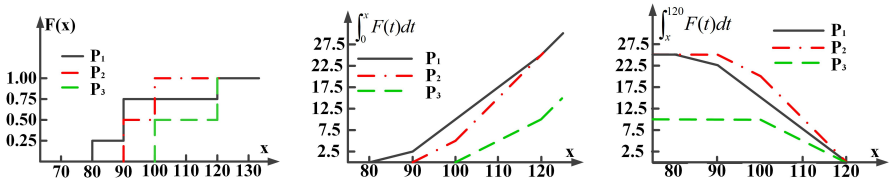
When making decisions under uncertainty, different users may have different preferences—*risk-loving*, *risk-averse* or *risk-neutral*. A risk-loving user, e.g., an ambulance that transports a cardiac arrest patient, prefers P_1 that offers the possibility of arriving very early although there is also a risk of arriving very late. A risk-averse user, e.g., a vehicle that transports perishables, prefers P_2 that guarantees arrival within 100 minutes. A risk-neutral user has no clear preference for P_1 versus P_2 and may choose either P_1 or P_2. However, path P_3 is not interesting to any user since it can neither make a user arrive very early nor guarantee arrival within 100 minutes.

Different risk preferences can be represented by different categories of utility functions. Specifically, we categorize utility functions into *concave*, *convex*, and *other* functions. Recall that we only consider *non-increasing* functions since smaller measurements (e.g., smaller travel times) are always preferred (see Section B.3.1). Figures B.2 (a), (b), and (c) show a convex function, a concave function, and a function that is neither convex nor concave.

Next, the relationships between risk preferences and utility function categories are shown in Table B.3. All users use non-increasing utility functions. The preferences of risk-loving and risk-averse users can be captured by convex and concave functions, respectively. The last column will be explained later in Section B.3.2.

Risk Attitudes	Utility Functions	Stochastic Dominance
Risk-neutral	Non-increasing	First order
Risk-loving	Non-incr., Convex	Second convex order
Risk-averse	Non-incr., Concave	Second concave order

Table B.3: Risk preferences, Utility Functions, and Stochastic Dominance



(a) cdf, $F_{X_i}(x)$ (b) Integral of cdf, $\widehat{F}_{X_i}(x)$ (c) Integral of cdf, $\widetilde{F}_{X_i}(x)$

Fig. B.3: Distributions of P_1, P_2, and P_3.

To further elaborate on the connection between risk preferences and utility functions, we consider the three functions in Figure B.2 and the three paths in Table B.1. The expected utilities when using the three utility functions are shown in Table B.4, where the highest expected utilities for each utility function are highlighted in bold.

	P_1	P_2	P_3	Optimal
u_C	**850**	650	200	P_1
u_V	1650	**1850**	800	P_2
u_O	**450**	**450**	200	P_1, P_2

Table B.4: Expected Utilities for Paths P_1, P_2, and P_3.

We have already seen that P_1 and P_2 are optimal for risk-loving and risk-averse users, respectively. Table B.4 shows that P_1 and P_2 have the largest expected utilities for the convex utility function $u_C(\cdot)$ and the concave utility function $u_V(\cdot)$, which is consistent with the relationship in Table B.3. Further, when a user has no clear risk-loving or risk-averse attitude, P_1 and P_2 may be optimal. However, P_3 is not of interest to any user.

B.3.2 Stochastic Dominance

Stochastic dominance enables comparison of two random variables. We introduce three different orders of stochastic dominance and describe how these notations of dominance apply to decision making with different categories of utility functions.

First Order Stochastic Dominance

Definition B.3.1
First Order Stochastic Dominance (FSD). Given two random variables X_1 and X_2, if $\forall a \in \mathbb{R}^+$, $F_{X_1}(a) \geqslant F_{X_2}(a)$, X_1 *first order stochastically dominates* X_2, denoted by $X_1 \succ_{fsd} X_2$.

Following the example in Table B.1, the cdfs of the three paths are shown in Figure B.3(a). Here, P_1's travel time random variable first order stochastically dominates that of P_3 since for every possible travel time $a \in [70, 120]$, P_1's cdf is no smaller than that of P_3. Similarly, P_2's travel time random variable dominates that of P_3. However, P_1 and P_2 do not dominate each other because P_1 has a larger cumulative probability before 100 minutes and P_2 has a larger cumulative probability after 100 minutes.

Theorem B.3.1
Given a non-increasing utility function $u(a)$, $a \in R^+$, if $X_1 \succ_{fsd} X_2$ then the expected utility of X_1 is no smaller than that of X_2, i.e., $E_U(X_1) \geqslant E_U(X_2)$.

Proof. We prove that $\Delta = E_U(X_1) - E_U(X_2) \geqslant 0$. For readability, we define $X_{max} = max(X_1.max, X_2.max)$, and $X_{min} = min(X_1.min, X_2.min) - \epsilon$, where $\epsilon > 0$. Then:

$$
\begin{aligned}
\Delta &= \int_{X_1.min}^{X_1.max} u(a) f_{X_1}(a) da - \int_{X_2.min}^{X_2.max} u(a) f_{X_2}(a) da \\
&= \int_{X_{min}}^{X_{max}} u(a) f_{X_1}(a) da - \int_{X_{min}}^{X_{max}} u(a) f_{X_2}(a) da \\
&= \int_{X_{min}}^{X_{max}} u(a) \cdot (f_{X_1}(a) - f_{X_2}(a)) da \\
&= \int_{X_{min}}^{X_{max}} u(a) d(F_{X_1}(a) - F_{X_2}(a)) \\
&= [u(a)(F_{X_1}(a) - F_{X_2}(a))]_{X_{min}}^{X_{max}} \\
&\quad - \int_{X_{min}}^{X_{max}} u'(a) \cdot (F_{X_1}(a) - F_{X_2}(a)) da \\
&= - \int_{X_{min}}^{X_{max}} u'(a) \cdot (F_{X_1}(a) - F_{X_2}(a)) da \qquad \text{(B.1)}
\end{aligned}
$$

From line 3 to line 6, we use *integration by parts* [18]. At X_{max}, we have $F_{X_1}(X_{max}) = 1$ and $F_{X_2}(X_{max}) = 1$; and at X_{min}, we have $F_{X_1}(X_{min}) = 0$ and $F_{X_2}(X_{min}) = 0$. Thus, we have $[u(a)(F_{X_1}(a) - F_{X_2}(a))]_{X_{min}}^{X_{max}} = 0$, which explains the final transformation.

Since X_1 first order stochastically dominates X_2, we have $F_{X_1}(a) \geqslant F_{X_2}(a)$ and thus $F_{X_1}(a) - F_{X_2}(a) \geqslant 0$. Since utility function $u(a)$ is non-increasing, we have $u'(a) \leqslant 0$. Thus, $\Delta \geqslant 0$.

Theorem B.3.1 implies that if a random variable X_1 first order stochastically dominates a random variable X_2, the object represented by X_2 is not interesting because the expected utility of X_1 is always no smaller than the expected utility of X_2. In our example, P_1 and P_2 may be optimal paths, but P_3 is uninteresting regardless of the utility function.

Second Order Stochastic Dominance

The second order stochastic dominance between two random variables is defined based on two different forms of integrals of the two random variables' cumulative distribution functions, $\widehat{F}_{X_i}(\cdot)$ and $\widetilde{F}_{X_i}(\cdot)$. In particular, $\widehat{F}_{X_i}(\cdot)$ denotes the integral of the cdf $F_{X_i}(\cdot)$ from 0, shown in Equation B.2.

$$\widehat{F}_{X_i}(a) = \int_0^a F_{X_i}(t)dt \tag{B.2}$$

And $\widetilde{F}_{X_i}(\cdot)$ denotes the integral of the cdf $F_{X_i}(\cdot)$ until $+\infty$, shown in Equation B.3.

$$\widetilde{F}_{X_i}(a) = \int_a^{+\infty} F_{X_i}(t)dt \tag{B.3}$$

In the two definitions that follow, we distinguish two cases—second convex order stochastic dominance, defined based on $\widehat{F}_{X_i}(\cdot)$, and second concave order stochastic dominance, defined based on $\widetilde{F}_{X_i}(\cdot)$.

Definition B.3.2
Second Convex Order Stochastic Dominance (SSD). Given random variables X_1 and X_2, if $\forall a \in \mathbb{R}^+$, $\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a)$, X_1 *second convex order stochastically dominates* X_2 , denoted by $X_1 \succ_{ssd} X_2$.

Figure B.3(b) shows the integrals of the cdfs from 0 for the three paths. We have $P_1 \succ_{ssd} P_2$, $P_2 \succ_{ssd} P_3$, and $P_1 \succ_{ssd} P_3$.

Theorem B.3.2
Given a non-increasing and convex utility function $u(a)$, $a \in R^+$, if $X_1 \succ_{ssd} X_2$ then $E_U(X_1) \geqslant E_U(X_2)$.

Proof. We prove $\Delta = E_U(X_1) - E_U(X_2) \geqslant 0$. According to Equation B.1,

$$\Delta = -\int_{X_{min}}^{X_{max}} u'(a) \cdot (F_{X_1}(a) - F_{X_2}(a))da$$

Next, we construct a helper function as follows.

$$I(a) = \int_{X_{min}}^{a} (F_{X_1}(t) - F_{X_2}(t))dt = \widehat{F}_{X_1}(a) - \widehat{F}_{X_2}(a)$$

Then, we have

$$\begin{aligned}
\Delta =& -\int_{X_{min}}^{X_{max}} u'(a)dI(a) \\
=& -u'(a)I(a)|_{X_{min}}^{X_{max}} + \int_{X_{min}}^{X_{max}} I(a)u''(a)da \\
=& -u'(X_{max})I(X_{max}) + u'(X_{min})I(X_{min}) \\
&+ \int_{X_{min}}^{X_{max}} I(a)u''(a)da
\end{aligned} \tag{B.4}$$

Consider the first term on the right hand side in Equation B.4. Since utility function u is non-increasing, $u'(X_{max}) \leqslant 0$. Because random variable X_1 second convex order stochastically dominates X_2, we have $\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a)$ and thus $I(X_{max}) = \widehat{F}_{X_1}(X_{max}) - \widehat{F}_{X_2}(X_{max}) \geqslant 0$. Thus, the first term is non-negative.

Next, as $I(X_{min}) = \widehat{F}_{X_1}(X_{min}) - \widehat{F}_{X_2}(X_{min}) = 0 - 0 = 0$, the second term is zero.

Finally, we use the property that the second derivative of a *convex* function is *non-negative*. Since utility function u is *convex*, $u''(a) \geqslant 0$. Since random variable X_1 second convex order stochastically dominates X_2, $\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a)$ and thus $I(a) = \widehat{F}_{X_1}(a) - \widehat{F}_{X_2}(a) \geqslant 0$. Thus, the third term is non-negative.

Theorem B.3.2 implies that when random variable X_1 second convex order stochastically dominates random variable X_2, and any risk-loving utility function, i.e., a convex function, is used, the choice represented by X_2 is not interesting because X_1's expected utility is always no smaller than that of X_2.

In our example, $P_1 \succ_{ssd} P_2$ and $P_1 \succ_{ssd} P_3$, leaving P_1 as the only optimal choice for a risk-loving users. Recall also that when a user has no clear risk-loving or risk-averse preference, both P_1 and P_2 can be optimal choices. This illustrates that when a user's preference becomes more specific, we can narrow down the optimal choices for the user using second convex order stochastic dominance.

Definition B.3.3
Second Concave Order Stochastic Dominance (SCSD). Given random variables X_1 and X_2, if $\forall a \in \mathbb{R}^+$, $\widetilde{F}_{X_1}(a) \geqslant \widetilde{F}_{X_2}(a)$, X_1 *second concave order stochastically dominates* X_2, denoted by $X_1 \succ_{scsd} X_2$.

Figure B.3(c) shows the integrals of the cdfs until $+\infty$ for the three paths in our example. According to Definition B.3.3, we have $P_2 \succ_{scsd} P_1$, $P_2 \succ_{scsd} P_3$, and $P_1 \succ_{scsd} P_3$.

Theorem B.3.3
Given a non-increasing and concave utility function $u(a)$, $a \in R^+$, if $X_1 \succ_{scsd} X_2$, then $E_U(X_1) \geqslant E_U(X_2)$.

Proof. We prove $\Delta = E_U(X_1) - E_U(X_2) \geqslant 0$. According to Equation B.1,

$$\Delta = -\int_{X_{min}}^{X_{max}} u'(a) \cdot (F_{X_1}(a) - F_{X_2}(a))da$$

Next, we construct a helper function as follows.

$$\begin{aligned} I(a) &= \widetilde{F}_{X_1}(a) - \widetilde{F}_{X_2}(a) \\ &= \int_a^{X_{max}} (F_{X_1}(t) - F_{X_2}(t))dt \\ &\quad + \int_{X_{max}}^{+\infty} (F_{X_1}(t) - F_{X_2}(t))dt \\ &= \int_a^{X_{max}} (F_{X_1}(t) - F_{X_2}(t))dt \end{aligned} \tag{B.5}$$

This holds because $F_{X_1}(t) = F_{X_2}(t) = 1$ if $t > X_{max}$.

Then, we have

$$\begin{aligned}\Delta &= \int_{X_{min}}^{X_{max}} u'(a)dI(a) \\ &= u'(a)I(a)|_{X_{min}}^{X_{max}} - \int_{X_{min}}^{X_{max}} I(a)u''(a)da \\ &= u'(X_{max})I(X_{max}) - u'(X_{min})I(X_{min}) \\ &\quad - \int_{X_{min}}^{X_{max}} I(a)u''(a)da \end{aligned} \tag{B.6}$$

Since $I(X_{max}) = 0$, the first term in Equation B.6 is 0.

Next, as utility function u is non-increasing, we have $u'(X_{min}) \leqslant 0$. Further, X_1 second concave order stochastically dominates X_2, so $\widetilde{F}_{X_1}(a) \geqslant \widetilde{F}_{X_2}(a)$ and thus $I(X_{min}) \geqslant 0$. Thus, the second term is non-negative.

Finally, we use the property that the second derivative of a *concave* function is *non-positive*. Because utility function u is *concave*, we have $u''(a) \leqslant 0$. Since X_1 second concave order stochastically dominates X_2, we have $\widetilde{F}_{X_1}(a) \geqslant \widetilde{F}_{X_2}(a)$ and thus $I(a) = \widetilde{F}_{X_1}(a) - \widetilde{F}_{X_2}(a) \geqslant 0$. As a result, the third term in Equation B.4 is non-negative.

Theorem B.3.3 implies that a user with a risk-averse utility function, i.e., a concave function, is not interested in choosing X_2 if $X_1 \succ_{scsd} X_2$, no matter the specific form of the concave function. In our example, P_2 is the optimal choice for risk-averse users.

Optimal Random Variable Set

Given a set of random variables *RVS*, if a random variable is not dominated by any other random variable, it is an optimal random variable, where dominance can be based on first order, second convex order, or second concave order dominance. For example, the first order optimal random variable set is defined as follows. $O_{fsd}(RVS) = \{X \in RVS | \nexists X' \in RVS(X' \succ_{fsd} X \wedge X' \neq X)\}$. The second convex and concave order optimal random variable sets $O_{ssd}(RVS)$ and $O_{scsd}(RVS)$ are defined similarly.

To help users make decisions, only the optimal random variables should be returned. In particular, if a user is risk-loving, only the optimal random variable set w.r.t. second convex order dominance, i.e., $O_{ssd}(RVS)$, should be returned; and if a user is risk-averse, only the optimal random variable set w.r.t. second concave order dominance, i.e., $O_{scsd}(RVS)$, should be returned; further, if a user is risk-neural, the optimal random variable set w.r.t. first order dominance, i.e., $O_{fsd}(RVS)$, should be returned. This explains the last column in Table B.3.

B.3.3 Temporal Dominance Query

Having defined the necessary concepts, we are able to define formally the *temporal dominance query* $Q(x, R, TS)$.

The query takes as input a stochastic dominance relationship parameter x, which can be *fsd*, *ssd*, or *scsd*; a interval range $R = [s, e]$ that represent a time horizon of interest, and a UTS collection $TS = \{T_1, T_2, \ldots, T_k\}$, where each UTS $T_i = \langle X_i^{(1)}, X_i^{(2)}, \ldots, X_i^{(N)} \rangle$.

The query returns a sequence of optimal random variable sets $q = \langle q^{(s)}, q^{(s+1)}, \ldots, q^{(e)} \rangle$, where $q^{(j)} = O_x(RVS^{(j)})$ and $RVS^{(j)} = \{X_1^{(j)}, X_2^{(j)}, \ldots, X_k^{(j)}\}$ denotes the set of the random variables during the j-th interval in TS where $s \leqslant j \leqslant e$, and x is *fsd*, *ssd*, or *scsd*.

Temporal dominance queries provide a generic solution to users with varying risk preferences in the sense that:
(1) if a user does not have a clear risk preference, *fsd* should be used, and the user can choose any object in $O_{fsd}(RVS)$;
(2) if a user has either a risk-loving or risk-averse preference, *ssd* or *scsd* should be used, and the user can choose any object in $O_{ssd}(RVS)$ or $O_{scsd}(RVS)$;
(3) if a user provides a specific utility function, we first categorize it as convex, concave, or other; then we only need to compute expected utilities for the objects in $O_{fsd}(RVS)$, $O_{ssd}(RVS)$, or $O_{scsd}(RVS)$, not for other dominated objects, and finally, and return the object with the highest expected utility.

B.4 Checking Stochastic Dominance

To efficiently compute temporal dominance queries on UTSs, we first need efficient means of checking stochastic dominance between two random variables. In this section, we present algorithms for checking these three kinds of dominance considered between two random variables.

For a random variable X, we denote the expectation of the random variable as $E(X)$ and the minimum and maximum value of the random variable as $X.min$ and $X.max$, respectively. Next, we introduce $X.min^- = X.min - \rho$ where ρ is a small positive value. When considering travel time, ρ represents a time that is shorter than the finest time granularity ϵ. For example, if the finest time granularity ϵ is a second, ρ can be half second or a millisecond.

Recall that we use $F_X(\cdot)$ to denote the cdf of random variable X. Thus, we have $F_X(X.min^-) = 0$; $\forall X.min < x < X.max$, $0 < F_X(x) < 1$; and $\forall x \geqslant X.max$, $F_X(x) = 1$.

B.4.1 Checking FSD

We first consider a naive algorithm based on the definition of FSD. Next, we propose an initial check strategy to improve the naive algorithm. Finally, we propose a speedup algorithm.

Naive algorithm: Given two random variables X_1 and X_2, a naive algorithm for checking whether X_1 first order dominates X_2 is to use Definition B.3.1. For each value $a \in Dom$ where $Dom = [\min(X_1.min, X_2.min), \max(X_1.max, X_2.max)]$, we check whether $F_{X_1}(a) \geqslant F_{X_2}(a)$ always holds. If yes, X_1 first order dominates X_2. Otherwise, X_1 does not first order dominate X_2.

Assume that the finest granularity of travel-time measurements is ϵ, e.g., 1 second or 1 minute. We have $N = \frac{Dom}{\epsilon}$ possible values to check. Thus, the complexity of the naive algorithm is linear in N, i.e., the number of *comparisons* between two random variables' CDFs. The pseudo code of the naive algorithm is shown in Algorithm 1.

Algorithm 1 NaiveFSD

Input:
Random variables: X_1 and X_2;
Output:
Dominance relationship between X_1 and X_2 w.r.t. FSD;
1: **for** each $a \in [\min(X_1.min, X_2.min), \max(X_1.max, X_2.max)]$ **do**
2: **if** $F_{X_1}(a) < F_{X_2}(a)$ **then**
3: **return** X_1 does not dominate X_2;
4: **end if**
5: **end for**
6: **return** X_1 dominates X_2;

Naive algorithm with initial check: The naive algorithms is inefficient as it needs to check every $a \in [\min(X_1.min, X_2.min), \max(X_1.max, X_2.max)]$. We present an improved algorithm with an initial check. If X_1 and X_2 fail this check, X_1 does not dominate X_2. The initial check is based on Lemma B.4.1.

Lemma B.4.1
Given two random variables X_1 and X_2, if $X_1 \succ_{fsd} X_2$ then $X_1.min \leqslant X_2.min$ and $X_1.max \leqslant X_2.max$, and $E(X_1) \leqslant E(X_2)$.

Proof. We prove by contradiction that $X_1.min \leqslant X_2.min$ and $X_1.max \leqslant X_2.max$.

First, assume that $X_1.min > X_2.min$. Based on the assumption and the definition of $X.min^-$, we get $X_1.min > X_1.min^- > X_2.min$. Consequently, we have $F_{X_2}(X_1.min^-) > 0 = F_{X_1}(X_1.min^-)$. However, since $X_1 \succ_{fsd} X_2$, we have $F_{X_1}(x) \geqslant F_{X_2}(x)$ for any $x \in (-\infty, +\infty)$, including the case when $x = X_1.min^-$. This results in a contradiction. Thus, the assumption is invalid, and we must have $X_1.min \leqslant X_2.min$.

Second, assuming that $X_1.max > X_2.max$, we have that $F_{X_1}(X_2.max) < 1$ and $F_{X_2}(X_2.max) = 1$. Thus, $F_{X_1}(X_2.max) < F_{X_2}(X_2.max)$. However, since $X_1 \succ_{fsd} X_2$, we have $F_{X_1}(X_2.max) \geqslant F_{X_2}(X_2.max)$, which also results in a contradiction. Thus, the assumption is invalid, and we must have $X_1.max \leqslant X_2.max$.

Finally, we apply Theorem B.3.1 to prove $E(X_1) < E(X_2)$. Consider a non-increasing utility function $u(x) = -x$. According to the Theorem B.3.1, we have $E_U(X_1) \geqslant E_U(X_2)$. Since $u(x) = -x$, we have $E_U(X_1) = -E(X_1)$ and $E_U(X_2) = -E(X_2)$. Thus, we have $E(X_1) \leqslant E(X_2)$.

Based on Lemma B.4.1, we first check if the three conditions, i.e., $X_1.min \leqslant X_2.min$, $X_1.max \leqslant X_2.max$, and $E(X_1) \leqslant E(X_2)$, hold. If yes, X_1 may stochastically dominate X_2, and then we call the naive algorithm to further check if X_1 dominates X_2; otherwise, X_1 does not dominate X_2. Hence, we propose an *Initial Check* method before calling the naive FSD algorithm, which is shown in Algorithm 2.

