Analysis and Prediction of Spatiotemporal Impact of Traffic Incidents for Better Mobility and Safety in Transportation Systems

Final Report
METRANS Project 14-04

December 2015

Dr. Cyrus Shahabi
Dr. Ugur Demiryurek

Computer Science Department
Viterbi School of Engineering
University of Southern California
Los Angeles, CA 90089

DISCLAIMER

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation, University Transportation Centers Program, and California Department of Transportation in the interest of information exchange. The U.S. Government and California Department of Transportation assume no liability for the contents or use thereof. The contents do not necessarily reflect the official views or policies of the State of California or the Federal Highway Administration. This report does not constitute a standard, specification, or regulation.

ABSTRACT

The goal of this research is to develop a machine learning framework to predict the spatiotemporal impact of traffic accidents on the upstream traffic and surrounding region. The main objective of the framework is, given a road accident, to forecast when and how the travel-time delay will occur on transportation network. Towards this end, we have developed a Dynamic Topology-aware Temporal (DTT) machine learning algorithm that learns the behavior of traffic in both normal conditions and during accidents from the historical traffic sensor datasets. This research exploits four years of real-world Los Angeles traffic sensor data and California Highway Patrol (CHP) accidents logs collected from Regional Integration of Intelligent Transportation Systems (RIITS) under Archived Traffic Data Management System (ADMS) project.

TABLE OF CONTENTS

List of Figures **6**

List of Tables **7**

1 Introduction **10**

2 Related Work and Background **12**
2.1 Traffic Analysis . . . 12
2.2 Nonnegative Matrix Factorization . . . 12

3 Problem Definition **13**
3.1 Preliminaries . . . 13

4 Dynamic Topology-Aware Temporal Traffic Model **14**
4.1 Static Topology-Aware Traffic Model . . . 14
4.2 Dynamic Topology-Aware Traffic Model . . . 16
4.2.1 Temporal Effect of Latent Attributes . . . 16
4.2.2 Transition Matrix . . . 16
4.3 Edge Traffic Prediction with Missing Data . . . 16

5 Learning the DTT Model **17**
5.1 Global Learning Algorithm . . . 17
5.1.1 Update Rule of U_t . . . 17
5.1.2 Update Rule of B and A . . . 18
5.2 Incremental Learning Algorithm . . . 18
5.2.1 Framework of Incremental Algorithm . . . 19
5.2.2 Topology-Aware Incremental Update . . . 19

6 Real-Time Forecasting **22**

7 Experiment **23**
7.1 Dataset . . . 23
7.2 Experimental Setting . . . 23
7.2.1 Algorithms . . . 23
7.2.2 Configurations and Measures. . . . 24
7.3 Comparison for Missing Value Completion . . . 25
7.4 Comparison with Edge Traffic Prediction . . . 25
7.4.1 One-Step Ahead Prediction . . . 25
7.4.2 Multi-Steps Ahead Prediction . . . 26
7.5 Scalability of Different Methods . . . 27
7.6 Comparison for Real-Time Forecasting . . . 28
7.7 Varying Parameters of Our Methods . . . 29
7.7.1 Effect of Varying T . . . 29
7.7.2 Effect of Varying Span . . . 29
7.7.3 Effect of Varying k and λ . . . 31

8 Conclusion **32**

9 Appendix **34**
9.1 Derivatives of L with Respect to U_t in Equation 7. 34
9.2 Update Rule of A and B 36
9.3 Extra Experiment Results Based on Root Mean Square Error (RMSE) 37
9.4 Effect of Missing Data 39

List of Figures

1 Overall Framework . . . 11
2 An Example of Road Network . . . 14
3 An Example of Our Traffic Model. . . . 15
4 Illustration of Algorithms . . . 19
5 Two Challenges of Adjusting the Latent Attribute with Feedbacks. . . . 21
6 A Batch Recomputing Framework for Real-Time Forecasting. . . . 22
7 Sensor Distribution and Los Angles Road Network. . . . 23
8 Missing Value Completion Mean Average Percentage Error (MAPE) . . . 25
9 One-Step Ahead Prediction MAPE . . . 26
10 Six-Steps Ahead Prediction MAPE . . . 27
11 Converge Rate . . . 28
12 Online prediction MAPE . . . 29
13 Online Prediction time . . . 30
14 Effect of Varying T . . . 30
15 Effect of Varying Span . . . 31
16 Effect of Varying k and λ on Prediction Accuracy. . . . 31
17 Missing Value Completion RMSE on SMALL . . . 37
18 Edge Traffic Prediction RMSE One Step on SMALL . . . 37
19 Edge Traffic Prediction RMSE Six Steps Ahead on SMALL . . . 38
20 Online Prediction RMSE on SMALL . . . 38
21 Missing Rate During Training Stages for Support Vector Regression (SVR) and Auto-Regressive Integrated Moving Average (ARIMA) . . . 38

List of Tables

1 Notations and Explanations 13
2 Experiment Parameters 24
3 Running Time Comparisons. 28

DISCLOSURE

This project was funded in entirety under this contract to California Department of Transportation.

ACKNOWLEDGEMENTS

We acknowledge the United States Department of Transportation, California Department of Transportation (Caltrans) and National Center for Metropolitan Transportation Research (METRANS) for their interest and generous support to this research. We would also like to thank Regional Integration of Intelligent Transportation Systems (RIITS) to provide us Los Angeles County traffic sensor data for this research.

1 Introduction

The roads of almost all major cities in the world are equipped with traffic sensors – some of which video-camera-based (Closed-Circuit Television) and some based on customized sensors installed on the surface of the pavements (i.e., loop-detectors). These sensors continuously measure the average speed and the number of cars passed through a certain segment of the road. For example, there are approximately 15000 loop detectors installed on the highways and arterial streets of Los Angeles County (covering 3420 miles, cumulatively) collecting several main traffic parameters such as occupancy, volume, and speed at the rate of 1 reading per sensor per min. These data streams enable traffic prediction, which in turn improve route navigation, traffic regulation, urban planning, etc.

However, this dataset is far from perfect for several reasons. First, while many edges of the road network are equipped with sensors, there are still many edges with no traffic sensors, termed the *missing-sensor* problem. Second, even for the edges with sensors, every once in a while, no value is reported for some time span due to various device and network failures, termed the *missing-value* problem. Meanwhile, there is an opportunity to complete this missing information because the traffic flow is highly correlated spatially and temporally. For example, the readings before and after a set of missing values as well as the readings of other sensors on the same road segment during the time-span of missed values can be used to predict the missing values. The same intuition applies to predict traffic flow on edges with no sensors. The challenge is that certain sensors (or edges) can be better predictors as they may have the same features as the missing value (or edge). For example, a north-south edge going from a business district to a residential district is a better predictor for a missing-sensor edge with the same set of features. However, it is difficult to manually identify all the similar features as they are time-dependent and vary depending on various traffic conditions (e.g., morning vs. afternoon rush-hour).

Fortunately, the huge amount of historical datasets that have been collected in the past five years from these sensors in many major cities opens up a data-driven approach to both identify the edge and/or sensor feature similarities and at the same time exploit them for traffic prediction. Another good news is that for traffic prediction, if one waits long enough, the actual ground-truth will be eventually observed. For example, if the goal is to predict the traffic on a certain segment 10 minutes into the future, after 10 minutes, the actual traffic flow will be observed, which can be used in a feedback loop to improve the data-driven prediction approach. Therefore, in this paper, we propose a holistic data-driven approach to deal with both missing-sensor and missing-value challenges for real-time traffic prediction, based on the latent space modeling of road network (similar to latent space modeling for social networks), where two nodes that have similar attributes in a latent space, are more likely to be in the same cluster or with similar traffic patterns. Moreover, our approach takes advantage of the newly observed data as feedback to dynamically improve the latent space model.