Speedup Algorithm: We propose a speed-up algorithm to verify the first order stochastic dominance relationship between two random variables, which can significantly reduce the number of comparisons. We first apply the initial check specified in Lemma B.4.1 to filter the cases where two random variables cannot dominate each

Algorithm 2 NaiveFSDInitialCheck

Input:
Random variables: X_1 and X_2;
Output:
Dominance relationship between X_1 and X_2 w.r.t. FSD;
1: **if** $E(X_1) \leqslant E(X_2) \wedge X_1.min \leqslant X_2.min \wedge X_1.max \leqslant X_2.max$ **then**
2: Call the naive FSD algorithm on X_1 and X_2;
3: **else if** $E(X_2) \leqslant E(X_1) \wedge X_2.min \leqslant X_1.min \wedge X_2.max \leqslant X_1.max$ **then**
4: Call the naive FSD algorithm on X_2 and X_1;
5: **else**
6: **return** no dominance;
7: **end if**

other. We then assume that $X_1.min \leqslant X_2.min$ and $X_1.max \leqslant X_2.max$. Then, the speed-up algorithm needs to check whether $F_{X_1}(a) \geqslant F_{X_2}(a)$ for each $a \in [X_2.min, X_1.max]$.

The speed-up algorithm does not only rely on the two random variables' cdfs, i.e., $F_{X_1}(\cdot)$ and $F_{X_2}(\cdot)$, but also on the integral of cdfs, i.e., $\widehat{F}_{X_1}(\cdot)$ and $\widehat{F}_{X_2}(\cdot)$. This means that $\widehat{F}'_{X_1}(a) = F_{X_1}(a)$ and $\widehat{F}'_{X_2}(a) = F_{X_2}(a)$ for any a. Figure B.4(a) shows an example of the integrals of the cdfs of two random variables X_1 and X_2. Recall that at a point on the curve of an integral of a cdf, the slope of the point is the point's corresponding cumulative distribution.

The speed up algorithm $SpeedupFSD(X_1, X_2, s, e)$ follows a divide-and-conquer approach. It takes as input two random variables X_1 and X_2 and a range $[s, e]$, and it returns a Boolean value indicating whether $F_{X_1}(x) \geqslant F_{X_2}(x)$ for any $x \in [s, e]$. The pseudo code is shown in Algorithm 3. First, we call $SpeedupFSD(X_1, X_2, X_2.min, X_1.max)$.

Algorithm 3 first checks whether $F_{X_1}(x) \geqslant F_{X_2}(x)$ when x equals to the two boundary values s and e, i.e., whether $F_{X_1}(s) \geqslant F_{X_2}(s)$ and $F_{X_1}(e) \geqslant F_{X_2}(e)$ (lines 1–2). If not, X_1 cannot dominate X_2, and the algorithm returns *False*. Next, algorithm 3 uses Lemma B.4.2 to check whether the dominance relationship between X_1 and X_2 can be identified by using their cdfs at s and e (lines 4–5).

Lemma B.4.2
Given random variables X_1 and X_2 and a range $[s, e]$, if $F_{X_1}(s) \geqslant F_{X_2}(e)$ then $F_{X_1}(x) \geqslant F_{X_2}(x)\ \forall x \in [s, e]$.

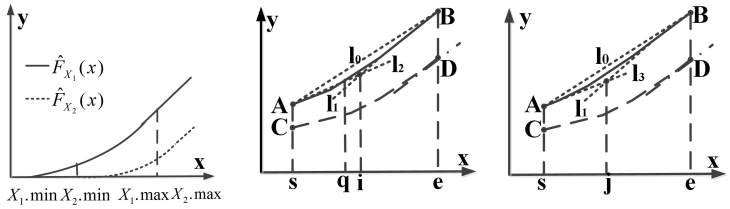
Proof. Since $F_X(x)$ is non-decreasing, we have $F_X(s) \leqslant F_X(x) \leqslant F_X(e), \forall x \in [s, e]$. Thus, we obtain $F_{X_1}(x) \geqslant F_{X_1}(s) \geqslant F_{X_2}(e) \geqslant F_{X_2}(x), \forall x \in [s, e]$.

To better explain the next lines in Algorithm 3, we introduce four important points: $A = (s, \widehat{F}_{X_1}(s))$, $B = (e, \widehat{F}_{X_1}(e))$, $C = (s, \widehat{F}_{X_2}(s))$, and $D = (e, \widehat{F}_{X_2}(e))$, where points A and C are the left endpoints of the curves of $\widehat{F}_{X_1}$ and $\widehat{F}_{X_2}$ in the range $[s, e]$, and points B and D are the right endpoints. Examples of the four points are shown in Figures B.4(a) and (b).

Next, we construct a line segment l_0 by connecting A and B, and we compute the slope k_{AB} of line segment l_0. At point B, we compute the corresponding slope

Algorithm 3 SpeedupFSD (X_1, X_2, s, e)

Input:
Random variables X_1 and X_2; a range $[s, e]$
Output:
True if $X_1 \succ_{fsd} X_2$; *False*, otherwise.
if $F_{X_1}(s) < F_{X_2}(s)$ or $F_{X_1}(e) < F_{X_2}(e)$ **then**
  return *False*;
end if
if $F_{X_1}(s) \geqslant F_{X_2}(e)$ **then**
  return *True*; //acc. to Lemma B.4.2
end if
$k_{AB} \leftarrow$ the slope of the line segment AB;
$k_A \leftarrow F_{X_1}(s)$; $k_B \leftarrow F_{X_1}(e)$; $k_D \leftarrow F_{X_2}(e)$;
Construct l_1 through point B with slope k_B;
if $k_D < k_{AB}$ **then**
  Construct l_2 through point A with slope k_D;
  $i \leftarrow x$-coordinate of the intersection between l_1 and l_2;
  return SpeedupFSD(X_1, X_2, s, i) //acc. to Lemma B.4.3
else
  Construct l_3 from point A with slope k_A;
  $j \leftarrow x$-coordinate of the intersection between l_1 and l_3;
  Boolean $b1 \leftarrow$ SpeedupFSD(X_1, X_2, s, j);
  Boolean $b2 \leftarrow$ SpeedupFSD(X_1, X_2, j, e);
  return $b1 \wedge b2$;
end if



(a) $\widehat{F}_{X_1}(x)$ and $\widehat{F}_{X_2}(x)$ (b) Case 1 (c) Case 2

Fig. B.4: Speedup Algorithm for Checking FSD.

k_B on the curve of $\widehat{F}_{X_1}$, which equals the cumulative distribution of e on X_1, i.e., $k_B = \widehat{F}'_{X_1} = F_{X_1}(e)$. We have $k_B \geqslant k_{AB}$. We construct a straight line l_1 that goes through point B with slope k_B. At point D, we compute the corresponding slope k_D on the curve of $\widehat{F}_{X_2}$, which equals the cumulative distribution of e on X_2, i.e., $k_D = \widehat{F}'_{X_2} = F_{X_2}(e)$. Then we distinguish two cases depending on the ordering of k_D and k_{AB}.

Case 1 (lines 10–13): $k_D < k_{AB}$ (see Figure B.4(b)). We make a straight line l_2 through point A with slope k_D. Since $k_D < k_{AB}$, l_2 and l_1 must intersect, and the x-coordinate of the intersection i must be between s and e.

According to Lemma B.4.3, we have $F_{X_1}(x) > F_{X_2}(x), \forall x \in [i, e]$. Therefore, range $[i, e]$ can be pruned safely and we only need to check if $F_{X_1}(x) \geqslant F_{X_2}(x), \forall x \in [s, i]$. To do this, we recursively call *SpeedupFSD*(X_1, X_2, s, i).

Lemma B.4.3
If $k_D < k_{AB}$, then $F_{X_1}(x) \geqslant F_{X_2}(x), \forall x \in [i, e]$.

Proof. Let the x-coordinate of the intersection between l_2 and $\widehat{F}_{X_1}(x)$ be q (see Figure B.4(b)). Further, we have $\widehat{F}'_{X_1}(q) = F_{X_1}(q) = k_D$. Since $\widehat{F}_{X_1}(x)$ is convex and increasing, we have (1) $F_{X_1}(a) \geqslant k_D$ if $a \geqslant q$ and (2) $q \leqslant i$. Based on the above, we have $\forall x \in [i, e], F_{X_1}(x) \geqslant F_{X_1}(q) = k_D = F_{X_2}(e) \geqslant F_{X_2}(x)$.

Case 2 (lines 14–19): $k_D \geqslant k_{AB}$ (see Figure B.4(c)). We make a straight line l_3 through point A with slope $k_A = F_{X_1}(s)$. Since k_A must be no larger than k_{AB}, l_3 and l_1 must intersect, and the x-coordinate of the intersection j must be between s and e. In this case, we cannot guarantee $F_{X_1}(a) \geqslant F_{X_2}(a)$ for neither sub-range $[s, j]$ nor sub-range $[j, e]$. Therefore, we recursively call *SpeedupFSD*(X_1, X_2, s, j) and *SpeedupFSD*(X_1, X_2, j, e) to check both sub-ranges.

B.4.2 Checking SSD

We present a naive algorithm, a naive algorithm with an initial check, and a speedup algorithm to check second convex order stochastic dominance (SSD) between two random variables.

Naive Algorithm for SSD: Similar to the naive algorithm for FSD checking, the naive algorithm for SSD checking applies the definition of SSD, i.e., Definition B.3.2. For each value $a \in [\min(X_1.min, X_2.min), \max(X_1.max, X_2.max)]$, we check whether

$\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a)$ always holds. If so, X_1 dominates X_2; otherwise, X_1 does not dominate X_2. The pseudo code is shown as follows.

Algorithm 4 NaiveSSD

Input:
Random variables: X_1 and X_2;
Output:
Dominance relationship between X_1 and X_2 w.r.t. SSD;
1: **for** each $a \in [\min(X_1.min, X_2.min), \max(X_i.max, X_i.max)]$ **do**
2: **if** $\widehat{F}_{X_1}(a) < \widehat{F}_{X_2}(a)$ **then**
3: **return** X_1 does not dominate X_2;
4: **end if**
5: **end for**
6: **return** X_1 dominates X_2;

Naive Algorithm with Initial check for SSD: We present an initial check based on Lemma B.4.4, which is similar to the initial check of FSD. If random variables X_1 and X_2 do not pass the check, X_1 does not dominate X_2 w.r.t. SSD. The proof of Lemma B.4.4 and the pseudo code of Algorithm 5 with the initial check according to Lemma B.4.4 follow.

Lemma B.4.4
If $X_1 \succ_{ssd} X_2$ then $X_1.min \leqslant X_2.min$ and $E(X_1) \leqslant E(X_2)$.

Proof. We prove $X_1.min \leqslant X_2.min$ by contradiction. If $X_1.min > X_2.min$, we have $\widehat{F}_{X_1}(X_1.min) = 0$ and $\widehat{F}_{X_2}(X_1.min) > 0$. Thus, we have $\widehat{F}_{X_2}(X_1.min) > \widehat{F}_{X_1}(X_1.min)$. According to the definition of SSD, when $X_1 \succ_{ssd} X_2$, we have $\widehat{F}_{X_1}(X_1.min) \geqslant \widehat{F}_{X_2}(X_1.min)$. This yields a contradiction, and we must have $X_1.min \leqslant X_2.min$.

We apply Theorem B.3.2 to prove $E(X_1) \leqslant E(X_2)$. Consider a non-increasing convex utility function $u(x) = -x$. According to Theorem B.3.2, we have $E_U(X_1) \geqslant E_U(X_2)$. Since $u(x) = -x$, we have $E_U(X_1) = -E(X_1)$ and $E_U(X_2) = -E(X_2)$. Thus, $E(X_1) \leqslant E(X_2)$.

Speedup Algorithm for SSD: We propose a speed-up algorithm for SSD checking between two random variables. We first utilize the initial check specified in Lemma B.4.4 to filter cases where two random variables cannot dominate each other. We can then assume that $X_1.min \leqslant X_2.min$, and the algorithm needs to check if X_1 dominates X_2 w.r.t. SSD, i.e., whether $\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a)$, $\forall a \in [X_2.min, \max(X_1.max, X_2.max)]$.

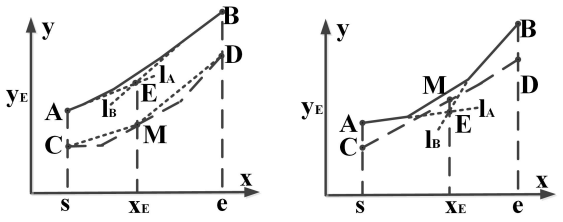
The speed-up algorithm also employs the two random variables' cdfs, i.e., $F_{X_1}(\cdot)$ and $F_{X_2}(\cdot)$, and the integrals of the cdfs, i.e., $\widehat{F}_{X_1}(\cdot)$ and $\widehat{F}_{X_2}(\cdot)$.

Algorithm $SpeedupSSD(X_1, X_2, s, e)$ follows a divide-and-conquer approach. It takes as input two random variables X_1 and X_2 and a range $[s, e]$, and it returns a Boolean value indicating whether $\widehat{F}_{X_1}(a) \geqslant \widehat{F}_{X_2}(a)$ for each $a \in [s, e]$. The pseudo code is shown in Algorithm 6. First, we call $SpeedupSSD$ $(X_1, X_2, X_2.min, \max(X_1.max, X_2.max))$.

Algorithm 5 NaiveSSDInitialCheck

Input:
Random variables X_1 and X_2;
Output:
Dominance relationship between X_1 and X_2 w.r.t. SSD;
1: **if** $E(X_1) \leqslant E(X_2) \wedge X_1.min \leqslant X_2.min$ **then**
2: Call the naive SSD algorithm on X_1 and X_2;
3: **else if** $E(X_2) \leqslant E(X_1) \wedge X_2.min \leqslant X_1.min$ **then**
4: Call the naive SSD algorithm on X_2 and X_1;
5: **else**
6: **return** no dominance.
7: **end if**

As for SSD checking, we make use of the four important points, where points A and C are the left endpoints and points B and D are the right endpoints of the curves of $\widehat{F}_{X_1}$ and $\widehat{F}_{X_2}$ in the range $[s, e]$. Examples of the four points are shown in Figures B.5 (a) and (b).



(a) Case 1 (b) Case 2

Fig. B.5: Speedup Algorithm for Checking SSD.

At point A, we compute the corresponding slope k_A on the curve of $\widehat{F}_{X_1}$, which equals the cumulative distribution of s on X_1, i.e., $k_A = \widehat{F}'_{X_1}(s) = F_{X_1}(s)$. Then we construct a line l_A through A with slope k_A. Next, at point B, we compute the corresponding slope k_B on the curve of $\widehat{F}_{X_1}$, which equals the cumulative distribution of e on X_1, i.e., $k_B = \widehat{F}'_{X_1}(e) = F_{X_1}(e)$. Then we construct a line l_B through B with slope k_B. Assume that l_A and l_B intersect at point $E = (x_E, y_E)$ where the x- and y-coordinates of point E are x_E and y_E, respectively. We denote the point on curve of $\widehat{F}_{X_2}$ whose x-coordinate is x_E as $M = (x_E, \widehat{F}_{X_2}(x_E))$.

We proceed to consider two cases.

Case 1: *Point E is not below point M* (see Figure B.5(a)). According to Lemma B.4.5, we have $\widehat{F}_{X_1}(x) \geqslant \widehat{F}_{X_2}(x), \forall x \in [s, e]$. Thus, we return *True*.

Lemma B.4.5
If $y_E \geqslant \widehat{F}_{X_2}(x_E)$ then $\widehat{F}_{X_1}(x) \geqslant \widehat{F}_{X_2}(x), \forall x \in [s, e]$.

Proof. Since $\widehat{F}_{X_1}(x)$ and $\widehat{F}_{X_2}(x)$ are convex, the tangent line l_A at point A of $\widehat{F}_{X_1}(x)$ satisfies $\widehat{F}_{X_1}(x) \geqslant l_A(x), \forall x \in [s, x_E]$; and the chord line l_{CM} that connects points C

Algorithm 6 SpeedupSSD (X_1, X_2, s, e)

Input:
Random variables X_1 and X_2; a range $[s, e]$;
Output:
True, if $X_1 \succ_{dcx} X_2$; *False*, otherwise;
if $\widehat{F}_{X_1}(s) < \widehat{F}_{X_2}(s) \vee \widehat{F}_{X_1}(e) < \widehat{F}_{X_2}(e)$ **then**
  return *False*
end if
if $\widehat{F}_{X_1}(s) \geqslant \widehat{F}_{X_2}(e)$ **then**
  return *True*; //acc. to Lemma B.4.4
end if
$k_A \leftarrow F_{X_1}(s)$; $k_B \leftarrow F_{X_1}(e)$;
Construct line l_A through A with the slope k_A;
Construct line l_B through B with the slope k_B;
Point $E \leftarrow$ intersection of lines l_A and l_B;
Point $M \leftarrow$ the point on curve of $\widehat{F}_{X_2}$ whose x-coordinate is x_E;
if E is not below M **then**
  return *True*; //acc. to Lemma B.4.5
else
  $b1 \leftarrow$ SpeedupSSD(X_1, X_2, s, x_E);
  $b2 \leftarrow$ SpeedupSSD(X_1, X_2, x_E, e);
  return $b1 \wedge b2$;
end if

and M of $\widehat{F}_{X_2}(x)$ satisfies $l_{CM}(x) \geqslant \widehat{F}_{X_2}(x), \forall x \in [s, x_E]$.

Moreover, we have $l_A(s) = \widehat{F}_{X_1}(s) \geqslant \widehat{F}_{X_2}(s) = l_{CM}(s)$, meaning that the left end point of l_A is not below the left end point of l_{CM}; and $l_A(x_E) = y_E \geqslant \widehat{F}_{X_2}(x_E) = l_{CM}(x_E)$, meaning that the right end point of l_A is not below the right end point of l_{CM}. Thus, line l_A is not below line l_{CM} between s and x_E, i.e., $l_A(x) \geqslant l_{CM}(x), \forall x \in [s, x_E]$. Thus, we have $\widehat{F}_{X_1}(x) \geqslant \widehat{F}_{X_2}(x), \forall x \in [s, x_E]$.

We are able to derive the same conclusion for range $[x_E, e]$. Therefore, the lemma holds.

Case 2: *Point E is below point M* (see Figure B.5(b)). In this case, we cannot give any guarantee, and the range $[s, e]$ is divided into two sub-ranges $[s, x_E]$ and $[x_E, e]$ based on the divided point, here x_E. We then recursively call *SpeedupSSD* on $[s, x_E]$ and $[x_E, e]$, respectively.

B.4.3 Checking SCSD

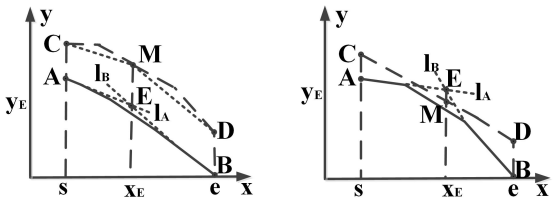
Similar to the case for checking SSD, the naive algorithm is based on the definition of SCSD. Lemma B.4.6 enables an initial check for SCSD.

Lemma B.4.6
If $X_1 \succ_{scsd} X_2$, then $X_1.max \leqslant X_2.max$ and $E(X_1) \leqslant E(X_2)$.

Proof. We prove $X_1.max \leqslant X_2.max$. Assuming that $X_1.max > X_2.max$ by contradiction, we have $\widetilde{F}_{X_1}(X_1.max) = 0$ and $\widetilde{F}_{X_2}(X_1.max) > 0$. Thus, $\widetilde{F}_{X_2}(X_1.max) < \widetilde{F}_{X_1}(X_1.max)$. According to the definition of SCSD, when $X_1 \succ_{scsd} X_2$, we have $\widetilde{F}_{X_1}(X_1.max) \geqslant \widetilde{F}_{X_2}(X_1.max)$. This yields a contradiction. Thus, the assumption is invalid, and $X_1.max \leqslant X_2.max$.

We apply Theorem B.3.3 to prove $E(X_1) \leqslant E(X_2)$. Consider a non-increasing convex utility function $u(x) = -x$. According to Theorem B.3.3, we have $E_U(X_1) \geqslant E_U(X_2)$. Since $u(x) = -x$, we have $E_U(X_1) = -E(X_1)$ and $E_U(X_2) = -E(X_2)$. Thus, we have $E(X_1) \leqslant E(X_2)$.

Finally, we present a speedup algorithm for checking SCSD. The speedup algorithm for check SCSD is similar to the speedup algorithm for checking SSD. The only difference is that we need to check if point E is not above M, while when checking for SSD, we check if point E is not below M. We give an intuitive idea of how to perform speedup algorithm for SCSD in Figure B.6.



(a) Case 1 (b) Case 2

Fig. B.6: Speedup Algorithm for Checking SCSD.

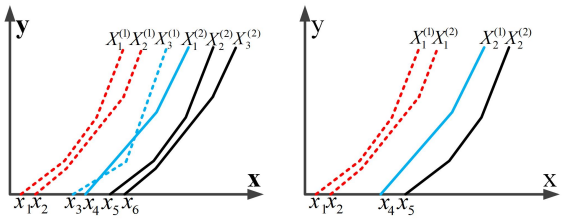
B.5 Finding Temporal Dominance

We proceed to provide efficient algorithms for checking temporal dominance among UTSs. We first solve the case of checking temporal dominance between two UTSs, and then the case of checking temporal dominance among multiple UTSs.

B.5.1 Naive Algorithm

Given two uncertain time series $T_1 = \langle X_1^{(1)}, X_1^{(2)}, \cdots, X_1^{(N)} \rangle$ and $T_2 = \langle X_2^{(1)}, X_2^{(2)}, \cdots, X_2^{(N)} \rangle$, we would like to find the temporal dominance between T_1 and T_2. In particular, during each interval j, we want to determine the dominance relationship between $X_1^{(j)}$ and $X_2^{(j)}$ w.r.t. a given order: FSD, SSD, or SCSD.

A naive algorithm checks, for each interval j, the relationship between $X_1^{(j)}$ and $X_2^{(j)}$ using the proposed speedup algorithms. For example, consider two UTSs, where each has 3 intervals $T_1 = \langle X_1^{(1)}, X_1^{(2)}, X_1^{(3)} \rangle$ and $T_2 = \langle X_2^{(1)}, X_2^{(2)}, X_2^{(3)} \rangle$, and assume that we consider FSD. The integral cdfs of the random variables in the UTSs are shown in Figure B.7. The naive algorithm needs to determine the dominance relationship between three pairs of random variables.



(a) Naive (b) Grouping

Fig. B.7: Intuition for the Grouping Strategy.

This can be very inefficient, especially when long UTSs have many intervals. To contend better with long UTSs, we group the random variables in the UTSs and compare groups. If one group dominates another, any random variable in the former group dominates every random variable in the latter group. This grouping can significantly reduce the cost of dominance checking.

For example, we form G_1 from $X_1^{(1)}$ and $X_1^{(2)}$, and form G_2 from $X_2^{(1)}$ and $X_2^{(2)}$, as shown in Figure B.7(b). Since the lower boundary curve of G_1 is always above the upper boundary curve of G_2, the random variables in G_1 dominates the random variables in G_2. Thus, we determine the dominance relationships for two intervals, by comparing one group pair. Likewise, when groups have multiple random variables, if one group dominates another, we are able to determine dominance relationships for multiple intervals, thus improving efficiency. On the other hand, large groups are less likely to dominate each other even if there is dominance between pairs of variables in them. Thus, we proceed to design a general grouping framework that is able to identify groups with appropriate sizes.

B.5.2 A General Grouping Framework

We choose the *expectations* of random variables as the *grouping criterion*. This is because $E(X_1) \leqslant E(X_2)$ is a necessary condition for $X_1 \succ_{fsd} X_2$, $X_1 \succ_{ssd} X_2$, and $X_1 \succ_{scsd} X_2$ according to Lemmas B.4.1, B.4.4, and B.4.6.

Analogous to the ideas of the initial check covered in Section B.4, we propose a grouping strategy to produce groups such that for each group, the random variables from a UTS, say T_1, have smaller expectations and the random variables from the other UTS, say T_2, have larger expectations. This way, the random variables from T_2 with larger expectations cannot dominate the random variables in T_1 with smaller expectations, and thus we only need to check if the random variables in T_1 dominate the random variables in T_2. The resulting strategy is shown in Algorithm 7. We illustrate the strategy using the example in Table B.5 that lists the expectations of two UTSs with 8 intervals.

Interval j	1	2	3	4	5	6	7	8
$E(X_1^{(j)})$ in T_1	20	39	60	80	30	35	55	75
$E(X_2^{(j)})$ in T_2	40	40	45	54	40	40	50	54

Table B.5: Expectations for UTSs T_1 and T_2.

Algorithm 7 GroupingBasedOnExpectations(T_1, T_2)

Input:
UTSs: T_1 and T_2.
Output:
A sequence of optimal random variables, q;
Identify the master UTS T_m and the slave UTS T_s;
Make an empty sequence $q \leftarrow \emptyset$;
Non-grouped interval set $NGIS \leftarrow$ all intervals;
while $NGIS \neq \emptyset$ **do**
  $E_{min} \leftarrow$ the smallest expectation in $NGIS$ in T_m;
  $E'_{min} \leftarrow$ the smallest expectation in $NGIS$ in T_s;
  if $E'_{min} < E_{min}$ **then**
    Make an empty group G;
    for each interval $j \in NGIS$ **do**
      if the expectation of the j-th interval in T_s is no larger than E_{min} **then**
        $G \leftarrow G \cup \{j\}$; Remove j from $NGIS$;
      end if
    end for
    $I \leftarrow \text{GroupCheck}(T_s, T_m, G, \min(G), \max(G))$;
    Update(q, I, T_s, T_m);
  else
    $E_{min} \leftarrow E'_{min}$;
    Make a new empty group G;
    for each interval $j \in NGIS$ **do**
      if the expectation of the j-th interval in T_m is no larger than E_{min} **then**
        $G \leftarrow G \cup \{j\}$; Remove j from $NGIS$;
      end if
    end for
    $I \leftarrow \text{GroupCheck}(T_m, T_s, G, \min(G), \max(G))$;
    Update(q, I, T_m, T_s);
  end if
end while
return q;

We choose the UTS with the smaller average expectation as the *master* and call the other UTS the *slave*. In the example, T_2 is the master and T_1 is the slave.

The algorithm identifies appropriate groups iteratively. In each iteration, it identifies the smallest expectation E_{min} among the expectations of non-grouped intervals in the master. In the first iteration in the example, $E_{min} = 40$ because 40 is the smallest expectation in T_2. The algorithms stops when all intervals are grouped (line 4). In each iteration, it distinguishes between two cases.

Case 1: The slave has intervals with expectations that are smaller than E_{min}. In this case, these intervals form a group (lines 7–15).

In the example, the first iteration belongs to case 1 because intervals 1, 2, 5, and 6 in T_1 have expectations 20, 39, 30, and 35, which are all smaller than $E_{min} = 40$. Thus $G_1 = \{1, 2, 5, 6\}$ is the first group.

The intuition is that when considering the random variables in G_1, the random variables from T_1 may dominate the random variables from T_2, while the random variables in T_2 cannot dominate the random variables from T_1. Then, we only need to further check if the random variables in T_1 dominate the random variables in T_2 by calling algorithm *GroupCheck*$(T_s, T_m, G, \min(G), \max(G))$ (line 14), where $\min(G)$ and $\max(G)$ return the minimum and maximum values of the random variables in the intervals that belong to group G from both master T_m and slave T_s, respectively. Algorithm *GroupCheck* returns the intervals in G for which the corresponding RVs in T_s dominate those in T_m. Thus, for the removed intervals in G, the RVs in T_s and T_m do not dominate each other. Based on this, a helper function *update* is called to update the result q.

Case 2: The slave has no intervals with expectations that are smaller than E_{min}. In this case, we update E_{min} to be the smallest expectation among the expectations in the non-grouped intervals in the slave. Then, in the master, the intervals whose expectations are no larger than the updated E_{min} are grouped (lines 16–26).

In the example, the second iteration belongs to case 2. After the first iteration, the non-grouped intervals are 3, 4, 7, and 8. The minimum expectation among non-grouped intervals in the master is E_{min} with value 45. In the slave, intervals 3, 4, 7, and 8 have expectations 60, 80, 55, and 75, all of which exceed 45. We then set E_{min} to 55, i.e., the minimum expectation of non-grouped intervals in the slave. Since the expectations of the non-grouped intervals in the master are 45, 54, 50, 54, which are all no larger than 55, we form the group $G_2 = \{3, 4, 7, 8\}$.

The intuition is that when considering the random variables in G_2, the random variables from the master may dominate the random variables from the slave, while the random variables in the slave cannot dominate random variables in the master. This is because the random variables in T_2 have smaller expectations than the expectations of the random variables in T_1. We then call *GroupCheck* (T_m, T_s, G, $\min(G)$, $\max(G)$) (line 24).

B.5.3 Dominance Checking for Groups

We provide a detailed discussion only for the case of SSD checking. Since checking of FSD and SCSD for groups can be done in a similar manner, we omit the details.

Dominance checking for groups is achieved by calling function *GroupCheck*(T_1, T_2,

G, min, max), shown in Algorithm 8. In the following discussion, we only consider random variables during intervals in G. Recall that when groups are formed using Algorithm 7, we ensure that T_1's random variables have smaller expectations, meaning that T_2's random variables cannot dominate T_1's random variables. Thus, we only need to check whether T_1's random variables dominate T_2's random variables.

Algorithm 8 GroupCheck(T_1, T_2, G, min, max)

Input:
UTSs: T_1 and T_2; Group G; Double: min and max;
Output:
Interval set IS of intervals where T_1 dominates T_2;
1: $IS \leftarrow \emptyset$;
2: **if** $\neg$ BoundaryCondition(T_1, T_2, G, min, max) **then**
3: Make sub-groups from G;
4: **for** each sub-group sg **do**
5: $IS \leftarrow IS \cup$ GroupCheck(T_1, T_2, sg, min, max);
6: **end for**
7: **else**
8: $s \leftarrow min(T_2, G)$; $e \leftarrow max(T_1, G)$;
9: $k_A \leftarrow min(T_1, G, s)$; $k_B \leftarrow max(T_1, G, e)$;
10: Construct line l_A from A with the slope k_A;
11: Construct line l_B from B with the slope k_B;
12: Point $E \leftarrow$ intersection of lines l_A and l_B;
13: Point $M \leftarrow$ the point on the top-most curve among the curves of T_2's random variables whose x-coordinate is x_E;
14: **if** E is not below M **then**
15: $IS \leftarrow G$;
16: **else**
17: $G' \leftarrow$ GroupCheck(T_1, T_2, G, s, i);
18: $IS \leftarrow$ GroupCheck(T_1, T_2, G', i, e);
19: **end if**
20: **end if**
21: return IS;

We use $G = \{1, 2, 5, 6\}$, i.e., the first group found in Section B.5.2, as an example. The integrals of the cdfs of the random variables in G from T_1 and T_2 are shown in Figure B.8(a).

Boundary Condition

We start by checking a boundary condition (line 2 in Algorithm 8). Consider the random variables (RVs) in G. If the RVs in T_1 dominate the RVs in T_2, the curves of the integral of the cdfs of the RVs in T_1 should appear above the corresponding curves for T_2. The boundary condition checks if this is true on the left and right boundaries.

If this is not true on either boundary, T_1's curves cannot always appear above T_2's curves, and thus we cannot guarantee that T_1's RVs dominate T_2's RVs. Thus, we need to partition G into sub-groups and further check for each sub-group whether T_1's curves can always appear above T_2's curves.

We show how the boundary condition works for the left boundary. It works similarly for the right boundary. We first identify the left boundary lb as the smallest minimum values of the RVs in T_2. For each interval $i \in G$, we consider two copies, one copy for the random variable in interval i in T_1, represented as a square, and one copy for the random variable in interval i in T_2, represented as a circle—see Figure B.8(b).

Next, we associate with each interval copy a value that equals its random variable's integral of cdf at lb. In particular, for interval copy $\boxed{5}$, which corresponds to random variable $X_1^{(5)}$ from T_1, the associated value is $\widehat{F}_{X_1^{(5)}}(lb)$. For interval copy 1, which corresponds to random variable $X_2^{(1)}$ from T_2, and the associated value is $\widehat{F}_{X_2^{(1)}}(lb)$. Next, we order the interval copies according to their associated values. Figure B.8(b) shows such ordered interval copies. Specifically, since RVs in the 1st and 5th intervals from T_1 have non-zero values y_1 and y_2, and all the remaining RVs have 0s, $\boxed{1}$ and $\boxed{5}$ appear at the top, and the remaining interval copies follow.

Next, we identify the interval copies from T_2 that appear at the top. In our example, it is interval 1 from T_2, i.e., 1. All interval copies from T_1 that appear above 1 form a sub-group. In the example, intervals 1 and 5 form a sub-group. This is so because, on the left boundary and during intervals 1 and 5, the RVs from T_1 appear above the RVs from T_2, meaning that it is possible that the RVs during intervals 1 and 5 from T_1 dominates those from T_2. The remaining intervals form another sub-group.

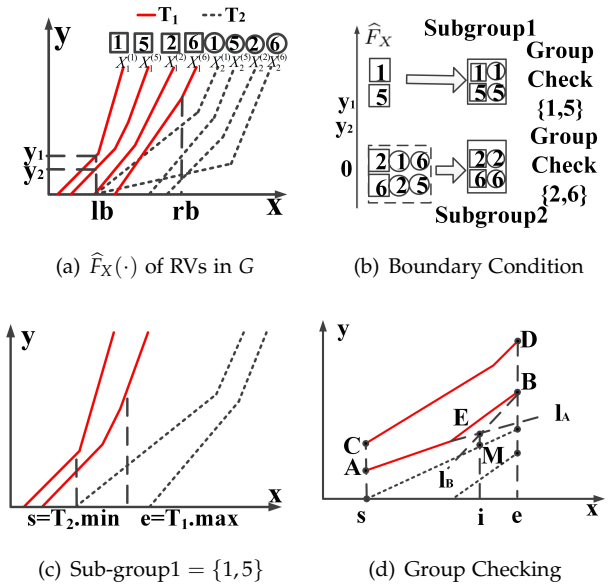
If one of the sub-groups is empty, this means that it is not necessary to make further sub-groups at boundaries and that group G passes the boundary condition checking. Otherwise, we check dominance for each sub-group by recursively calling *GroupCheck* on each sub-group (lines 3–5 in Algorithm 8).

Group Check with Lower and Upper Bounds

We follow an idea similar to that used in the speedup algorithm for checking SSD. We first identify s as the the smallest minimum value of the RVs in T_2 and e as the largest maximum value of the RVs in T_1 (line 8), see Figure B.8(c). In order to better explain the idea of group check, we give several definitions and lemmas.

Definition B.5.1
Lower Bound. We use H to denote the integral cdfs of a set of RVs. Given a range $[s, e]$ on the x-axis, the minimum values of H at s and e are denoted as $H_{s,min} = \min_{\widehat{F}_i \in H} \widehat{F}_i(s)$ and $H_{e,min} = \min_{\widehat{F}_i \in H} \widehat{F}_i(e)$, respectively. The minimum value of the derivatives of H at s is denoted as $H'_{s,min} = \min_{\widehat{F}_i \in H} \widehat{F}'_i(s)$, and the maximum value of the derivatives of H at e is denoted as $H'_{e,max} = \max_{\widehat{F}_i \in H} \widehat{F}'_i(s)$. The lower bound in range $[s, e]$, denoted as $lb(H, s, e)$, consists of the following two line segments.

$$\begin{cases} y = H'_{s,min} \cdot (x - s) + H_{s,min}, \text{ when } x \in [s, i] \\ y = H'_{e,max} \cdot (x - e) + H_{e,min}, \text{ when } x \in (i, e] \end{cases}$$



(a) $\widehat{F}_X(\cdot)$ of RVs in G (b) Boundary Condition

(c) Sub-group1 $= \{1,5\}$ (d) Group Checking

Fig. B.8: Group based Dominance Checking.

Here, i is the x-coordinate of the intersection of the two line segments.

For example, consider the RVs in T_1, whose integral cdfs are shown in Figure B.8(d). Points A and B hold the minimum values of the integral cdfs of the RVs at s and e, respectively. This means that point A's y-coordinate is $H_{s,min}$ and point B's y-coordinate is $H_{e,min}$. The slopes of lines l_A and l_B equal $H'_{s,min}$ and $H'_{e,max}$, respectively. Lines l_A and l_B intersect at point E, whose x-coordinate is i. Thus, the lower bound of the RVs in T_1 consists of line segment AE when $x \in [s,i]$ and line segment EB when $x \in [i,e]$.

Lemma B.5.1
For any value $x \in [s,e]$ on the x-axis, and any integral cdf $\widehat{F}_i \in H$, we have $lb(H,s,e)(x) \leqslant \widehat{F}_i(x)$, where $lb(H, s, e)(x)$ returns the y-coordinate of the point that is on the lower bound and whose x-coordinate is x.

Proof. We use proof by contradiction. We start proving on the left range $[s,i]$. Assume that there exists a value $x' \in [s,i]$ and an integral cdf $\widehat{F}_i \in H$ such that $\widehat{F}_i(x') < lb(H, s, e)(x')$. If so, the slope of the chord line that connects point $(s, \widehat{F}_i(s))$ and point $(x', \widehat{F}_i(x'))$ is smaller than $H'_{s,min}$. However, since any integral cdf is convex, the slope of the chord line of any integral cdf in H on range $[s,i]$ should be no samller than $H'_{s,min}$. This yields a contradiction. The proof for the right range $[i,e]$ is similar. Thus, the lemma holds.

Definition B.5.2
Upper Bound. We use H to denote the integral cdfs of a set of RVs. Given a range $[s,e]$ on the x-axis, the maximum values of H at s and e are denoted as $H_{s,max} =$

$\max_{\widehat{F}_i \in H} \widehat{F}_i(s)$ and $H_{e,max} = \max_{\widehat{F}_i \in H} \widehat{F}_i(e)$, respectively. For any $d \in [s,e]$, we define $H_{d,max}$ in a similar way. Given $d \in [s,e]$, the upper bound of H, denoted as $ub(H,s,e,d)$, consists of the following two line segments.

$$\begin{cases} y = \frac{H_{d,max} - H_{s,max}}{d-s} \cdot (x-s) + H_{s,max}, \text{ when } x \in [s,d] \\ y = \frac{H_{e,max} - H_{d,max}}{e-d} \cdot (x-e) + H_{e,max}, \text{ when } x \in (d,e] \end{cases}$$

Lemma B.5.2
For any value $x \in [s,e]$ on the x-axis, and any integral cdf $\widehat{F}_i \in H$, we have $ub(H,s,e,d)(x) \geqslant \widehat{F}_i(x)$, where $ub(H, s, e, d)(x)$ returns the y-coordinate of the point that is on the upper bound and whose x-coordinate is x.

Proof. We use proof by contradiction.We consider the left range $[s,d]$. Assume $\exists x' \in [s,d]$ and $\exists \widehat{F}_i \in H$ such that $\widehat{F}_i(x') > ub(H,s,e,d)(x')$. If so, the slope k of the chord line that connects point $(s, \widehat{F}_i(s))$ and point $(x', \widehat{F}_i(x'))$ exceeds the slope k' of the upper bound $ub(H,s,e,d)$ on range $[s,d]$, i.e., $k > k'$.

Further, since any cdf is a non-decreasing function, we have $\widehat{F}_i'(x'') \geqslant \widehat{F}_i'(x')$ if $x'' > x'$. Since $d > x'$, this leads to $\widehat{F}_i'(d) \geqslant \widehat{F}_i'(x') = k > k'$. Thus, we have $\widehat{F}_i(d) > ub(H,s,e,d)(d) = H_{d,max}$. This yields a contradiction. The proof for the right range $[d,e]$ is similar. Therefore, the lemma holds.

With Lemmas B.5.1 and B.5.2, it is straightforward to obtain the following lemma to decide the dominance relationships between two sets of RVs.

Lemma B.5.3
Given a master set of RVs with integral cdfs H^{master}, a slave set of RVs with integral cdfs H^{slave}, and a range $[s,e]$, if H^{master} and H^{slave} meet the following three conditions then $\forall \widehat{F}_k \in H^{master}, \forall \widehat{F}_j \in H^{slave}, \forall x \in [s,e]$, we have $\widehat{F}_k(x) \geqslant \widehat{F}_j(x)$.

(1) $H^{master}_{s,min} \geqslant H^{slave}_{s,max}$;
(2) $H^{master}_{e,min} \geqslant H^{slave}_{e,max}$;
(3) $lb(H^{master}, s, e)(i) \geqslant ub(H^{slave}, s, e, i)(i)$.

Proof. Conditions (1) and (3) guarantee that, in range $[s,i]$, the lower bound of H^{master} is above the upper bound of H^{slave}. Similarly, conditions (2) and (3) guarantee that, in range $[i,d]$, the lower bound of H^{master} is above the upper bound of H^{slave}.

Algorithm 8 proceeds according to Lemma B.5.3. We identify the important points A and B, where A and B are the *lowest* points of the integral cdfs of RVs in T_1 whose x-coordinate are s and e, respectively, as shown in Figure B.8(d). We identify two slopes k_A and k_B, where k_A is the *smallest* slope of the integral cdfs of RVs in T_1 when the x-coordinate is s and k_B is the *largest* slope of the integral cdfs of RVs in T_1 when the x-coordinate is e. We construct line l_A through point A with slope k_A and line l_B through point B with slope k_B (lines 9–11). We then compute the intersection E of lines l_A and l_B. If E does not appear below the upper bound of T_2 then all RVs in G from T_1 dominate all RVs in G from T_2 (lines 12–15). Otherwise, we check sub-ranges (s,i) and (i,e) (lines 16–18).

B.5.4 Checking Multiple UTSs

Based on the proposed grouping strategy, we have an efficient method to check the temporal dominance between two UTSs. However, a UTS collection TS may contain multiple UTSs. We propose two alternative methods to check the temporal dominance among all UTSs in a TS. The first simply checks every UTS pair using the efficient algorithm based on the grouping strategy. The second employs a merge-sort like procedure to compare UTS pairs while employing the intermediate comparison results to construct the final result.

For example, consider $TS = \{T_1, T_2, T_3, T_4\}$. The first method checks the dominance relationships between pairs $(T_1,\ T_2)$, $(T_1,\ T_3)$, $(T_1,\ T_4)$, $(T_2,\ T_3)$, $(T_2,\ T_4)$, and $(T_3,\ T_4)$. The second method first checks $(T_1,\ T_2)$ and $(T_3,\ T_4)$ and then uses the results we consider to construct the final result.

B.5.5 User Defined Constraints

Users may specify additional travel time constraints. For example, a user may only be interested in paths with an arrival times that are no later than 6 p.m. or that offer a possibility of arriving before 4 p.m.

We use Left Constraint (LC) and Right Constraint (RC) to denote the shortest and longest travel times that a user is interested in. For example, assume that the current time is 2 p.m. and that a user specifies the following contraint: "the arrival time must not exceed 6 p.m. or there must be a possibility of arriving before 4 p.m." We can then set LC to 120 mins and RC to 240 mins.

Based on LC and RC, we can prune random variables, before checking the stochastic dominance relationships. Specifically, using LC, we can prune random variable X if the minimum value of X exceeds LC. In other words, if $F_X(LC) \leq 0$, we prune X. Similarly, based on RC, we can prune random variable X if the maximum value of X exceeds RC. In other words, if $F_X(RC) < 1$, we prune X. We only consider a random variable X if and only if X satisfies both the LC and the RC constraints.

Algorithm 9 shows how we use the two constraints. Given constraints, T_1 and T_2 and time interval j, (1) if the j-th random variable in T_1 does not meet one of the constraints and the j-th random variable in T_2 meets both constraints then T_2 is the optimal choice in time interval j (lines 4–6); (2) if the j-th random variable in T_2 does not meet one of the constraints and the j-th random variable in T_1 meets both constraints then T_1 is the optimal choice in time interval j (lines 7–9); (3) if the j-th random variables in both UTSs do not meet the constraints, time interval j has no optimal choice and is removed (lines 10–11).

To incorporate the user-defined constraints, Algorithm 9 must be called to filter random variables that do not satisfy the constraints. Specifically, Algorithm 9 needs to be called after line 3 in Algorithm 7 to update q and $NGIS$, respectively. Afterwards, Algorithm 7 continues to check the remaining random variables that satisfy the constraints.

Algorithm 9 UserDefinedConstraintsCheck(T_1, T_2, LC, RC)

Input:
UTSs: T_1 and T_2, LC, RC.
Output:
A sequence of optimal random variables, q, and remaining intervals I;
Make an empty sequence $q \leftarrow \emptyset$;
Non-grouped interval set $I \leftarrow$ all intervals;
for each interval $j \in I$ **do**
  if ($T_1[j].min > LC$ or $T_1[j].max > RC$) and ($T_2[j].min \leq LC$ and $T_2[j].max \leq RC$) **then**
    Remove j from I;
    Update(q, j, T_2, T_1);
  else if ($T_1[j].min \leq LC$ and $T_1[j].max \leq RC$) and ($T_2[j].min > LC$ or $T_2[j].max > RC$) **then**
    Remove j from I;
    Update(q, j, T_1, T_2);
  else if ($T_1[j].min > LC$ or $T_1[j].max > RC$) and ($T_2[j].min > LC$ or $T_2[j].max > RC$) **then**
    Remove j from I;
  end if
end for
return q, I;

B.6 Empirical Study

B.6.1 Experimental Setup

We consider two different collections of uncertain time series.
Real UTSs (RU): We use a large GPS data set of more than 180 million GPS records collected in Denmark from January 2007 to December 2008. We eliminate GPS records with unreasonable speeds, i.e., more than 200 km/h, since the speed limit on highways in Denmark is 130 km/h.