Many studies have focused on the traffic prediction problem. Some focused on missing values [19] or missing sensors [11, 31], but not both. Some studies [17, 32] focus on utilizing temporal data which models each sensor (or edge) independently and makes predictions using time series approaches (e.g., ARIMA [17], SVR [20] and Gaussian Process (GP) [32]), and very few studies [13, 27] utilize spatiotemporal model with correlated time series based on Hidden Markov Model (HMM), but only for small number of time series and not always using the network space as the spatial dimension (e.g., using Euclidean space [9]). One study considers using the newly arrived data as feedback to reward one classifier vs. the other [25] but not for dynamically updating the model. Note that many existing studies [6, 11, 23, 26, 27, 31] on traffic prediction are based on Global Positioning System (GPS) dataset, which has a fundamental difference with the sensor dataset, where we have fine-grained and steady readings from road-equipped sensors. Another major distinction of our work is that we exploit the road network topology in every aspect of our approach. For example, we use graph Laplacian on the road network graph to fill in zero entries (missing data) and for our model update, we use topological order to perform sequential updates utilizing the new readings. We are not aware of any study that uses latent space modeling (considering both time and network topology) for real-time traffic prediction from incomplete (i.e., missing sensors and values) sensor datasets (see Section 2 for more detailed discussion).

In particular, we proposed a Dynamic Topology-Aware Temporal (DTT) traffic model, which provides a unified framework for both missing value and missing sensor completion. First, we built a static spatial traffic model on the latent space of road network (similar to latent space modeling for social networks [28, 30]), where two nodes that have similar attributes in a latent space, are more likely to be in the same cluster or with similar traffic patterns. Specifically,

we focused on the attributes of each vertex of the road network (e.g., highway vs. arterial, business vs. residential), where each vertex can have an overlapping representation of attributes. The attribute distribution of vertices and how the attributes interact with each other jointly determine the underlying traffic pattern. To tackle the sparsity of sensor data (due to missing values and sensors), we utilized the road topology by adding a graph Laplacian constraint, so that missing values of an edge can be completed by a set of similar edges with non-zero readings. Subsequently, we incorporated the temporal properties into our DTT model by considering time-dependent latent attributes, where at different timestamps one vertex could exhibit different properties. Finally, we further learned the underlying transition process of road network via a transition matrix, which depicts the likelihood of a vertex to be transited from one attribute to another. With these time-dependent latent attributes and the transition matrix, we were able to understand how traffic patterns are formed and evolve.

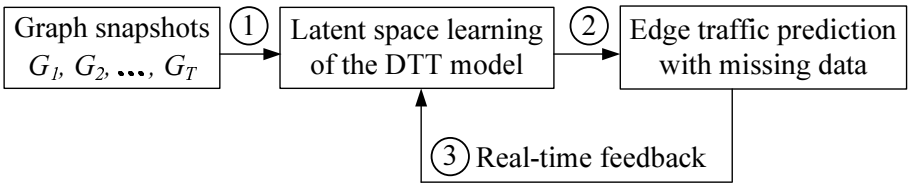


Figure 1: Overall Framework

Figure 1 shows an overview of the DTT framework. Given a series of road network snapshots, DTT processes them in three steps: (1) discover the latent attributes of vertices at each timestamp, which capture both the spatial and temporal properties; (2) understand the traffic patterns and build a predictive model of how these latent attributes change over time; and (3) exploit real time traffic information to adjust the existing latent space models. Note that our approaches are applicable for both normal and abnormal traffic conditions, and thus are able predict the spatiotemporal impact of traffic accidents on the upstream traffic and surrounding region.