Next, we align GPS records with the road network to obtain trajectories using a well-known map-matching algorithm [12]. Given an origin and a destination (OD) pair, we first count the number of trajectories between the OD pair. Then we select the top 15,000 OD pairs with the largest numbers of trajectories. For each chosen OD pair, we consider all paths that have been used by trajectories that connect the pair.

We derive UTSs for the edges in the road network using the method outlined in Section B.2. Then, for each considered path, we construct a UTS using the UTSs of the edges in the path using an existing method [7, 14].

This way, we obtain a UTS collection for each OD pair. As OD pairs are connected by between 2 and 6 paths, we obtain UTS collections with cardinalities that vary from 2 to 6, as shown in Table B.7 and the corresponding number of UTS collections is

shown in Table B.6.

$\|\mathcal{TS}\|$	2	3	4	5	6
Num of Collections	12,461	1,266	164	20	4

Table B.6: Number of UTS Collections.

Synthetic UTSs (SU): Following a common practice from studies of UTSs [19–21], we also construct UTSs from deterministic time series (DTSs). In particular, we use a publicly available DTS data set[3]. In a DTS, each interval is associated with a deterministic value c rather than a random variable. To construct a UTS from a DTS, we construct a random variable based on the deterministic value c for each interval. Specifically, generate three distributions—normal, uniform, and exponential—and use c as their mean. The standard deviation is chosen as τ times the standard deviation of all the deterministic values in a DTS, where τ varies from 0.1 to 2. This procedure follows a recommendation from a recent UTS benchmark [19].

Data Cleaning: In both datasets, we smooth the probability mass function (pmf) of each random variable. In particular, we set $pmf(x) = 0$ if $pmf(x) < \epsilon$, and we set ϵ to 1×10^{-8} in the experiments; and then we normalize each pmf so that $\sum pmf(x) = 1$. By doing this, we remove insignificant biases.

Parameters: For both RU and SU, we use equi-width histograms to represent the random variables. We vary the number b of bins in a histogram from 100 to 1,000. We also vary the cardinality $|\mathcal{TS}|$ of a UTS collection, parameter τ, and the distribution of the synthetic UTSs according to Table B.7, where the default values are in bold.

Parameter	*Values*
b (10^2)	1, 2, 3, 4, 5, 6, 7, 8, **9**, 10
$\|\mathcal{TS}\|$	2, 3, 4, **5**, 6
τ	0.1, 0.5, 1, **2**
Distributions	**Normal (n)**, Exponential (e), Uniform (u)

Table B.7: Parameter Settings.

Since SSD and SCSD are defined in a similar way and also due to the space limitation, we only report findings for SSD.

Note that temporal dominance queries only require users to provide a stochastic dominance parameter x that indicates *fsd*, *ssd*, or *scsd*—users need not provide specific utility functions.

Implementation Details: All algorithms are implemented in Python. We conduct experiments on a server with a 64-core AMD Opteron(tm) 2.24 GHZ CPU, 528 GB main memory under Ubuntu Linux.

[3]http://academic.udayton.edu/kissock/http/Weather/

B.6.2 Efficiency

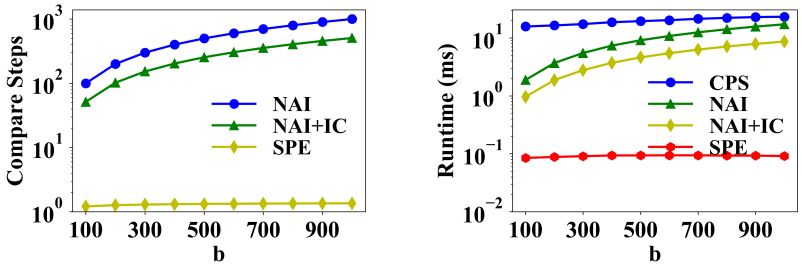
Checking Stochastic Dominance

We first study the performance of checking stochastic dominance between two random variables. We consider four algorithms: (1) *NAI*: the naive algorithm according to the definition of the corresponding order. (2) *NAI+IC*: the naive algorithm with initial checks. This also corresponds to a state-of-the-art method [22, 23], where a detailed discussion is included in Section B.7. (3) *SPE*: the speedup algorithm. (4) *CPS*: a method that represents different orders of stochastic dominance as constraints in linear programming and solves the resulting linear problem using CPLEX[4], a state-of-the-art optimization software package that is widely used in statistics and operations research [24].

We consider two measurements—the number of comparison steps and runtime. For *NAI* and *NAI+IC*, the number of comparison steps corresponds to the number of the comparisons between two random variables' cdfs or cdf integrals. For *SPE*, the number of comparison steps correspond to the number of the recursive calls in Algorithms 3 and 6, because they compare two random variables' cdfs or integrals of cdfs in each recursive call.

For brevity, in this set of experiments, we only report results for SSD and omit results for FSD if they are similar to those for SSD.

Impact of b: We first evaluate the scalability of the different algorithms w.r.t. the number of histogram bins, with results shown in Figure B.9 (RU) and Figure B.10 (SU). For both data sets, the number of comparison steps and the runtime of *NAI* and *NAI+IC* increase almost linearly with b. However, the speedup algorithm *SPE* has stable performance and is insensitive to b. Further, *SPE* shows substantial improvement over *NAI* and *NAI+IC*.

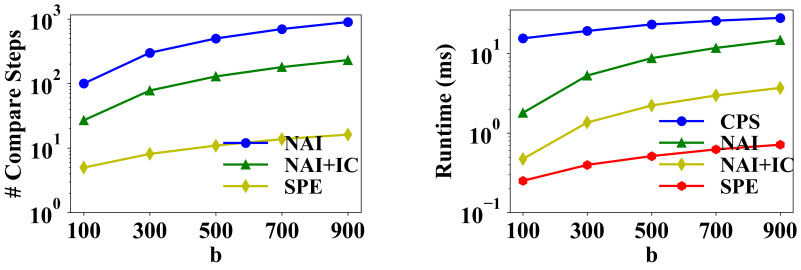


(a) Comparison Steps (b) Runtime

Fig. B.9: Impact of b, SSD, RU.

Next, we consider the method *CPS*. As *CPS* works as a black box, we are unable to count the number of comparison steps and only measure runtime. Figures B.9(b) and B.10(b) show that although *CPS* is also insensitive to b, the runtime is even worse

[4]https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

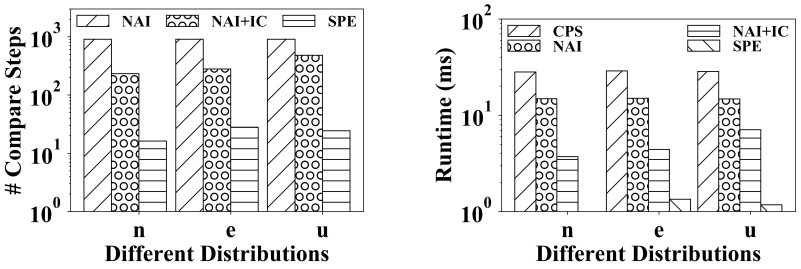


(a) Comparison Steps (b) Runtime

Fig. B.10: Impact of b, SSD, SU.

than that of *NAI*. This is because *CPS* has an initialization overhead for constructing its linear programming model.

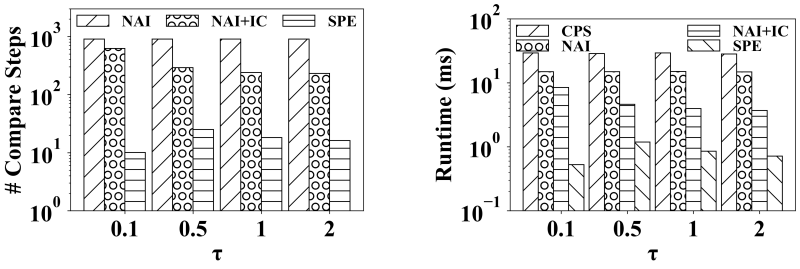
Impact of distributions: Figure B.11 reports the performance of the different methods for different distributions, where n, e, and u denote normal, exponential, and uniform distributions. We see that *SPE* significantly outperforms other methods in all settings. This experiment implies that *SPE* is robust to different distributions.



(a) Comparison Steps (b) Runtime

Fig. B.11: Impact of Distributions, SSD, SU.

Impact of τ: Next, we consider the impact of varying standard deviations. Figure B.12 shows the results for normal distributions. As the standard deviation increases, the performance of *NAI+IC* improves slightly. This is because although the range that needs to be checked increases with a larger standard deviation, *NAI+IC* may find an inconsistent point early and may thus terminate in a few steps, which results in less runtime. The runtime for *SPE* increases when τ increases from 0.1 to 0.5 and then decreases when τ increases from 0.5 to 2. The reason is two-fold. First, when τ is very small, the compared cdf integrals may be separated by substantial gaps that yield a clear dominance relationship, which results in less runtime. Second, when τ becomes larger, e.g., when $\tau > 1$, the gaps become smaller, and the cdf integrals are increasingly likely to intersect, resulting in situations where random variables cannot dominate each other. Therefore, when τ is either very small or very large, it takes less time to check dominance reltionships.



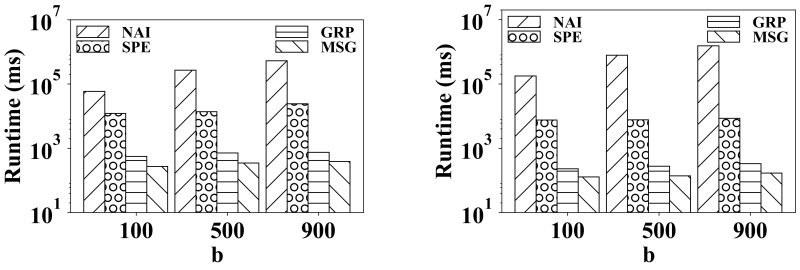
(a) Comparison Steps (b) Runtime

Fig. B.12: Impact of τ, SSD, SU.

Temporal Dominance Queries in UTSs

We evaluate the performance of processing temporal dominance queries on UTS collections. We consider four methods for temporal dominance queries. Neither *NAI* nor *SPE* use the grouping strategy, but rather use the naive algorithm with initial checking and the speedup algorithm to check the dominance relationships for each interval. Both *GRP* and *MSG* use the grouping strategy, and *GRP* does not use the merge sort like procedure that is used by *MSG*.

Impact of b: Figure B.13 shows the runtime when varying b on RU. The runtime of *NAI* increases almost linearly with the increase in b, while the other three methods, which are all based on the proposed speedup algorithm, only increases slightly. This occurs because the speedup algorithm for checking FSD and SSD between two random variables is insensitive to b. Figure B.13 also shows that the proposed grouping strategy is effective—the two methods that use the grouping strategy, i.e., *GRP* and *MSG*, outperform the other two methods that do not use the grouping strategy, i.e., *NAI* and *SPE*.

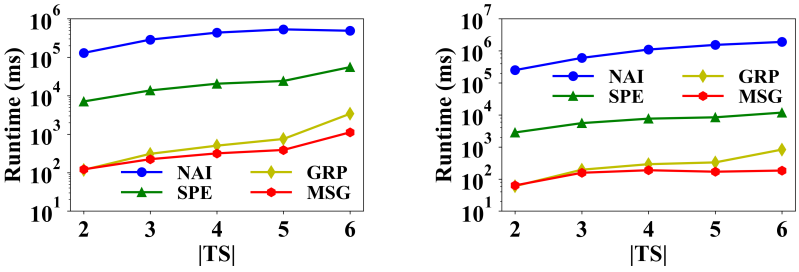


(a) FSD (b) SSD

Fig. B.13: Impact of b, UTSs, RU.

Impact of $|TS|$: Figure B.14 shows the runtime of the different methods when varying the UTS collection cardinality $|TS|$ on RU. The results show that as the car-

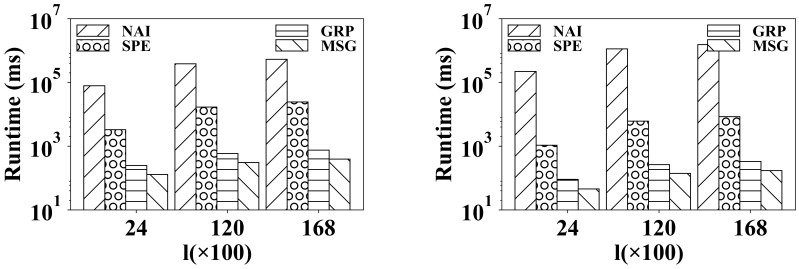
dinality increases, the methods using the grouping strategy achieve more significant runtime improvements. Further, the use of a merge sort like procedure enables *MSG* to achieve better run-time than *GRP*, especially when cardinality is large.



(a) FSD (b) SSD

Fig. B.14: Impact of $|TS|$, UTSs, RU.

Impact of l: We show the impact of the length l of UTSs on RU in Figure B.15. As length l increases, the runtimes of all methods increase. However, runtimes of the two methods using the grouping strategy, i.e., *GRP* and *MSG*, only increase slightly, as they are able to group similar random variables together and identify dominance relationships between random variable groups. When the time series are long, there are more random variables that can be grouped. For all values of l, *GRP* and *MSG* are significantly faster than the other two methods that do not use the grouping strategy, on both FSD and SSD.

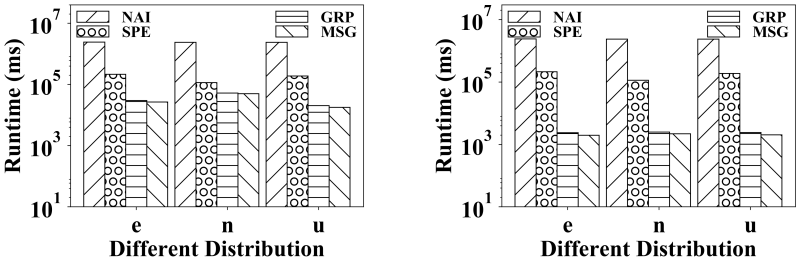


(a) FSD (b) SSD

Fig. B.15: Impact of Length, UTSs, RU.

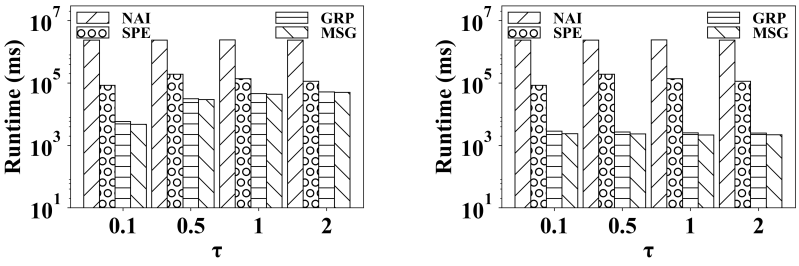
Impact of distributions: Figure B.16 shows the effect of different distributions on SU. *GRP* and *MSG*, which use the grouping strategy, outperform the methods without grouping strategy for all distributions. This indicates that the proposed grouping strategy is robust to varying distributions.

Impact of τ: Figure B.17 shows the impact of varying standard variations when using normal distributions on SU. On FSD, the performance of the grouping strategy degrades slightly with larger standard deviations because such standard deviations



(a) FSD (b) SSD

Fig. B.16: Impact of Distribution, UTSs, SU.



(a) FSD (b) SSD

Fig. B.17: Impact of τ, UTSs, Normal Dist., SU.

have the effect that the initially formed groups based on means will be split into many sub-groups later on, which adversely affects the efficiency. On SSD, the performance of the grouping strategy is more stable and is insensitive to standard deviation variations. In all settings, *GRP* and *MSG* are faster than the other two methods that do not use grouping, and *MSG* is the fastest.

B.6.3 Effectiveness

Comparison with Expectations

To gain insight into the functionality improvements of the proposed methods that take into account users' risk preferences, we compare with a method that does not take users' risk preferences into account. Specifically, we employ a method that uses expected values, i.e., expected travel time in our setting. This means that one random variable dominates another random variable if the former has a smaller expected value.

Given a UTS collection, this expectation-based method always chooses, for each interval, the random variable with the smallest expected travel time, meaning that it does not consider any risk preferences. In contrast, the temporal dominance queries choose, for each interval, the random variables that are not dominated by other ran-

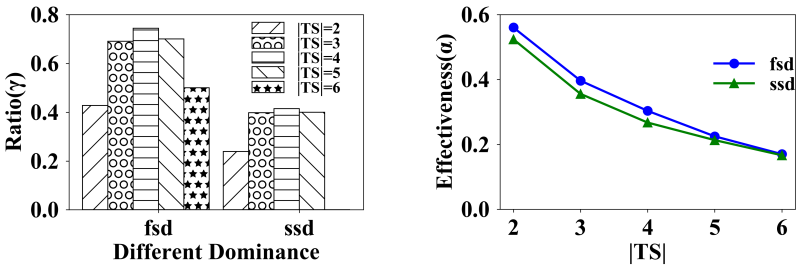
dom variables w.r.t. fsd (i.e., for risk-neutral users), ssd (i.e., for risk-loving users), or scsd (i.e., for risk-averse users).

We compute the ratio $\gamma = \frac{Diff}{Total}$, where *Diff* is the number of queries whose results returned by the two methods are different and *Total* is the number of all queries. Thus, the ratio expresses the difference between the results returned by the two methods. Figure B.18(a) shows that the temporal dominance queries w.r.t. fsd and ssd return significantly different results when compared to the method using expected values. This offers evidence that the proposed temporal dominance queries are able to support different scenarios with different risk preferences.

Ratio of Non-Dominated Variables

In this experiment, we evaluate the effectiveness of the proposed temporal dominance queries when a specific utility function is provided, i.e., the third scenario in Section B.3.3. In this scenario, we consider two options. First, we compute the expectations for all random variables based on the given utility function. We denote the number of expectation computations by *total*. Second, we first categorize each utility function as convex, concave, or other; then we only need to compute expected utilities for the random variables in $O_{fsd}(RVS)$, $O_{ssd}(RVS)$, or $O_{scsd}(RVS)$, not for other dominated *RVs*. We denote the number of expectation computation in the second option as *NonDominated*.

Next, we use the ratio $\frac{NonDominated}{Total}$ to measure the effectiveness of calculating expected utilities using the two options. The lower the ratio is, the better pruning effectiveness the temporal dominance queries provide. Figure B.18(b) shows that the ratios w.r.t fsd and ssd go down when the cardinalities of the UTS increase. On average, more than half of the random variables can be safely pruned when the cardinality exceeds 2. In addition, we also observe that the ratio for fsd is always larger than that for ssd. This is true because the non-dominated set regarding ssd is always a subset of that regarding fsd.



(a) Comparison with Expectations (b) Ratio of Non-Dominated Variables

Fig. B.18: Effectiveness, RU.

B.7 Related Work

We review related work in relation to stochastic dominance, uncertain time series, and path selection.

Stochastic Dominance: Stochastic dominance is fundamentally different from the dominance notion that is used widely in *skyline* queries [25], where dominance is defined in terms of pareto-optimality for multi-dimensional, deterministic data. In contrast, the stochastic dominance considered here is defined on one-dimensional, uncertain data.

Skyline queries can also be applied to uncertain data, yielding *probabilistic* [26, 27] and *stochastic* [7, 22, 23, 28] skyline queries. Probabilistic skylines adopt a possible worlds semantics [29]. A skyline is computed for each possible world, and an uncertain object has a probability of being in the skyline when considering all possible worlds. Stochastic dominance does not use possible world semantics, but instead considers cdfs and integrals of cdfs.

Stochastic skylines [22, 23] are defined in terms of stochastic dominance. However, only first order dominance is considered, and algorithms exist only for discrete distributions in the form of $(value, probability)$ pairs. The efficiency of an existing method relies on the number of *distinct values* [23]. When the *value*s in the discrete distributions are sparse and not aligned, there exist only a limited number of distinct values, which offers good efficiency. In contrast, when the *value*s in the discrete distributions are not sparse or well aligned, e.g., discrete representations of continuous distributions, all the *value*s become distinct values, and the method deteriorates to the *NAI+IC* method covered in Section B.6.2. Our proposal supports stochastic dominance with different orders of dominance and supports both discrete and continuous distributions. Further, the stochastic skyline operator does not take into account random variable sequences, while our proposal considers this setting and provides efficient group dominance checking for random variable sequences.

Uncertain time series: Most studies involving uncertain time series (UTS) concern similarity search in a top-k or threshold-based manner [19–21, 30, 31]. In contrast, our temporal dominance query on uncertain time series identifies the optimal random variables in each considered time interval, which is different from similarity search.

In particular, MUNICH [20] and PROUD [30] consider threshold-based similarity search where two thresholds are provided in advance—a distance threshold ϵ and a probability threshold τ. Given a query UTS T_Q, a threshold-based similarity search returns a set of UTSs. Each returned UTS, say T_i, has a sufficiently large (i.e., larger than τ) probability that the distance between T_i and the query UTS T_Q is smaller than the distance threshold ϵ. MUNICH assumes that the random variables in different intervals are independent and offers speed-up techniques to answer threshold-based similar searches. PROUD assumes that statistical information, e.g., means and variances, are available for all random variables in different intervals and utilizes the central limit theorem to prune dissimilar UTSs.

DUST [31] also considers similarity search but relies on only a single distance threshold ϵ. In particular, given two UTSs, DUST is able to return a deterministic distance value that measures the similarity between the two UTSs without requiring a probability threshold τ. DUST performs better than MUNICH and PROUD when

random variables follow different distributions during different intervals. A benchmark [19] compares the three methods, i.e., MUNICH, PROUD, and DUST, for UTS similarity search and makes recommendations on the scenarios in which each method performs the best.

Holistic-PkNN [21] investigates the problem of efficiently computing top-k nearest neighbors of UTSs. Given a query UTS T_Q, the probability of a UTS being the nearest neighbor to T_Q is defined, and an efficient query processing algorithm is proposed to find k UTSs having the top-k largest probabilities.

One study considers probabilistic skylines on uncertain time series [32]. It adopts possible worlds semantics to model a UTS as a collection of multi-dimensional points and then applies probabilistic skyline queries to them. Our study also differs from this study, as we do not use possible worlds semantics.

Path selection: A few studies consider path selection under different user preference [13, 33–35]. They either do not consider uncertain travel times or consider only uncertain travel times for some specific departure time or interval. Some of these studies also consider specific preference functions or preference vectors, but they do not consider preference categories. Thus, the paper's proposal also differs from this line of research. A recent paper considers identifying non-dominated paths w.r.t. fsd [36] for a given interval, e.g., a peak or off-peak interval, while our paper is able to identify non-dominated paths w.r.t fsd, ssd, and scsd for different intervals.

B.8 Conclusions and Outlook

We propose and study temporal dominance queries on uncertain time series. We provide a comprehensive analysis of stochastic dominance and user risk preferences, and utility functions. Efficient methods for checking stochastic dominance between two random variables and two random variable groups are proposed to efficiently compute temporal dominance queries on UTSs. Empirical studies with two real world UTS collections suggest that the proposed methods are effective and efficient.

In future work, it is of interest to consider decision making with multi-variate random variables, e.g., travel paths with both travel time and fuel consumption [37–39] distributions. It is also of interest to consider grouping strategies involving different UTSs, not only different intervals, to further improve temporal dominance query processing efficiency.

References

[1] C. Guo, C. S. Jensen, and B. Yang, "Towards total traffic awareness," *SIGMOD Record*, vol. 43, no. 3, pp. 18–23, 2014.

[2] J. Walpen, E. M. Mancinelli, and P. A. Lotito, "A heuristic for the od matrix adjustment problem in a congested transport network," *European Journal of Operational Research*, vol. 242, no. 3, pp. 807–819, 2015.

[3] E. Youngblom, "Travel time in macroscopic traffic models for origin-destination estimation," Master's thesis, University of Wisconsin-Milwaukee, 2013.

[4] The Department of Geosciences and Geography, University of Helsinki, "Multimodal accessibility measures in the Helsinki metropolitan region: MetropAccess-Travel Time Matrix," http://blogs.helsinki.fi/accessibility/data/, 2015.

[5] F. Wang and Y. Xu, "Estimating o–d travel time matrix by google maps api: implementation, advantages, and implications," *Annals of GIS*, vol. 17, no. 4, pp. 199–209, 2011.

[6] B. Yang, M. Kaul, and C. S. Jensen, "Using incomplete information for complete weight annotation of road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1267–1279, 2014.

[7] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, "Stochastic skyline route planning under time-varying uncertainty," in *Proceedings of the 30th IEEE International Conference of Data Engineering*, 2014, pp. 136–147.

[8] B. Yang, C. Guo, and C. S. Jensen, "Travel cost inference from sparse, spatio-temporally correlated time series using markov models," *PVLDB*, vol. 6, no. 9, pp. 769–780, 2013.

[9] E. Erkut, A. Ingolfsson, and G. Erdoğan, "Ambulance location for maximum survival," *Naval Research Logistics (NRL)*, vol. 55, no. 1, pp. 42–58, 2008.

[10] D. Woodard, G. Nogin, P. Koch, D. Racz, M. Goldszmidt, and E. Horvitz, "Predicting travel time reliability using mobile phone GPS data," *Transportation Research Part C: Emerging Technologies*, vol. 75, pp. 30 – 44, 2017.

[11] E. Jenelius, "The value of travel time variability with trip chains, flexible scheduling and correlated travel times," *Transportation Research Part B: Methodological*, vol. 46, no. 6, pp. 762–780, 2012.

[12] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *Proceedings of the 17th ACM International Conference on Advances in Geographic Information Systems*, 2009, pp. 336–343.

[13] B. Yang, C. Guo, Y. Ma, and C. S. Jensen, "Toward personalized, context-aware routing," *The VLDB Journal–The International Journal on Very Large Data Bases*, vol. 24, no. 2, pp. 297–318, 2015.

[14] J. Dai, B. Yang, C. Guo, C. S. Jensen, and J. Hu, "Path cost distribution estimation using trajectory data," *PVLDB*, vol. 10, no. 3, pp. 85–96, 2016.

[15] J. Hu, B. Yang, C. S. Jensen, and Y. Ma, "Enabling time-dependent uncertain eco-weights for road networks," *GeoInformatica*, vol. 21, no. 1, pp. 57–88, 2017.

[16] M. Kijima and M. Ohnishi, "Stochastic orders and their applications in financial optimization," *Mathematical Methods of Operations Research*, vol. 50, no. 2, pp. 351–372, 1999.

[17] H. Levy, "Stochastic dominance and expected utility: survey and analysis," *Management Science*, vol. 38, no. 4, pp. 555–593, 1992.

[18] A. W. Jerrold Marsden, *Calculus II*, 2nd ed. Springer-Veriag New York Inc., 1985.

[19] M. Dallachiesa, B. Nushi, K. Mirylenka, and T. Palpanas, "Uncertain time-series similarity: Return to the basics," *PVLDB*, vol. 5, no. 11, pp. 1662–1673, 2012.

[20] J. Aßfalg, H. Kriegel, P. Kröger, and M. Renz, "Probabilistic similarity search for uncertain time series," in *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, 2009, pp. 435–443.

[21] M. Dallachiesa, T. Palpanas, and I. F. Ilyas, "Top-k nearest neighbor search in uncertain data series," *PVLDB*, vol. 8, no. 1, pp. 13–24, 2014.

[22] X. Lin, Y. Zhang, W. Zhang, and M. A. Cheema, "Stochastic skyline operator," in *Proceedings of the 27th IEEE International Conference of Data Engineering*, 2011, pp. 721–732.

[23] W. Zhang, X. Lin, Y. Zhang, M. A. Cheema, and Q. Zhang, "Stochastic skylines," *ACM Transactions on Database Systems*, vol. 37, no. 2, p. 14, 2012.

[24] Q. Li, Y. M. Nie, S. Vallamsundar, J. Lin, and T. Homem-de Mello, "Finding efficient and environmentally friendly paths for risk-averse freight carriers," *Networks and Spatial Economics*, vol. 16, no. 1, pp. 255–275, 2016.

[25] S. Börzsönyi, D. Kossmann, and K. Stocker, "The skyline operator," in *Proceedings of the 17th IEEE International Conference of Data Engineering*, 2001, pp. 421–430.

[26] J. Pei, B. Jiang, X. Lin, and Y. Yuan, "Probabilistic skylines on uncertain data," in *PVLDB*, 2007, pp. 15–26.

[27] W. Zhang, X. Lin, Y. Zhang, W. Wang, and J. X. Yu, "Probabilistic skyline operator over sliding windows," in *Proceedings of the 25th IEEE International Conference of Data Engineering*, 2009, pp. 1060–1071.

[28] C. Guo, B. Yang, O. Andersen, C. S. Jensen, and K. Torp, "Ecosky: Reducing vehicular environmental impact through eco-routing," in *Proceedings of the 31th IEEE International Conference of Data Engineering*, 2015, pp. 1412–1415. [Online]. Available: http://dx.doi.org/10.1109/ICDE.2015.7113389

[29] A. D. Sarma, O. Benjelloun, A. Halevy, and J. Widom, "Working models for uncertain data," in *Proceedings of the 22nd IEEE International Conference of Data Engineering*, 2006, pp. 7–7.

[30] M. Yeh, K. Wu, P. S. Yu, and M. Chen, "PROUD: a probabilistic approach to processing similarity queries over uncertain data streams," in *Proceedings of the 12th International Conference on Extending Database Technology*, 2009, pp. 684–695.

[31] S. R. Sarangi and K. Murthy, "DUST: a generalized notion of similarity between uncertain time series," in *Proceedings of the 16th ACM international conference on Knowledge discovery and data mining*, 2010, pp. 383–392.

[32] G. He, L. Chen, C. Zeng, Q. Zheng, and G. Zhou, "Probabilistic skyline queries on uncertain time series," *Neurocomputing*, vol. 191, pp. 224–237, 2016.

[33] J. Dai, B. Yang, C. Guo, and Z. Ding, "Personalized route recommendation using big trajectory data," in *Proceedings of the 31th IEEE International Conference of Data Engineering*, 2015, pp. 543–554.

[34] D. Delling, A. V. Goldberg, M. Goldszmidt, J. Krumm, K. Talwar, and R. F. Werneck, "Navigation made personal: inferring driving preferences from gps traces," in *Proceedings of the 25th ACM International Conference on Advances in Geographic Information Systems*, 2015, pp. 31:1–31:9.

[35] A. Balteanu, G. Jossé, and M. Schubert, "Mining driving preferences in multi-cost networks," in *Proceedings of the 13th International Symposium on Spatial and Temporal Databases*, 2013, pp. 74–91.

[36] S. Aljubayrin, B. Yang, C. S. Jensen, and R. Zhang, "Finding non-dominated paths in uncertain road networks," in *Proceedings of the 24th ACM International Conference on Advances in Geographic Information Systems*, 2016, pp. 15:1–15:10.

[37] C. Guo, B. Yang, O. Andersen, C. S. Jensen, and K. Torp, "Ecomark 2.0: empowering eco-routing with vehicular environmental models and actual vehicle fuel consumption data," *GeoInformatica*, vol. 19, no. 3, pp. 567–599, 2015.

[38] C. Guo, Y. Ma, B. Yang, C. S. Jensen, and M. Kaul, "Ecomark: evaluating models of vehicular environmental impact," in *Proceedings of the 20th ACM International Conference on Advances in Geographic Information Systems*, 2012, pp. 269–278.

[39] O. Andersen, C. S. Jensen, K. Torp, and B. Yang, "Ecotour: Reducing the environmental footprint of vehicles using eco-routes," in *Proceedings of the 14th IEEE International Conference on Mobile Data Management*, 2013, pp. 338–340.

Paper C

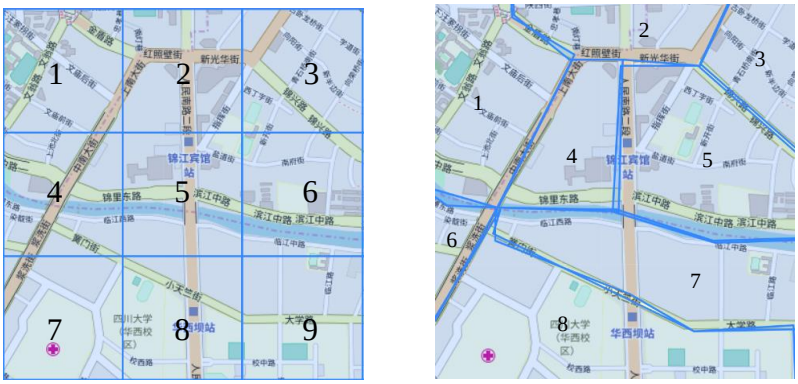
Stochastic Origin-Destination Matrix Forecasting Using Dual-Stage Graph Convolutional, Recurrent Neural Networks

Jilin Hu, Chenjuan Guo, Bin Yang, Christian S. Jensen, Lu Chen

The paper has been submitted for publication.

Abstract

Given a partitioning of a road network into regions, an origin-destination (OD) matrix records the cost of travel between any pair of regions, example costs being travel speed or greenhouse gas emission. OD matrices have a broad range of uses in transportation and logistics. We consider an increasingly pertinent setting where a set of vehicle trajectories is used for instantiating historical OD matrices. As a cost such as travel speed varies over time, e.g., due to varying congestion, matrices are created for different time intervals during a day, e.g., one matrix for every 15 minutes. A cost is modeled as a distribution because different vehicles, as captured by the set of trajectories, may travel at different speeds during the same time interval, e.g., due to different driving styles or different waiting times at intersections. The resulting historical OD matrices are likely to be sparse. We address the problem of forecasting complete near future OD matrices from such sparse historical OD matrices. To solve the problem, we propose a generic learning framework that employs matrix factorization and graph convolutional neural networks to contend with data sparseness and that captures temporal dynamics via recurrent neural networks. Empirical studies using two taxi trajectory data sets offer detailed insight into the properties of the framework and indicate that it is effective.



(a) Grid-based Partition (b) Road-based Partition

Fig. C.1: Partition a City into Regions

C.1 Introduction

Origin-destination (OD) matrices [1, 2] are applied widely in location based services and online map services (e.g., transportation-as-a-service), where OD matrices are used for the scheduling of trips, for computing payments for completed trips, and for estimating arrival times. For example, Google Maps[1] and ESRI ArcGIS Online[2] offer OD matrix services to help developers to develop location based applications. Further, increased urbanization contributes to making it increasingly relevant to capture and study city-wide traffic conditions. OD matrices may also be applied for this purpose.

To use OD-matrices, a city is partitioned into *regions*, and a day is partitioned into *intervals*. Each interval is assigned its own an OD-matrix, and an element (i, j) in matrix described the attribute (e.g., travel speed, fuel consumption, or travel demand) of travel from region i to region j during the interval that the matrix represents. Different approaches can be applied to partition a road network, e.g., using a uniform grid or using major roads, as exemplified in Figure C.1. In this paper, we focus on speed matrices. However, the proposed techniques can be applied on other travel attributes or costs, such as travel time, fuel consumption, and travel demand.

As part of the increasing digitization of transportation, increasingly vast volumes of vehicle trajectory, trajectory data are becoming available. We aim to exploit such data for composing OD matrices. Specifically, an element (i, j) of a speed matrix for a given time interval can be instantiated from the speeds observed in trajectories that went from region i to region j during the relevant time interval.

We consider *stochastic OD matrices* where the elements represent uncertain costs by meaning of cost distributions rather than deterministic, single-valued costs. The use of distribution models reality better and enables more reliable decision-making. For example, element (i, j) has a speed histogram $\{([10, 20), 0.5), ([20, 40), 0.3), ([40, 60], 0.2)\}$,

[1] `https://tinyurl.com/7vmtk4y`
[2] `https://tinyurl.com/ydhq765h`

meaning that the probability of traveling speed from region i to region j at 0-20 km/h is 0.5, at $[20, 30)$ is 0.3, and at $[30, 40)$ is 0.2, respectively. If a passenger needs to go from his home in region i to catch a flight in an airport in region j, and the shortest path from his home to the airport is 20 km, then we are able to derive a travel time (minutes) distribution: $\{([30, 40], 0.5), (40, 60], 0.3), (60, 120], 0.2)\}$. Therefore, the passenger needs to reserve at least 120 minutes for not being late. However, when only using average speed to derive an average travel time of 54 minutes, it makes the passenger runs into a risk of missing the flight.

We address the problem of *stochastic origin-destination matrix forecasting*—based on historical stochastic OD-matrices, we predict future OD-matrices. Figure C.2 shows a specific example: given stochastic OD-matrices for 3 historical intervals $T^{(t-2)}$, $T^{(t-1)}$, and $T^{(t)}$, we aim at predicting the stochastic OD-matrices for the 3 future intervals $T^{(t+1)}$, $T^{(t+2)}$, and $T^{(t+3)}$.

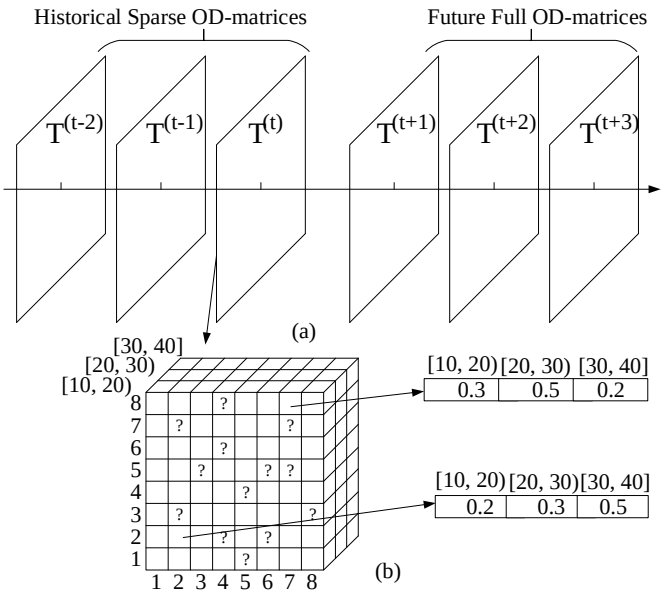


Fig. C.2: Stochastic Origin-Destination Matrix Forecasting.

Here, a stochastic OD-matrix is represented as a 3-dimensional tensor, where the first dimension represents source regions, the second dimension represents destination regions, and the third dimension represents cost ranges. For example, Figure C.2(b) shows the stochastic OD-matrix for interval $T^{(t)}$, which is represented as a $\mathcal{R}^{8\times8\times3}$ tensor with 8 source regions, 8 destination regions, and 3 speed (km/h) ranges [10, 20), [20-30), and [30-40]. Element (8, 8) in the OD-matrix is a vector (0.3, 0.5, 0.2), meaning that, when traveling within region 8, the travel speed histogram is $\{([10, 20), 0.3), ([20, 30), 0.5), ([30, 40], 0.2)\}$.

Solving the stochastic OD-matrix forecasting problem is non-trivial as it is necessary to contend with two difficult challenges.

(1) **Data Sparseness.** To instantiate a stochastic OD-matrix in an interval using trajectories, we need to have sufficient trajectories for each region pair during the interval. However, even massive trajectory data sets are often spatially and temporally skewed [3–7], making it almost impossible to cover all region pairs for all intervals.

For example, the New York City taxi data set[3] we use in experiments has more than 29 million trips from November and December 2013. Yet, this massive trip set only covers 65% of all "taxizone" pairs in Manhattan, the most densely traversed region in New York City. If we further split the data set according to the temporal dimension, e.g., into 15-min intervals, the spareness problem becomes even more severe.

The data sparseness in turn results in sparse historical stochastic OD-matrices, where some elements are empty (e.g., those elements with "?" in Figure C.2(b)). Yet, decision making requires full OD-matrices. The challenge is how to use sparse historical OD-matrices to predict full future OD-matrices.

(2) **Spatio-temporal Correlations.** Traffic is often spatio-temporally correlated—if a region is congested during a time interval, its neighboring regions are also likely to be congested in subsequent intervals. Thus, to predict accurate OD-matrices, we need to account for such spatio-temporal correlations. However, the OD-matrices themselves do not necessarily capture spatial proximity. No matter which partition method is used, we cannot always guarantee that two geographically adjacent regions are represented by adjacent rows and columns in the matrix. For example, in Figure C.1(a), regions 1 and 4 are geographically adjacent, but they are not adjacent in the OD matrices; in Figure C.1(b), regions 4 and 7 are adjacent but they are again not adjacent in the OD matrices. This calls for a separate mechanism that is able to take into account the geographical proximity of regions.

We propose a data-driven, end-to-end deep learning framework to forecast stochastic OD matrices that aims to effectively address the challenges caused by data sparseness and spatio-temporal correlations. First, to address the data sparseness challenge, we factorize a sparse OD matrix into two small dense matrices with latent features of the source regions and the destination regions, respectively. Second, we model the spatial relationships among source regions and among destination regions using two graphs, respectively. Then, we employ two graph covolutional, recurrent neural networks (GR) on the two dense matrices to capture the spatio-temporal correlations. Finally, the two GRs predict two dense, small matrices. We apply the multiplication to the two dense, small matrices to obtain a full predicted OD-matrix.

To the best of our knowledge, this is the first study of stochastic OD matrix forecasting that contends with data sparseness and spatio-temporal correlations. The study makes four contributions. First, it formalizes the stochastic OD matrix forecasting problem. Second, it proposes a generic framework to solve the problem based on matrix factorization and recurrent neural networks. Third, it extends the framework by embedding spatial correlations using two graph convolutional neural networks. Fourth, it encompasses an extensive experiments using two real-world taxi datasets that offers insight into the effectiveness of the framework.

The remainder of the paper is organized as follows. Section 2 covers related works. Section 3 defines the setting and formalizes the problem. Section 4 introduces

[3] `http://www.nyc.gov/html/tlc/html/technology/raw_data.shtml`

a basic framework and Section 5 presents an advanced framework. Section 6 reports experiments and Section 7 concludes.

C.2 Related Work

C.2.1 Travel Cost Forecasting

We consider three types of travel cost forecasting methods in turn: segment-based methods [8–13], path-based methods [5, 6, 14–17], and OD-based methods [3, 4, 18].

Segment-based methods focus on predicting the travel costs of individual road segments. For example, by modeling the travel costs of a road segment as a time series, techniques such as time-varying linear regression [8], Markov models [9, 10], and support vector regression [11] can be applied to predict future travel costs. Most such models consider time series from different edges independently. As an exception, the spatio-temporal Hidden Markov model [10] takes into account the correlations among the costs of different edges. Some other studies focus on estimating high-resolution travel costs, such as uncertain costs [12] and personalized costs [13].

Path-based methods focus on predicting the travel costs of paths. A naive approach is to predict the costs of the edges in a path and then aggregate the costs. However, this approach is inaccurate since it ignores the dependencies among the costs of different edges in paths [5, 15]. Other methods [5, 14, 15] use sub-trajectories to capture such dependencies and thus to provide more accurate travel costs for paths. A few studies propose variations of deep neural networks [6, 16, 17] to enable accurate travel-time prediction for paths.

Finally, OD-based methods aim at predicting the travel cost for given OD pairs. Our proposal falls into this category. A simple and efficient baseline [3] is to compute a weighted average over all historical trajectories that represent travel from the origin to the destination in an OD pair. However, it does not address data sparseness, which means that if no data is available for a given OD pair, it cannot provided a prediction. In contrast, our proposal is able to predict full OD-matrices without empty elements based on historical, sparse OD-matrices. A recent study [18] utilizes deep learning and multi-task learning to predict OD travel time while taking into account considers the road network topology and the paths used in the historical trajectories. However, path information may not always be available. An example is the New York taxi data set that we use in the experiments. This reduces the applicability of the model. In contrast, our proposal does not require path information. Further, existing proposals support only deterministic costs, while our proposal also supports stochastic costs.

C.2.2 Graph Convolutional Neural Network

Convolutional Neural Networks (CNNs) have been used successfully in the contexts of images [19], videos [20], speech [21], and taxi supply-demand [22], where the underlying data is represented as a matrix [23, 24]. For example, when representing an image as a matrix, nearby elements, e.g., pixels, share local features, e.g., represent parts of the same object. In contrast, in our setting, an OD-matrix may not satisfy the assumption that helps make CNNs work—two adjacent rows in an OD matrix may

represent two geographically distant regions and may not share any features; and two separated rows in an OD matrix may represent geographically close regions that share many features.

Graph convolutional neural networks (GCNNs) [23, 24] aim to address this challenge. In particular, the geographical relationships among regions can be modeled as a graph, and GCNNs then take into account the graph while learning. One study [25] applies GCNNs to solve semi-supervised classification in the setting of citation networks and knowledge graphs. One study continues to study semi-supervised classification via dual graph convolutional networks [26]. Another study [27] constructs GCNNs together with a Recurrent Neural Network (RNN) to forecast traffic. Both studies consider a setting where only one dimension needs to be modeled as a graph. In contrast, in our study, both dimensions, i.e., the source region dimension and the destination region dimension, need to be modeled as two graphs. An additional, recent study focuses on so-called geomatrix completion which considers a similar setting where two dimensions need to be modelded as two graphs. It uses multi-graph neural networks [28] with RNNs. However, the RNNs in this study are utilized to perform iterations to approximate the geomatrix completion, not to capture temporal dynamics as in our study. To the best of our knowledge, our study is the first that constructs a learning framework involving dual-graph convolution and employing RNNs to forecast the future.

C.3 Preliminaries

C.3.1 OD Stochastic Speed Tensor

A **trip** $\boldsymbol{p}$ is defined as a tuple $\boldsymbol{p} = (o, d, t, l, \tau)$, where o, d denote an origin and a destination, t is a departure time, l represents the trip distance, and τ is the travel time of the trip. Given $\boldsymbol{p}.l$ and $\boldsymbol{p}.\tau$, we derive the average travel speed v of $\boldsymbol{p}$. We use $\mathbb{P}$ to denote a set of historical trips.

To capture the time-dependent traffic, we partition the time domain TI of interest, e.g., a day, into a number of time intervals, e.g., 96 15-min intervals. For each time interval $T_i \in TI$, we obtain the set of historical trips $\mathbb{P}_{T_i}$ from $\mathbb{P}$ whose departure times belong to time interval T_i, i.e., $\mathbb{P}_{T_i} = \{\boldsymbol{p_i} | \boldsymbol{p_i}.t \in T_i \wedge \boldsymbol{p_i} \in \mathbb{P}\}$.

We further partition a city into M regions $\mathbb{V} = \{\mathbb{V}_1, \cdots, \mathbb{V}_M\}$. An **Origin-Destination (OD) pair** is defined as a pair of regions $(\mathbb{V}_o, \mathbb{V}_d)$ where $1 \leq o, d, \leq M$.

Given a time interval T_i, two regions $\mathbb{V}_o$ and $\mathbb{V}_d$, we obtain a trip set $\mathbb{P}_{T_i,\mathbb{V}_o,\mathbb{V}_d} = \{\boldsymbol{p}_i | \boldsymbol{p}_i.o \in \mathbb{V}_o \wedge \boldsymbol{p}_i.d \in \mathbb{V}_d \wedge \boldsymbol{p}_i.t \in T_i\}$, meaning that each trip in $\mathbb{P}_{T_i,\mathbb{V}_o,\mathbb{V}_d}$ starts from region $\mathbb{V}_o$, at a time in interval T_i, and ends at region $\mathbb{V}_d$.

Next, we construct an equi-width histogram $\mathbb{H}_{T_i,\mathbb{V}_o,\mathbb{V}_d}$ to record the stochastic speed of trips in $\mathbb{P}_{T_i,\mathbb{V}_o,\mathbb{V}_d}$. In particular, an equi-width histogram is a set of K bucket-probability pairs, i.e., $\mathbb{H}_{T_i,\mathbb{V}_o,\mathbb{V}_d} = \{(b_j, pr_j)\}$. A bucket $b_j = [v_s, v_e)$ represents the speed range from v_s to v_e, and all buckets have the same range size. Probability pr_j is the probability that the average speed of a trip falls into the range b_j. For example, the speed histogram $\{([0, 20), 0.5), ([20, 40), 0.3), ([40, 60), 0.2)\}$ for $\mathbb{P}_{T_i,\mathbb{V}_o,\mathbb{V}_d}$ means that the probabilities that the average speed (km/h) of a trip in $\mathbb{P}_{T_i,\mathbb{V}_o,\mathbb{V}_d}$ falls into $[0, 20)$, $[20, 40)$, and $[40, 60)$ are 0.5, 0.3, and 0.2, respectively.

Definition C.3.1
Given a time interval T_i, an **OD stochastic speed tensor** is defined as a matrix $\mathbf{M}^{(i)} \in \mathbb{R}^{N \times N' \times K}$, where the first and second dimensions range over the origin and destination regions, respectively, and the third dimension ranges over the stochastic speeds. For generality, the origin and destination regions can be the same or can be different; thus the first and second dimensions have N and N' instances, respectively. The third dimension defines K speed buckets.

$M_{o,d,k}$ represents the element (o, d, k) of tensor $\mathbf{M}^{(i)} \in \mathbb{R}^{N \times N' \times K}$ and represents the probability of trips in $\mathbb{P}_{T_i, \mathbb{V}_o, \mathbb{V}_d}$ traveling at an average speed that falls into the k-th bucket.

Following the example in Figure C.2(b), given a time interval T_i, for origin region 7 and destination region 8, we obtain a stochastic speed of trips as a histogram, in which the first bucket records that the probability of trips, starting at region 7 during time interval T_i and ending at region 8, traveling at an average speed of $[5, 10)$ is 0.3.

As shown in Figure C.2(b), not all cells have a histogram to capture the stochastic speed. Specifically, the cells with question marks have no histograms because no trip records are available for those cells, i.e., $\mathbb{P}_{T_i, \mathbb{V}_o, \mathbb{V}_d} = \emptyset$, so that $\mathbb{H}_{T_i, \mathbb{V}_o, \mathbb{V}_d} = \emptyset$. We refer to such tensor as **sparse OD stochastic speed tensor**.

Given a time interval T_i, we refer to a tensor where each cell has a stochastic speed $\mathbb{H}_{T_i, \mathbb{V}_o, \mathbb{V}_d}$ as a **full OD stochastic speed tensor**.

C.3.2 Problem Definition

Given s sparse OD stochastic speed tensors $\mathbf{M}^{(t-s+1)}$, ..., $\mathbf{M}^{(t)}$ during s historical time intervals $T^{(t-s+1)}$, ..., $T^{(t)}$, we aim to predict the stochastic speeds for the next h time intervals $T^{(t+1)}$, ..., $T^{(t+h)}$ in the form of h full OD stochastic speed tensors $\mathbf{M}^{(t+1)}$, ..., $\mathbf{M}^{(t+h)}$ by learning the following function f.

$$f : [\mathbf{M}^{(t-s+1)}, \ldots, \mathbf{M}^{(t)}] \rightarrow [\mathbf{M}^{(t+1)}, \ldots, \mathbf{M}^{(t+h)}]$$

C.4 Basic Stochastic Speed Forecasting

C.4.1 Framework and Intuition

Figure C.3 shows the basic framework for forecasting stochastic speeds, which consists of three steps: *Factorization*, *Forecasting*, and *Recovering*.

For the historical time intervals $T^{(t-s+1)}$, ..., $T^{(t)}$, we have sparse OD stochastic speed tensors $\mathbf{M}^{(t-s+1)}$, ..., $\mathbf{M}^{(t)}$. We factorize each stochastic speed tensor $\mathbf{M}^{(t-i+1)} \in \mathbb{R}^{N \times N' \times K}$, where $i \in [1, s]$ into two smaller tensors $\mathbf{R}^{(t-i+1)} \in \mathbb{R}^{N \times \beta \times K}$ and $\mathbf{C}^{(t-i+1)} \in \mathbb{R}^{\beta \times N' \times K}$, where $\beta \ll N, N'$. The aim is to use $\mathbf{R}^{(t-i+1)}$ and $\mathbf{C}^{(t-i+1)}$ to approximate $\mathbf{M}^{(t-i+1)}$. Here $\mathbf{R}^{(t-i+1)}$ and $\mathbf{C}^{(t-i+1)}$ model the correlated features of stochastic speeds among origin regions and among destination regions, respectively. And it is intuitive to assume that stochastic speeds among origin regions and among destination regions share correlated features, as traffic in a region affects the traffic in its nearby regions.

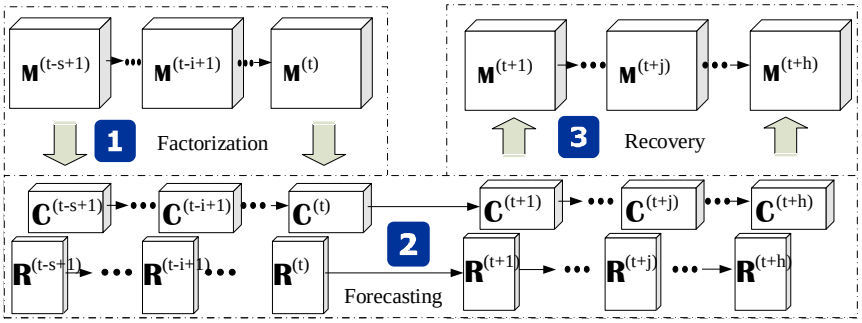


Fig. C.3: Framework Overview.

The factorization is supported by the intuition underlying low-rank matrix approximation [28–31]. Since $\mathbf{M}^{(t-i+1)}$ is a sparse tensor, we aim to find a low-rank tensor $\mathbf{M}'^{(t-i+1)}$ to approximate $\mathbf{M}^{(t-i+1)}$. When carrying out the approximation, we assume that the rank of $\mathbf{M}'^{(t-i+1)}$ is at most β and that it can be factorized as $\mathbf{M}'^{(t-i+1)} = \mathbf{R}^{(t-i+1)} \times \mathbf{C}^{(t-i+1)}$. Then, the problem of using $\mathbf{M}'^{(t-i+1)}$ to approximate $\mathbf{M}^{(t-i+1)}$ can be formulated as the problem of minimizing the following loss function.

$$\min_{\mathbf{R}^{(x)},\mathbf{C}^{(x)}} ||\mathbf{R}^{(x)}||_F^2 + ||\mathbf{C}^{(x)}||_F^2 + \frac{\mu}{2}||\Omega \circ (\mathbf{R}^{(x)}\mathbf{C}^{(x)} - \mathbf{M}^{(x)})||_F^2, \tag{C.1}$$

where $x = t - i + 1$, $||\cdot||_F$ denotes the Frobenius norm, $\Omega^{(x)}_{o,d,k} = 1$ if the element (o, d) of $M^{(x)}$ is not empty, and $\circ$ is the element-wise tensor multiplication.

Next, we consider $\mathbf{R}^{(t-s+1)}, \ldots, \mathbf{R}^{(t)}$ as an input sequence, from which we capture the temporal correlations among the origin regions of $\mathbf{M}^{(t-s+1)}, \ldots, \mathbf{M}^{(t)}$. We feed this input sequence into a sequence-to-sequence RNN model [32] to *forecast* an output sequence that represents the shared features among the origin regions in the future.

We apply a similar procedure to $\mathbf{C}^{(t+1)}, \ldots, \mathbf{C}^{(t+h)}$ to *forecast* an output sequence that represents the shared features among destination regions in the future.

Finally, we *recover* $\mathbf{M}^{(t+j)}$ as a full OD stochastic speed tensor from $\mathbf{R}^{(t+j)}$ and $\mathbf{C}^{(t+j)}$, $j \in [1, h]$. Since we obtain the predictions $\mathbf{R}^{(t+j)}$ and $\mathbf{C}^{(t+j)}$ from the historical $\mathbf{R}^{(t-i+1)}$ and $\mathbf{C}^{(t-i+1)}$, $i \in [1, s]$, the intuition of Equation C.1 also applies when reconstructing $\mathbf{M}^{(t+j)}$.

C.4.2 Factorization

Given an input sparse OD stochastic tensors $\mathbf{M}^{(t-i+1)} \in \mathbb{R}^{N \times N' \times K}$ at interval $T^{(t-i+1)}$, where $i \in [1, s]$, we proceed to describe the method for factorizing $\mathbf{M}^{(t-i+1)}$ into $\mathbf{R}^{(t-i+1)}$ and $\mathbf{C}^{(t-i+1)}$, which are able to capture the correlated features of stochastic speed among origin and destination regions, respectively.

We first flatten $\mathbf{M}^{(t-i+1)}$ into a vector $\boldsymbol{f}^{(t-i+1)} \in \mathbb{R}^l$, where $l = N \cdot N' \cdot K$, from which we generate two small factorization vectors, $\boldsymbol{c}^{(t-i+1)} \in \mathbb{R}^{N' \cdot K \cdot \beta}$ and $\boldsymbol{r}^{(t-i+1)} \in$

$\mathbb{R}^{N\cdot K\cdot\beta}$ via a fully-connected neural network layer (FC layer).

$$\boldsymbol{r}^{(t-i+1)} = \text{relu}(\boldsymbol{F_r} \times \boldsymbol{f}^{(t-i+1)} + \boldsymbol{b}_r) \tag{C.2}$$

$$\boldsymbol{c}^{(t-i+1)} = \text{relu}(\boldsymbol{F_c} \times \boldsymbol{f}^{(t-i+1)} + \boldsymbol{b}_c) \tag{C.3}$$

Here $\boldsymbol{F_r} \in \mathbb{R}^{(N\cdot K\cdot\beta)\times l}$ and $\boldsymbol{F_c} \in \mathbb{R}^{(N'\cdot K\cdot\beta)\times l}$ are parameter matrices, where β is a hyper-parameter to be set; $\boldsymbol{b}_r \in \mathbb{R}^{(N\cdot K\cdot\beta)}$ and $\boldsymbol{b}_c \in \mathbb{R}^{(N'\cdot K\cdot\beta)}$ are bias vectors; and $\text{relu}(\cdot)$ is the relu activation function.

Next, we reorganize the factorization vectors $\boldsymbol{r}^{(t-i+1)}$ and $\boldsymbol{c}^{(t-i+1)}$ into factorization tensors $\mathbf{R}^{(t-i+1)} \in \mathbb{R}^{N\times\beta\times K}$ and $\mathbf{C}^{(t-i+1)} \in \mathbb{R}^{\beta\times N'\times K}$, respectively.

C.4.3 Forecasting

Given historical time intervals $T^{(t-s+1)}, \ldots, T^{(t)}$, we learn the temporal correlations of $\mathbf{M}^{(t-s+1)}, \ldots, \mathbf{M}^{(t)}$ from the temporal correlations among origin regions $\mathbf{R}^{(t-s+1)}, \ldots, \mathbf{R}^{(t)}$ and the temporal correlation among destination regions $\mathbf{C}^{(t-s+1)}, \ldots, \mathbf{C}^{(t)}$.

Based on $\mathbf{R}^{(t-s+1)}, \ldots, \mathbf{R}^{(t)}$, we use a sequence-to-sequence RNN model [32] to forecast $\widehat{\mathbf{R}}^{(t+1)}, \ldots, \widehat{\mathbf{R}}^{(t+h)}$ for the future time intervals $T_{t+1}, \ldots, T_{t+h}$. In particular, we apply Gated Recurrent Units (GRUs) in the RNN architecture, since these can capture temporal correlations well by using gate units well and also offer high efficiency [33, 34]. The process is presented as follows.

$$\widehat{\mathbf{R}}^{(t+1)}, \ldots, \widehat{\mathbf{R}}^{(t+h)} = \mathit{seq2seqGRU}(\mathbf{R}^{(t-s+1)}, \ldots, \mathbf{R}^{(t)}). \tag{C.4}$$

A similar procedure is applied to obtain $\widehat{\mathbf{C}}^{(t+1)}, \ldots, \widehat{\mathbf{C}}^{(t+h)}$ from $\mathbf{C}^{(t-s+1)}, \ldots, \mathbf{C}^{(t)}$.

C.4.4 Recovery

Given predicted tensors $\widehat{\mathbf{R}}^{(t+j)} \in \mathbb{R}^{N\times\beta\times K}$ and $\widehat{\mathbf{C}}^{(t+j)} \in \mathbb{R}^{\beta\times N'\times K}$ for a future time interval $T^{(t+j)}$, with $j \in [1, h]$, we proceed to describe how to transform $\widehat{\mathbf{R}}^{(t+j)}$ and $\widehat{\mathbf{C}}^{(t+j)}$ into a full OD stochastic speed tensor $\mathbf{M}^{(t+j)} \in \mathbb{R}^{N\times N'\times K}$.

First, we slice each of $\widehat{\mathbf{R}}^{(t+j)}$ and $\widehat{\mathbf{C}}^{(t+j)}$ by the speed bucket dimension into K matrices. Specifically, we have $\mathit{slice}(\widehat{\mathbf{R}}^{(t+j)}) = \{\widehat{\mathbf{R}}^{(t+j)}_{:,:,1} \cdots, \widehat{\mathbf{R}}^{(t+j)}_{:,:,K}\}$ and $\mathit{slice}(\widehat{\mathbf{C}}^{(t+j)}) = \{\widehat{\mathbf{C}}^{(t+j)}_{:,:,1} \cdots, \mathbf{C}^{(t+j)}_{:,:,K}\}$, where $\widehat{\mathbf{R}}^{(t+j)}_{:,:,k} \in \mathbb{R}^{N\times\beta}$ and $\widehat{\mathbf{C}}^{(t+j)}_{:,:,k} \in \mathbb{R}^{\beta\times N'}$, $k \in [1, K]$.

Next, we conduct a matrix multiplication as follows.

$$\widetilde{\boldsymbol{M}}^{(t+j)}_k = \widehat{\mathbf{R}}^{(t+j)}_{:,:,k} \times (\widehat{\mathbf{C}}^{(t+j)}_{:,:,k}), \tag{C.5}$$

where $\widetilde{\boldsymbol{M}}^{(t+j)}_k \in \mathbb{R}^{N\times N'}$, $k \in [1, K]$.

Finally, we are able to construct a tenor $\widetilde{\mathbf{M}}^{(t+j)} \in \mathbb{R}^{N\times N'\times K}$ by combining a total of K matrices, i.e., $\widetilde{\mathbf{M}}^{(t+j)}_{:,:,k} = \widetilde{\boldsymbol{M}}^{(t+j)}_k$, $k \in [1, K]$. Now, $\widetilde{\mathbf{M}}^{(t+j)}$ is a full tensor where each element has a value.

A histogram $\widehat{\mathbf{M}}_{o,d,:}^{(t+j)} \in \mathbb{R}^{1\times K}$ must meet two requirements to be a meaningful histogram: (1) $\widehat{\mathbf{M}}_{o,d,k}^{(t+j)} \in [0,1]$, $k \in [1,K]$, meaning that the probability of a speed falling into the k-th bucket for each OD pair (o,d) must between 0 and 1; and (2) $\sum_{k=1}^{K} \widehat{\mathbf{M}}_{o,d,k}^{(t+j)} = 1$, meaning that the probability of a speed falling into all K buckets for each (o,d) must equal 1.

To achieve this, we apply a softmax function to normalize values in $\widetilde{\mathbf{M}}^{(t+j)}$ into $\widehat{\mathbf{M}}_{o,d,:}^{(t+j)}$ that satisfies the histogram requirements.

$$\widehat{\mathbf{M}}_{o,d,:}^{(t+j)} = softmax(\widetilde{\mathbf{M}}_{o,d,:}^{(t+j)}), \forall o \in [1,N], \forall d \in [1,N']. \tag{C.6}$$

Thus, we obtain h meaningful full OD stochastic speed tensors for the future time intervals $T_{t+1},\ldots,T_{t+h}$ as the output of the recovery process: $\widehat{\mathbf{M}}^{(t+1)},\ldots,\widehat{\mathbf{M}}^{(t+h)}$.

C.4.5 Loss Function

The loss function is defined as the error between the recovered future tensor and the ground-truth future tensor.

$$\begin{aligned}\ell(\boldsymbol{F},\boldsymbol{b}) = \sum_{j=1}^{h}[\lambda||\widehat{\mathbf{R}}^{(t+j)}||_F^2 + \lambda||\widehat{\mathbf{C}}^{(t+j)}||_F^2 + \\ ||\Omega^{(t+j)} \circ (\mathbf{M}^{(t+j)} - \widehat{\mathbf{M}}^{(t+j)})||_F^2],\end{aligned} \tag{C.7}$$

where $\boldsymbol{F}$ and $\boldsymbol{b}$ represent the training parameters in the framework, and λ is a regularization parameter. Further, $\Omega^{(t+j)} \in \mathbb{R}^{N\times N'\times K}$ is an indication tensor, where $\Omega_{o,d,k}^{(t+j)} = 1$ if the OD pair (o,d) is not empty in the $t+j$'th future interval. Note that although we aim to predict full tensors, the ground truth tensors are sparse, so we compute the errors taking only into account the non-empty elements in the ground truth tensors. Next, $\circ$ is element-wise multiplication, $\widehat{\mathbf{M}}^{(t+j)}$ and $\mathbf{M}^{(t+j)}$ are predicted and ground truth tensors, respectively, $||\cdot||_F$ is the Frobenius-norm.

C.5 Forecast with Spatial Dependency

To improve forecast accuracy, we proceed to integrate spatial dependency into our framework in two different stages. First, in the factorization step, we apply graph convolutional neural networks to perform feature encoding for origin and destination dimensions, respectively. Second, in the forecasting step, we integrate graph convolutional with RNNs to capture spatio-temporal correlations.

C.5.1 Spatial Factorization

As in Section C.4.2, we aim to factorize tensor $\mathbf{M}^{(t-i+1)}$ during interval T_{t-i+1}, $i \in [1,s]$, into two smaller tensors $\mathbf{C}^{(t-i+1)}$ and $\mathbf{R}^{(t-i+1)}$. In Section C.4.2, $\mathbf{M}^{(t-i+1)}$ is simply flattened and followed by a fully-connected layer to construct $\mathbf{C}^{(t-i+1)}$ and

$\mathbf{R}^{(t-i+1)}$. This process does not take spatial correlations among the origin regions and among the destination regions into account, although such correlations are likely to exist. To accommodate spatial correlations, we first capture spatial correlations among origin and destination regions; then we use the captured spatial correlations to conduct factorization.

Spatial Correlation

We leverage the notion of a proximity matrix [35] to capture spatial correlations. We proceed to present the idea using origin regions as an example, which also applies to destination regions in a similar manner.

Given $\mathbf{M}^{(t-i+1)} \in \mathbb{R}^{N \times N' \times K}$, we have N origin regions, from which we build an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ to show region connections. Specifically, $\boldsymbol{A}_{u,v} = 1$ means that regions $\mathbb{V}_u$ and $\mathbb{V}_v$ are adjacent; otherwise, $\boldsymbol{A}_{u,v} = 0$.

We construct a weighted proximity matrix $\boldsymbol{W}^{(\alpha,\sigma)} \in \mathbb{R}^{N \times N}$ from $\boldsymbol{A}$ that describes the proximity between regions $\mathbb{V}_u$ and $\mathbb{V}_v$ and is parameterized by adjacency hops α and standard deviation σ. Specifically, if $\mathbb{V}_v$ can be reached from $\mathbb{V}_u$ in α adjacency hops using $\boldsymbol{A}$, $\boldsymbol{W}_{u,v}^{(\alpha,\sigma)} = e^{-x^2/\sigma^2}$, where x is the distance between the centroid of $\mathbb{V}_u$ and $\mathbb{V}_v$; otherwise $\boldsymbol{W}_{u,v}^{(\alpha,\sigma)} = 0$. In the experiments, we study the effect of α and σ (see Section C.6.2). The proximity matrix $\boldsymbol{W}^{(\alpha,\sigma)}$ is symmetric and non-negative.

The adjacency matrices for the source regions and destination regions may be different or the same. Consider two scenarios. First, we use OD matrices to model the travel costs within a city. In this case, the source regions and the destination regions are the same, and thus the two adjacency matrices are the same. Second, we may use OD matrices to model the travel costs between two different cities. Then, the source regions and the destination regions are in different cities. Thus, we need two different adjacency matrices. To avoid confusion, we use $\boldsymbol{W}$ and $\boldsymbol{W}'$ represent the adjacency matrices for source regions and destination regions, respectively.

Factorization

We proceed to show the factorization procedure. Specifically, we show how to obtain $\mathbf{R}^{(t-i+1)}$ from $\mathbf{M}^{(t-i+1)}$. The same procedure can be applied to obtain $\mathbf{C}^{(t-i+1)}$.

As shown in Figure C.4(a), we first slice $\mathbf{M}^{(t-i+1)} \in \mathbb{R}^{N \times N' \times K}$ by the origin region dimension into N matrices, i.e., $slice(\mathbf{M}^{(t-i+1)}) = [\mathbf{M}_{1,:,:}^{(t-i+1)}, \cdots, \mathbf{M}_{N,:,:}^{(t-i+1)}]$. Each of the sliced matrix is then applied with a GCNN operation. Accordingly, we obtain the GCNN output as $[\mathbf{R}_{1,:,:}^{(t-i+1)}, \cdots, \mathbf{R}_{N,:,:}^{(t-i+1)}]$. We then concatenate this to obtain $\mathbf{R}^{(t-i+1)} \in \mathbb{R}^{N \times \beta' \times K}$.

Figure C.4(b) shows a GCNN operation on a sliced matrix $\mathbf{M}_{j,:,:}^{(t-i+1)}$ $j \in [1, N]$, which transforms $\mathbf{M}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{K \times N'}$ into $\mathbf{R}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{K \times \beta'}$ via *Filtering* and *Pooling*.

Filtering: Given $\mathbf{M}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{K \times N'}$, we apply Q graph convolutional filters, which take into account the destination region adjacency matrix $\boldsymbol{W}$, to generate $\widetilde{\mathbf{R}}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{N' \times Q}$ that captures the correlated features among destination regions.

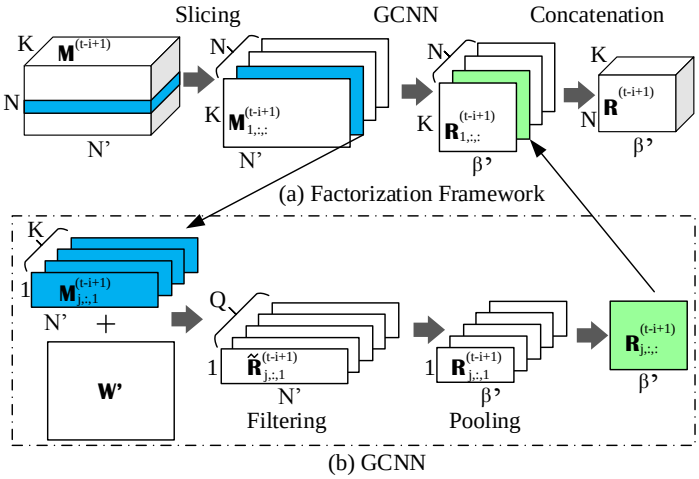


Fig. C.4: Spatial Factorization for $\boldsymbol{R}$.

We first slice $\mathbf{M}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{K \times N'}$ into K vectors $[\mathbf{M}_{j,:,1}^{(t-i+1)}, \cdots, \mathbf{M}_{j,:,K}^{(t-i+1)}]$, where vector $\mathbf{M}_{j,:,k}^{(t-i+1)} \in \mathbb{R}^{N'}$, $k \in [1, K]$, represents the probability of speeds falling into the k-th speed bucket when traveling from origin region $\mathbb{V}_j$ to all destination regions.

Next, we use a specific graph convolutional filter, namely Cheby-Net [23], due to its high accuracy and efficiency, on each vector $\mathbf{M}_{j,:,k}^{(t-i+1)}$. Specifically, before conducting actual convolutions, we compute $\boldsymbol{T}_k^{(t-i+1)} = [\boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_S]$, where $\boldsymbol{t}_s \in \mathbb{R}^{N'}$, $s \in [1, S]$, from $\mathbf{M}_{j,:,k}^{(t-i+1)}$. Here, $\boldsymbol{t}_1 = \mathbf{M}_{j,:,k}^{(t-i+1)}$, $\boldsymbol{t}_2 = \hat{\boldsymbol{L}} \times \mathbf{M}_{j,:,k}^{(t-i+1)}$, and $T_s = 2\hat{\boldsymbol{L}} \times T_{s-1} - T_{s-2}$ when $s > 2$, where $\hat{\boldsymbol{L}} = 2\boldsymbol{L}/\lambda_{max} - \boldsymbol{I}$ is a scaled Laplacian matrix and where $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}'$ is the Laplacian matrix and λ_{max} is the maximum eigenvalue of $\boldsymbol{L}$. Here, we use destination adjacency matrix $\boldsymbol{W}'$ because $\mathbf{M}_{j,:,k}^{(t-i+1)}$ represents the speed from source region j to all destination regions and we use $\boldsymbol{W}'$ to capture the spatial correlation among destination regions. After the whole computation, we get $\boldsymbol{T}_k \in \mathbb{R}^{N' \times S}$ as the encoded features for the k-th bucket while considering the spatial correlations among destination regions.

Then we proceed to apply Q filters to $\boldsymbol{T}_k$. Each filter is a vector $\mathbf{G}_q \in \mathbb{R}^S$, where $q \in [1, Q]$. We apply each filter to all $\{\boldsymbol{T}_k^{(t-i+1)}\}$, $\forall k \in [1, K]$, and then the sum is used as the output of the filter.

$$\widetilde{\mathbf{R}}_{j,:,q}^{(t-i+1)} = \mathbf{G}_q \otimes \mathbf{M}_{j,:,:}^{(t-i+1)} = \sum_{k=1}^{K} \left(\epsilon(\boldsymbol{T}_k^{(t-i+1)} \times \mathbf{G}_q + \boldsymbol{b}_q\right), \tag{C.8}$$

where $\otimes$ is the Cheby-Net graph convolution operation, $\boldsymbol{b}_q \in \mathbb{R}^{N'}$ is a bias vector, and $\epsilon(\cdot)$ is a non-linear activate function.

Finally, we arrange the results obtained from all Q filters as $\widetilde{\mathbf{R}}_{j,:,:}^{(t-i+1)} = [\widetilde{\mathbf{R}}_{j,:,1}^{(t-i+1)}, \ldots, \widetilde{\mathbf{R}}_{j,:,Q}^{(t-i+1)}]$, where $\widetilde{\mathbf{R}}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{Q \times N'}$.

Pooling: To further condense the features and to construct the final factorizations, we apply geometrical pooling [23] to $\widetilde{\mathbf{R}}_{j,:,:}^{(t-i+1)}$ over the destination region dimension to obtain $\mathbf{R}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{Q \times \beta'}$, where $\beta' = \frac{N'}{p}$ and p are the pooling and stride size, respectively. This process is shown as follows.

$$\mathbf{R}_{j,:,:}^{(t-i+1)} = \mathrm{P}(\widetilde{\mathbf{R}}_{j,:,:}^{(t-i+1)}), \tag{C.9}$$

where $\mathrm{P}(\cdot)$ is the pooling function that can be either max pooling or average pooling.

Since the pooling operation requires meaningful neighborhood relationships, we identify spatial clusters of destination regions. For example, in Figure C.1(b), if we use the order of ascending region ids, i.e., (1, 2, 3, 4, 5, 6, 7, 8) to conduct pooling with a pooling size of 2, then regions 3 and 4 are pooled together. However, regions 3 and 4 are not neighbors, so this procedure may yield inferior features that may in turn yield undesired results. Instead, if we identify clusters of regions, we are able to produce a new order, e.g., (6, 1, 2, 3, 5, 4, 7, 8). When again using a pooling size of 2, each pool contains neighboring regions.

The GCNN process, including filtering and pooling, is repeated several times with different numbers of filters Q and pooling stride size p. Eventually, we set $Q = K$ and get $\mathbf{R}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{\beta' \times K}$.

As shown in Figure C.4(b), the last operation is concatenation. We slice $\mathbf{M}^{(t-i+1)}$ by the origin region dimension into N matrices $[\mathbf{M}_{1,:,:}^{(t-i+1)}, \ldots, \mathbf{M}_{N,:,:}^{(t-i+1)}]$ and apply GCNN to each of them to obtain $[\mathbf{R}_{1,:,:}^{(t-i+1)}, \ldots, \mathbf{R}_{N,:,:}^{(t-i+1)}]$, where each $\mathbf{R}_{j,:,:}^{(t-i+1)} \in \mathbb{R}^{\beta' \times K}$. We then concatenate the $\mathbf{R}_{j,:,:}^{(t-i+1)}$, $j \in [1, N]$, to obtain $\mathbf{R}^{(t-i+1)} \in \mathbb{R}^{N \times \beta' \times K}$.

The same procedure can be applied to obtain $\mathbf{C}^{(t-i+1)}$ where we need to change $\boldsymbol{W}'$ to $\boldsymbol{W}$ when conducting the graph convolution.

C.5.2 Spatial Forecasting

To model temporal dynamics while keeping the spatial correlations in RNNs, we combine Cheby-Net based graph convolution with RNNs, yielding CNRNNs. Intuitively, we follow the structure of gated recurrent units while replacing the traditional fully connected layer by a Cheby-Net based graph convolution layer. Separate CNRNNs are employed to process $\mathbf{R}^{(t)}$ and $\mathbf{C}^{(t)}$.

Taking the source region dimension as an example, a CNRNN takes as input $\mathbf{R}^{(t)}$ at time interval $T^{(t)}$, and it predicts $\widehat{\mathbf{R}}^{(t+1)}$ for the future time interval $T^{(t+1)}$. This procedure is formulated as follows.

$$\mathbf{S}^{(t+1)} = \sigma(\mathbf{G}_{\mathbf{S}} \otimes [\mathbf{H}^{(t)} : \mathbf{R}^{(t)}] + \boldsymbol{b}_{\mathbf{S}}) \tag{C.10}$$

$$\mathbf{U}^{(t+1)} = \sigma(\mathbf{G}_{\mathbf{U}} \otimes [\mathbf{H}^{(t)} : \mathbf{R}^{(t)}] + \boldsymbol{b}_{\mathbf{U}}) \tag{C.11}$$

$$\mathbf{H}^{(t+1)} = \tanh(\mathbf{G}_{\mathbf{H}} \otimes [\mathbf{R}^{(t)} : (\mathbf{S}^{(t+1)} \circ \mathbf{H}^{(t)})] + \boldsymbol{b}_{\mathbf{H}}) \tag{C.12}$$

$$\widehat{\mathbf{R}}^{(t+1)} = \mathbf{U}^{(t+1)} \circ \mathbf{R}^{(t)} + (1 - \mathbf{U}^{(t+1)}) \circ \mathbf{H}^{(t+1)} \tag{C.13}$$

where $\mathbf{G_S}$, $\mathbf{G_U}$, and $\mathbf{G_H}$ are graph convolution filters; $\mathbf{R}^{(t)}$ and $\widehat{\mathbf{R}}^{(t+1)}$ are the input and output of a CNRNN cell at time interval $T^{(t)}$, respectively; $\mathbf{S}^{(t)}$ and $\mathbf{U}^{(t)}$ are the reset and update gates, respectively; $\otimes$ denotes the graph convolution which defined in Equation C.8, and here the graph convolution should take into account source adjacency matrix $\boldsymbol{W}$ since $\mathbf{R}$ captures features of source regions. $\circ$ denotes the Hadamard product between two tensors; and $\epsilon(\cdot)$, $\sigma(\cdot)$, and $\tanh(\cdot)$ are non-linear activation functions.

When applying CNRNN to predict $\widehat{\mathbf{C}}^{(t+1)}$, we need to change $\boldsymbol{W}$ to $\boldsymbol{W}'$ when conducting the graph convolution as $\mathbf{R}$ captures features of destination regions.

Given predicted factorization tensors $[\widehat{\mathbf{R}}^{(t+1)}, \ldots, \widehat{\mathbf{R}}^{(t+h)}]$ and $[\widehat{\mathbf{C}}^{(t+1)}, \ldots, \widehat{\mathbf{C}}^{(t+h)}]$, we apply the same recovery operation introduced in Section C.4.4 to obtain h full OD stochastic speed tensors for the future time intervals $T^{(t+1)}, \ldots, T^{(t+h)}$ as the recovery output: $\widehat{\mathbf{M}}^{(t+1)}, \ldots, \widehat{\mathbf{M}}^{(t+h)}$.

C.5.3 Loss Function

Similar to the construction covered in Section C.4.5, we present the loss function as follows.

$$\begin{aligned}\ell(\mathbf{G}, \boldsymbol{b}) = \sum_{i=1}^{h} [\lambda ||\mathbf{R}^{(t+j)}||_{\boldsymbol{W}}^2 + \lambda ||\mathbf{C}^{(t+j)}||_{\boldsymbol{W}}^2 + \\ ||\Omega^{(t+j)} \circ (\mathbf{M}^{(t+j)} - \widehat{\mathbf{M}}^{(t+j)})||_F^2]\end{aligned} \tag{C.14}$$

where $\mathbf{G}$ and $\boldsymbol{b}$ represent the training parameters in the framework (in particular, graph convolutional filters and bias vectors), $||\cdot||_{\boldsymbol{W}}^2$ is the Dirichlet norm under the proximity matrix $\boldsymbol{W}$, λ is the regularization parameter for the Dirichlet norm. We use the Dirichlet norm because it takes the adjacency matrix into account—nearby regions should share similar features in the dense tensors $\mathbf{R}$ and $\mathbf{C}$. Finally $\Omega^{(t+j)} \in \mathbb{R}^{N \times N' \times K}$ is an indication tensor, and $\widehat{\mathbf{M}}^{(t+j)}$ and $\mathbf{M}^{(t+j)}$, $j \in [1, h]$, are the predicted and ground truth tensors, respectively.

C.6 Experiments

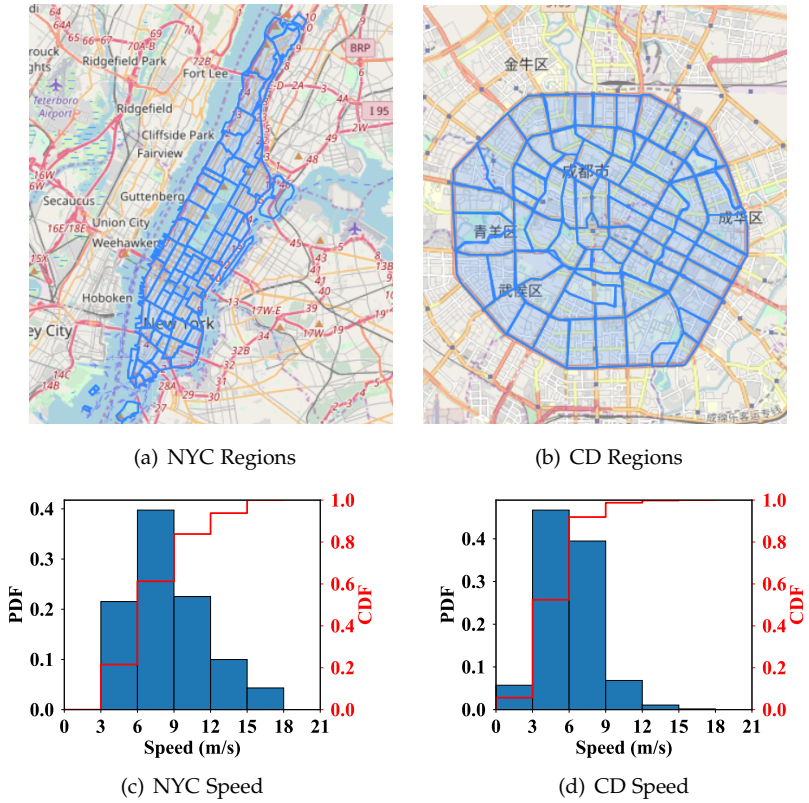
We describe the experimental setup and then present the experiments and the findings.

C.6.1 Experimental Setup

Datasets

We conduct experiments on two taxi trip datasets to study the effectiveness of the proposal.

We represent a stochastic speed (m/s) as a histogram with 7 buckets $[0, 3)$, $[3, 6)$, $[6, 9)$, $[9, 12)$, $[12, 15)$, $[15, 18)$, and $[18, \infty)$; and we consider 15-min intervals, thus obtaining 96 15-min intervals per day.



(a) NYC Regions (b) CD Regions

(c) NYC Speed (d) CD Speed

Fig. C.5: Region Representations of NYC and CD.

New York City Data Set (NYC): We use 14 million taxi trips collected from 2013-11-01 to 2013-12-31 from Manhattan, New York City. Each trip consists of a pickup time, a drop off time, a pickup location, a drop off location, and a total distance. Manhattan has 67 taxizones[4], each of which is used as a region. The regions are shown in Figure C.5(a). The OD stochastic speeds for NYC are represented as an $\mathbb{R}^{67\times67\times7}$ tensor.

Chengdu Data Set (CD): CD contains 1.4 billion GPS records from 14,864 taxis collected from 2014-08-03 to 2014-08-30 in Chengdu, China[5]. Each GPS record consists of a taxi ID, a latitude, a longitude, an indicator of whether the taxi is occupied, and a timestamp. We consider sequences of GPS records where taxis were occupied as trips. We use a total of 3,636,845 trips that occurred within the second ring road of Chengdu. Next, we partition Chengdu within the second ring road into 79 regions according to the main roads; see Figure C.5(b). The OD stochastic speeds for CD are represented as an $\mathbb{R}^{79\times79\times7}$ tensor.

Table C.1 shows the statistics of the two datasets. Figures C.5(d) and C.5(d) show the speed distributions for both datasets. We use 70% of the data for training, 10% for

[4] `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`

[5] `https://goo.gl/3VsEym`

	NYC	CD
# Trips	14,165,748	3,636,845
# Regions	67	79
Average Speed	8.8 m/s	6.0 m/s

Table C.1: Statistics of the Data Sets.

validation, and the remaining 20% for testing for both NYC and CD.

Forecast Settings

We consider settings where $s = 3$ or $s = 6$ while varying h among 1, 2, and 3. This means that we use stochastic OD matrices from 3 or 6 historical intervals to predict stochastic OD matrices during up to 3 future intervals, respectively. An example for $s = 6$ and $h = 3$ can be: given stochastic OD matrices in intervals [8:00, 8:15), [8:15, 8:30), [8:30, 8:45), [8:45, 9:00), [9:00, 9:15), and [9:15, 9:30), we predict stochastic OD matrices in intervals [9:30, 9:45), [9:45, 10:00), and [10:00, 10:15).

Baselines

To evaluate the effectiveness of the proposed base framework (BF) and the advanced framework (AF), we consider five baselines. (1) Naive Histograms (NH): for each OD pair, we use all travel speed records for the OD pair in the training data set to construct a histogram and use the histogram for predicting the future stochastic speeds. Next, we model the stochastic speeds for each OD pair as a time series of vectors, where each vector represents the stochastic speed of the OD pair in an interval. Based on this time series modeling, we consider three time series forecasting methods: (2) Support Vector Regression (SVR) [36], (3) Vector Auto-regression (VAR) [37], and (4) Gaussian Process Regression (GP) [38]. (5) Fully Connected (FC): this is a variant of BF where we only directly use a fully connected layer to obtain a single dense tensor (instead of performing factorization into two dense tensors) to replace the factorization step in BF.

Evaluation Metrics

To quantify the effectiveness of the proposed frameworks, we use three commonly used distance functions that work for distributions, i.e., Kullback-Leibler divergence (KL), Jensen-Shannon divergence (JS), and earth-mover's distance (EMD), to measure the accuracy of forecasts.

Specifically, the general dissimilarity metric is defined as follows.

$$\text{DisSim}^{(k)}_{\text{metric}} = \frac{\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N'}\Omega_{i,j}^{(t+k)}\text{metric}(\mathbf{M}_{i,j,:}^{(t+k)}, \widehat{\mathbf{M}}_{i,j,:}^{(t+k)})}{\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{j=1}^{N}\Omega_{i,j}^{(t+k)}}, \tag{C.15}$$

where $1 \leq k \leq h$ denotes the k-th step ahead forecasts, T is the size of testing set. Next, $\Omega^{(t+k)}$, $\mathbf{M}^{(t+k)}$, and $\widehat{\mathbf{M}}^{(t+k)}$ are the indication matrix, ground truth tensor, and forecast tensor, respectively. $\Omega^{(t+k)}_{(i,j)} = 1$ if observations exist from region i to region j at step $(t+k)$; otherwise, $\Omega^{(t+k)}_{(i,j)} = 0$. Moreover, metric$(\cdot)$ is a generic metric function that can be any of the metrics mentioned above and defined next. For simplicity, we use $\boldsymbol{m}$ and $\widehat{\boldsymbol{m}} \in \mathbb{R}^K$ to denote $\mathbf{M}^{(t+k)}_{i,j,:}$ and $\widehat{\mathbf{M}}^{(t+k)}_{i,j,:}$, respectively.

KL divergence,

$$\mathrm{KL}(\boldsymbol{m}, \widehat{\boldsymbol{m}}) = \sum_{k=1}^{K} \widehat{\boldsymbol{m}}_k \log(\frac{\widehat{\boldsymbol{m}}_k + \delta}{\boldsymbol{m}_k + \delta}), \tag{C.16}$$

where δ is a positive small value to prevent having a zero when using the *log* function. We use $\delta = 0.001$ in the experiment.

Jensen-Shannon divergence,

$$\mathrm{JS}(\boldsymbol{m}, \widehat{\boldsymbol{m}}) = \frac{\mathrm{KL}(\boldsymbol{m}, \widehat{\boldsymbol{m}}) + \mathrm{KL}(\widehat{\boldsymbol{m}}, \boldsymbol{m})}{2} \tag{C.17}$$

Earth mover's distance,

$$\mathrm{EMD}(\boldsymbol{m}, \widehat{\boldsymbol{m}}) = \frac{\sum_{i=1}^{K} \sum_{j=1}^{K} \boldsymbol{F}_{i,j} \boldsymbol{d}_{i,j}}{\sum_{i=1}^{K} \sum_{j=1}^{K} \boldsymbol{F}_{i,j}}, \tag{C.18}$$

where flow matrix $\boldsymbol{F}$ is the optimal flow that minimizes the overall cost from $\boldsymbol{m}$ to $\widehat{\boldsymbol{m}}$ [39].

All three functions capture the dissimilarity between an estimated and a ground-truth distribution. Thus, low values are preferred.

Model Construction

The proposed frameworks are trained by minimizing the two loss functions defined in Equation C.7 for BF and Equation C.14 for AF, using back-propagation. We use the Adam optimizer due to its good performance. The hyper-parameters were configured manually based on the loss on a separate validation set. Specifically, we set the initialization learning rate to 0.001, set with the decay rate to 0.8 at every 5 epochs, and set the dropout rate to 0.2.

Table C.2 shows the optimal configurations of the hyper-parameters for the three deep learning methods and numbers of weight parameters used in each model for both datasets. Baseline FC first encodes the input into a 2D latent space via an FC operation, denoted as FC_2. Then it calls a GRU with 3 units and 1 layer to capture the temporal dynamics, denoted as GRU^1_3. Finally, another FC is called to project the output from GRU to an OD stochastic tensor with the following dimensions: #Source Regions $\times$ #Destination Regions $\times$ #Buckets, e.g., $67 \times 67 \times 7 = 31,423$ dimensions for NYC, denoted as $FC_{31,423}$. For BF and AF, we apply two identical configurations for origin and destination factorization, which is why we have "2$\times$" on the first configuration, respectively. For BF, we first utilize FC_2 to encode the input for the first factorization. Then we adopt GRU^1_2 to learn the temporal dynamics. At the

Data	Model	Configuration	#Weights
NYC	FC	FC_3-GRU_3^1-$FC_{31,423}$	408,535
	BF	$2\times FC_2$-GRU_2^1-$FC_{2,345}$	391,182
	BF	$2\times GC_8^{32}$-$P4$-GC_4^{32}-$P2$-$GCR_2^{32\times 4}$	339,726
CD	FC	FC_3-GRU_3^1-FC_{43687}	567,967
	BF	$2\times FC_2$-GRU_2^1-$FC_{2,765}$	415,339
	AF	$2\times GC_8^{32}$-$P4$-GC_4^{32}-$P2$-$GCR_2^{32\times 4}$	367,502

Table C.2: Model Construction and Hyper-Parameter Selection.

end of GRU, we project the output into a corresponding factorization with the following dimensions: #Source Regions $\times$ r $\times$ #Buckets, where r is the rank of the factorized dense matrix which we set to 5, e.g., $67 \times 5 \times 7 = 2,345$ for NYC, denoted as $FC_{2,345}$. The configuration for AF is very different from the previous two models. First, we adopt two combinations of GCNN, GC_K^Q, where Q is the filter number and K is the filter size, and pooling operation, Pp, where p is the pooling size, e.g., GC_8^{32}-$P4$-GC_4^{32}-$P2$ for NYC. Then, the encoded features are fed into a CNRNN with n layers where each layer has four Cheby-Nets. Assuming that the number and size of the filters are Q_c and K_c, this operation can be written as $GCR_n^{Q_c\times K_c}$, e.g., $GCR_2^{32\times 4}$, implying 2 CNRNNs where the GCNN in each gate has 32 graph convolutional filters of size 4.

From the above configurations, although AF uses the most complex models, AF uses the fewest weight parameters (see the # weights column in Table C.2).

C.6.2 Experimental Results

Overall Results

We compare the accuracies of the different methods, using KL, JS, and EMD to evaluate the forecast accuracy; see in Tables C.3 and C.4. We also vary s, i.e., the number of historical stochastic speed matrices, and h, i.e., the h-intervals ahead forecasting, to study the effect of s and h. We have the following observations. (1) The deep learning based methods perform better than the other baselines in most cases. (2) The proposed basic framework BF performs better than other methods in most settings. This indicates that the proposed frameworks, which involve factorization and RNN based forecasting, are effective for OD matrix forecasting in settings with data sparseness. (3) The advanced framework AF is significantly better than other methods, including BF, in all settings. This suggests that by taking into account the spatial correlations among regions using two GCNNs, the learned features become more meaningful, which then improves forecasting accuracy. (4) The results on NYC are better than those on CD. This is because the regions in NYC are more homogeneous (i.e., within Manhattan) than the regions in CD that cover a much larger and more diverse region. This in turn makes the traffic situations in CD much more complex and more challenging to forecast. (5) When varying h, the accuracy of AF becomes worse, i.e., larger metric values. This suggests that forecast far into the future becomes more challenge. (6) When fixing h, we compare the two tables and observe that the performance of AF

Data	Metric	h	NH	SVR	VAR	GP	FC	BF	AF
NYC	KL	1	0.592	0.704	0.554	0.522	0.446	0.427	**0.311**
		2	0.592	0.713	0.562	0.535	0.438	0.417	**0.313**
		3	0.592	0.720	0.577	0.545	0.438	0.415	**0.314**
	JS	1	0.439	0.530	0.440	0.391	0.332	0.322	**0.299**
		2	0.439	0.537	0.452	0.400	0.327	0.317	**0.302**
		3	0.439	0.543	0.471	0.408	0.327	0.316	**0.305**
	EMD	1	0.293	0.452	0.305	0.266	0.250	0.246	**0.214**
		2	0.293	0.455	0.313	0.270	0.247	0.243	**0.216**
		3	0.293	0.458	0.317	0.274	0.247	0.243	**0.217**
CD	KL	1	0.697	0.818	0.699	0.674	0.694	0.582	**0.549**
		2	0.709	0.836	0.715	0.687	0.689	0.586	**0.555**
		3	0.700	0.822	0.780	0.684	0.807	0.792	**0.626**
	JS	1	0.580	0.672	0.814	0.597	0.517	0.438	**0.435**
		2	0.584	0.677	0.869	0.599	0.513	0.442	**0.444**
		3	0.592	0.692	0.875	0.619	0.590	0.571	**0.500**
	EMD	1	0.441	0.543	0.799	0.439	0.360	0.307	**0.289**
		2	0.443	0.539	0.871	0.434	0.361	0.313	**0.295**
		3	0.464	0.574	0.787	0.471	0.423	0.378	**0.311**

Table C.3: Forecast Accuracy with Varying h, $s = 3$.

is better at $s = 3$ than $s = 6$. This seems to indicate that the traffic variations are more dependent on short-term history (i.e., $s = 3$) than on long-term history (i.e., $s = 6$).

According to the above results, in the following, we only consider FC, BF, and AF, and we only consider the setting where $h = 1$ and $s = 6$, i.e., 1-step ahead forecasting with 6 historical observations.

Effect of Time of Day

In this experiment, we aim at investigating forecasting performance for different intervals during a day. To this end, we show the forecast accuracy across different time intervals. Figures C.6, C.7, and C.8 show the performance on both data sets when using EMD, KL, and JS. To visualize the results across time, we aggregate the results per each 3 hours. We use three curves to represent the accuracy of FC, BF, and AF. In addition, we use bars to represent the percentages of data we have per each 3 hours. CD does not contain any data from 00:00 to 06:00, which is why the figures for CD start at 6.

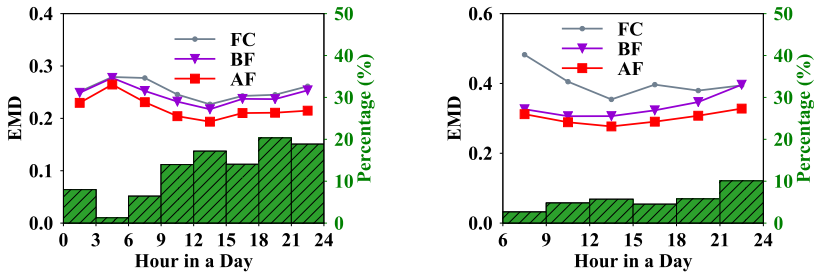
Figures C.6(a) and C.6(b) show the accuracy based on EMD. We observe that both AF and BF outperform FC in almost all the time intervals. This suggests the effectiveness of factorization in the proposed framework when contending with data sparseness. In addition, AF has the best performance and differs clearly from FC and BF. This suggests that by further capturing spatial and spatio-temporal correlations improves the forecast accuracy.

We observe that the EMD for all the three methods is the worst in NYC during [3:00, 6:00). This is because the amount of testing data during [3:00, 6:00) is quite small, only accounting for around 1% of the total testing data. On both data sets, the best EMD values appear during [12:00, 15:00). This indicates that the traffic conditions during this time period seems to have the least dynamics thus making the forecasting

Data	Metric	h	NH	SVR	VAR	GP	FC	BF	AF
NYC	KL	1	0.592	0.698	0.566	0.522	0.453	0.437	**0.343**
		2	0.592	0.706	0.576	0.535	0.445	0.421	**0.347**
		3	0.593	0.713	0.587	0.545	0.440	0.410	**0.348**
	JS	1	0.440	0.524	0.442	0.394	0.337	0.327	**0.305**
		2	0.440	0.531	0.455	0.404	0.333	0.318	**0.308**
		3	0.440	0.537	0.472	0.411	0.331	0.313	**0.310**
	EMD	1	0.293	0.447	0.299	0.265	0.248	0.238	**0.214**
		2	0.293	0.450	0.305	0.270	0.246	0.235	**0.216**
		3	0.294	0.452	0.314	0.274	0.245	0.233	**0.217**
CD	KL	1	0.686	0.804	0.741	0.671	0.753	0.668	**0.600**
		2	0.685	0.806	0.759	0.671	0.740	0.666	**0.605**
		3	0.686	0.807	0.739	0.674	0.735	0.667	**0.609**
	JS	1	0.578	0.674	0.808	0.610	0.554	0.517	**0.466**
		2	0.577	0.676	0.850	0.610	0.546	0.516	**0.475**
		3	0.578	0.678	0.903	0.612	0.543	0.517	**0.483**
	EMD	1	0.449	0.555	0.716	0.449	0.394	0.344	**0.304**
		2	0.448	0.557	0.767	0.448	0.389	0.344	**0.305**
		3	0.448	0.559	0.890	0.448	0.386	0.344	**0.306**

Table C.4: Forecast Accuracy with Varying h, $s = 6$.

less challenging. Similar trends can be observed when using KL and JS, as shown in Figures C.7 and C.8. Overall, the advanced framework AF achieves consistently the best forecasting performance on both datasets and on the three different evaluation metrics. More data enables more accurate forecasting.

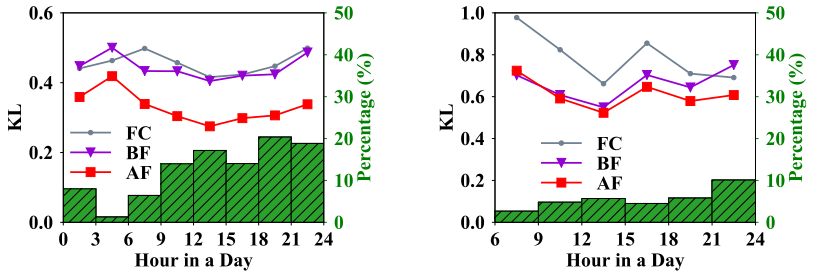


(a) NYC (b) CD

Fig. C.6: Effect of Time of a Day, EMD.

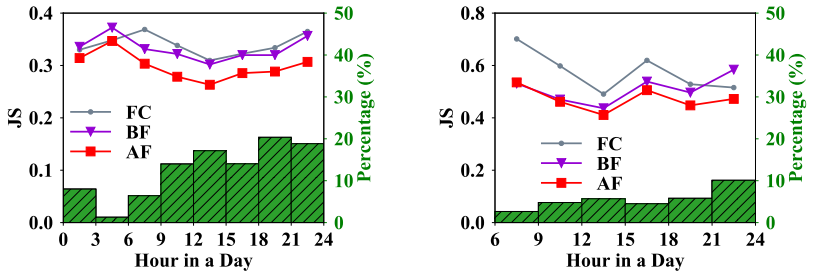
Effect of Distances

In this experiment, we aim at investigating the effect of the distances between source and destination regions. We thus report the forecast accuracy with different distances. Given a source and a destination region, we use the Euclidean distance between the centroids of the two regions as its corresponding distance. We group OD region pairs based on their distances into 6 groups as shown in Figures C.9, C.10, and C.11. We only consider OD region pairs that are below 3 km because less than 1% of the data



(a) NYC (b) CD

Fig. C.7: Effect of Time of a Day, KL.



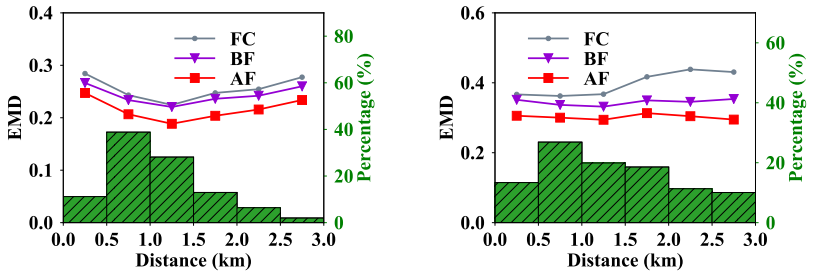
(a) NYC (b) CD

Fig. C.8: Effect of Time of a Day, JS.

is available for OD region pairs more than 3 km apart. Figures C.9, C.10, and C.11 report results on EMD, KL, and JS, respectively.

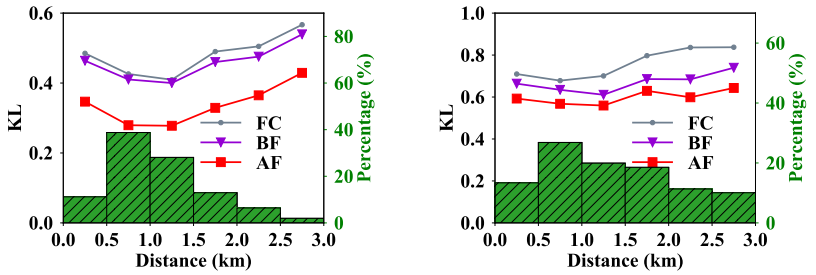
Figures C.9(a) and C.9(b) show the EMD values at varying distances on NYC and CD, respectively. We observe that (1) BF and AF outperform FC for all distance settings and on both datasets; (2) AF outperforms BF by a clear margin. This again offers evidence of effectiveness of the proposed advanced framework and suggests that the best performance is achieved by contending the sparseness and by capturing spatio-temporal correlations. Next, when considering distances from 0.5 to 1.5, i.e., the first three points of the curves, we observe a clear descending trend in NYC, but this trend is less obvious in CD.

We also observe that curves start to increase 1 km on NYC as shown in Figure C.9(a). The reason is amount of data in distance range [1.5, 3.0] decreases quickly, in turn introducing more fluctuations. We observe a subtle trend from distance range [1., 1.5) to distance range [1.5, 2.0) in Figure C.9(b) for BF and AF, however, FC is much worse in distance range [1.5, 3.0]. We have similar observations for NYC regarding the KL and JS evaluation metrics, which is shown in Figures C.10(a) and C.11(a). The trend is much clearer in CD for the evaluation metrics of KL and JS as is shown in Figures C.10(b) and C.11(b). Therefore, another explanation of the increasing tread is that as the distance increases, the route options also increase, which makes the speed more stochastic and harder to forecast. Overall, AF achieves the best performance on both datasets regarding to EMD, KL, and JS evaluation metrics.



(a) NYC (b) CD

Fig. C.9: Effect of Distances, EMD.



(a) NYC (b) CD

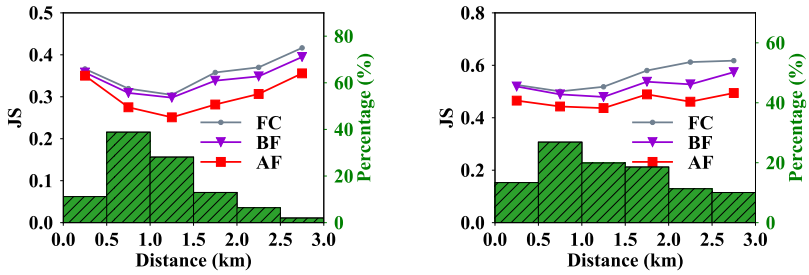
Fig. C.10: Effect of Distances, KL.

Effect of Proximity Matrices

We conduct a last set of experiments to investigate the effect of the parameters σ and α when constructing the proximity matrix $\boldsymbol{W}$ in the advanced framework. We only report results for CD due to the space limitation and because NYC yields similar results. Figures C.12(a) and C.12(b) show the accuracy when varying α and σ. The proposed AF is insensitive to σ and α. In other words, using proximity matrices is a robust way of capturing spatial correlations.

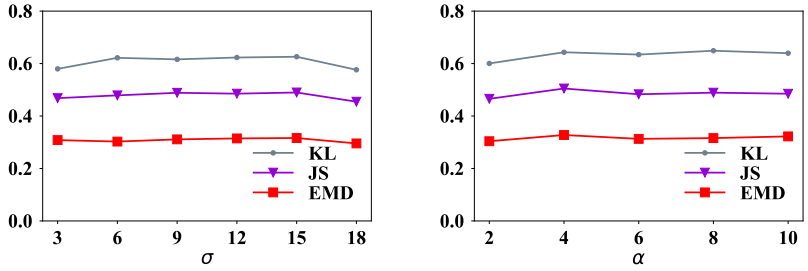
C.7 Conclusions and Outlook

An increasingly pertinent settings are asking for full OD matrices contain stochastic travel costs between any pair of regions in near future. However, instantiating such OD matrices calls for a large amount of vehicle trajectories which is almost an impossible task to fulfill in reality. We define and study the problem of stochastic origin-destination matrix forecasting in this setting. First, a data-driven, end-to-end deep learning framework is proposed to address the data sparseness problem by taking advantage of matrix factorization and recurrent neural networks. Further, a dual-stage graph convolution is integrated into factorization and recurrent neural networks to better capture the spatial correlations and thus lift performance. Empirical



(a) NYC (b) CD

Fig. C.11: Effect of Distances, JS.



(a) Impact of σ (b) Impact of α

Fig. C.12: Results w.r.t. Hop α.

studies on two real datasets from different countries, New York City and Chengdu City, demonstrate that the proposed framework outperforms other methods in all the experimental settings.

In future work, it is of interest to extend the framework to support continuous distribution models such as Gaussian mixture models.

References

[1] I. Ekowicaksono, F. Bukhari, and A. Aman, "Estimating origin-destination matrix of bogor city using gravity model," in *2016 IOP Conf. Ser.: Earth Environ. Sci.* 31 012021.

[2] J. Hu, B. Yang, C. S. Jensen, and Y. Ma, "Enabling time-dependent uncertain eco-weights for road networks," *GeoInformatica*, vol. 21, no. 1, pp. 57–88, 2017.

[3] H. Wang, Y.-H. Kuo, D. Kifer, and Z. Li, "A simple baseline for travel time estimation using large-scale trip data," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2016, pp. 61:1–61:4.

[4] I. Jindal, Tony, Qin, X. Chen, M. Nokleby, and J. Ye, "A Unified Neural Network Approach for Estimating Travel Time and Distance for a Taxi Trip," *ArXiv e-prints*, Oct. 2017.

[5] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proceedings of the 20th ACM International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 25–34.

[6] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *Proceedings of the 24th ACM International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 858–866.

[7] C. Guo, B. Yang, J. Hu, and C. S. Jensen, "Learning to route with sparse trajectory sets," in *Proceedings of the 34th IEEE International Conference of Data Engineering*, 2018, pp. 1073–1084.

[8] J. Rice and E. Van Zwet, "A simple and effective method for predicting travel times on freeways," in *Proceedings of the 33rd IEEE International Conference on Intelligent Transportation Systems*, 2001, pp. 227–232.

[9] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM International Conference on Knowledge Discovery and Data Mining*, 2011, pp. 316–324.

[10] B. Yang, C. Guo, and C. S. Jensen, "Travel cost inference from sparse, spatio-temporally correlated time series using markov models," *PVLDB*, vol. 6, no. 9, pp. 769–780, 2013.

[11] C. Wu, J. Ho, and D. Lee, "Travel-time prediction with support vector regression," *IEEE Trans. Intelligent Transportation Systems*, vol. 5, no. 4, pp. 276–281, 2004.

[12] B. Yang, C. Guo, C. S. Jensen, M. Kaul, and S. Shang, "Stochastic skyline route planning under time-varying uncertainty," in *Proceedings of the 30th IEEE International Conference of Data Engineering*, 2014, pp. 136–147.

[13] B. Yang, C. Guo, Y. Ma, and C. S. Jensen, "Toward personalized, context-aware routing," *The VLDB Journal–The International Journal on Very Large Data Bases*, vol. 24, no. 2, pp. 297–318, 2015.

[14] B. Yang, J. Dai, C. Guo, C. S. Jensen, and J. Hu, "PACE: a path-centric paradigm for stochastic path finding," *The VLDB Journal–The International Journal on Very Large Data Bases*, vol. 27, no. 2, pp. 153–178, 2018.

[15] J. Dai, B. Yang, C. Guo, C. S. Jensen, and J. Hu, "Path cost distribution estimation using trajectory data," *PVLDB*, vol. 10, no. 3, pp. 85–96, 2016.

[16] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? estimating travel time based on deep neural networks," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018, pp.2500–2507.

[17] H. Zhang, H. Wu, W. Sun, and B. Zheng, "Deeptravel: a neural network based travel time estimation model with auxiliary supervision," *arXiv preprint arXiv:1802.02147*, 2018.

[18] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu, "Multi-task representation learning for travel time estimation," in *Proceedings of the 24th ACM International Conference on Knowledge Discovery and Data Mining*, pp.1695–1704, 2018.

[19] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional neural networks applied to house numbers digit classification," in *Proceedings of the 21st IEEE International Conference on Pattern Recognition*, pp. 3288–3291, 2012.

[20] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3361–3368, 2011.

[21] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[22] D. Wang, W. Cao, J. Li, and J. Ye, "Deepsd: Supply-demand prediction for online car-hailing services using deep neural networks," in *Proceedings of the 33rd IEEE International Conference on Data Engineering*, pp. 243–254, 2017.

[23] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," *arXiv:1606.09375 [cs, stat]*, Jun. 2016.

[24] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral Networks and Locally Connected Networks on Graphs," *arXiv:1312.6203 [cs]*, Dec. 2013.

[25] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *arXiv:1609.02907 [cs, stat]*, Sep. 2016.

[26] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *Proceedings of the 27th World Wide Web Conferences*, pp. 499–508, 2018.

[27] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, "Deep learning: A generic approach for extreme condition traffic forecasting," in *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 777–785, 2017.

[28] F. Monti, M. M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proceedings of the 30th Advances in Neural Information Processing Systems*, pp. 3700–3710, 2017.

[29] N. Srebro, J. Rennie, and T. S. Jaakkola, "Maximum-margin matrix factorization," in *Proceedings of the 18th Advances in Neural Information Processing Systems*, pp. 1329–1336, 2005.

[30] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *Proceedings of the 20th International Conference on Machine Learning*, pp. 720–727, 2003.

[31] B. M. Marlin, "Modeling user rating profiles for collaborative filtering," in *Proceedings of the 17th Advances in Neural Information Processing Systems*, pp. 627–634, 2004.

[32] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

[33] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[34] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[35] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proceedings of the 6th International Conference on Learning Representations*, 2018, https://openreview.net/forum?id=SJiHXGWAZ.

[36] D. Basak, S. Pal, and D. C. Patranabis, "Support vector regression," *Neural Information Processing-Letters and Reviews*, vol. 11, no. 10, pp. 203–224, 2007.

[37] C. A. Sims, "Macroeconomics and reality," *Econometrica: Journal of the Econometric Society*, pp. 1–48, 1980.

[38] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced lectures on machine learning*, pp. 63–71, 2004.

[39] Y. Rubner, C. Tomasi, and L. J. Guibas, "A metric for distributions with applications to image databases," in *Proceedings of the 6th IEEE International Conference on Computer Vision*, pp. 59–66, 1998.

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-364-8

AALBORG UNIVERSITY PRESS