To learn those time-dependent latent attributes, we proposed a global learning algorithm which belongs to the category of traditional multiplicative update algorithms [15]. The concept is to jointly infer the whole latent attributes via cyclical updates until they become stable. To improve the efficiency of global learning, we proposed an incremental algorithm, by exploiting the idea of online learning mechanism, where we sequentially and adaptively learned the latent attribute from the temporal graph changes. During the sequential learning process, each time when the algorithm makes a prediction with the latent attributes we already learned from the previous snapshot, it receives the feedback (i.e., the ground truth snapshot we already have in hand) and subsequently modifies the latent attributes for more accurate predictions. Unlike traditional online learning which only performs one single task (e.g., one single edge) update per round, here, we made predictions for the entire road network, thus we need to update the latent attributes for many correlated vertices. We further designed a topology-aware incremental update algorithm, which adaptively updates the latent positions of vertices with topology constraints. Through this process, our incremental algorithm not only well utilized the spatial and temporal properties of vertices, but also scaled to large road networks.

With our global and incremental learning algorithm, our DTT model can achieve a good trade-off between prediction accuracy and efficiency under the scenario of real-time forecasting, where the edge readings arrive in a streaming fashion. Specifically, we considered a setting with a predefined time window (a time window is composed of many time spans): at each time span (e.g., 5 minutes), we learned our traffic model with the proposed incremental approach on-the-fly, and made predictions for the next time span. Meanwhile, we batched the re-computation of our traffic model at the end of one large time window (e.g., one hour). Under this setting, our DTT model enjoys the following two nice properties: (1) real-time feedback information can be seamlessly incorporated into our framework to adjust for the existing latent spaces, thus allowing for more accurate predictions; (2) our algorithms train and make predictions on-the-fly with the given data, which avoids the concept-drift phenomenon [34] observed with non-linear time series prediction techniques (e.g., SVR and GP).

We conducted extensive experiments using a large volume of real-world traffic sensor dataset. By exploring both spatial and temporal properties, we demonstrated that the DTT framework achieves better accuracy than the state-of-the-art prediction methods. Moreover, we showed our algorithm scales to large road networks. For example, it only takes seconds to make a prediction for a network with 199,86 edges, whereas the largest network used by previous

studies contains less than 2000 edges for multiple correlated time series prediction [26]. Finally, we showed that our batch window setting works perfectly for streaming data, alternating the executions of our global and incremental algorithms, which strikes a compromise between prediction accuracy and efficiency.

The remainder of this paper is organized as follows. In Section 2, we discuss the related work. We define our problem in Section 3, and explain our DTT traffic model in Section 4. We present the global learning and increment learning algorithms in Section 5. We discuss how to adapt our algorithms for real-time traffic forecasting in Section 6. In Section 7, we report the experiment results and Section 8 concludes the paper.

2 Related Work and Background

In this section, we discuss related studies in traffic analysis, including whether each approach utilized temporal and/or topology (road network) information, and which category of methods were applied. We also briefly describe and review methods about non-negative matrix factorization, to provide more backgrounds about our modeling and algorithms.

2.1 Traffic Analysis

In order to provide a good estimation, most of the machine learning algorithms used in traffic analysis (such as ARIMA [17], HMM [6,13,26], GP [32], Support Vector Regression [16,20], Robust Regression [21] and Ensemble [2] etc.) require sufficient historical training data; while our approaches train and predict on-the-fly with few given training data. Although Zhou et. al. [32] also proposed a semi-lazy framework which selected very few training data to scale the methods of GP, they still require a huge number of historical data to guarantee that they were able to obtain enough similar training data as the new prediction data.

From the modeling aspect, one of the compelling features of the proposed approach is that it provides a unified framework that accomplishes the task of missing value completion and edge traffic prediction simultaneously. Instead, most of the exiting approaches, usually either focus on a specific task, or require to perform missing value completion before edge/trajectory prediction such as [9]. Another advantage of the proposed approach is that it exploits the topology information from a large road network in every aspect of our approach, seamlessly with the temporal information. To the contrary, most of the sensor-based traffic analysis methods [17, 18, 32], simply utilize historical readings for each edge or trajectory independently and do not utilize any topology information, especially the correlation among sensors that are close to one another in road network. That is, for each edge/trajectory, they train a model beforehand and then apply the learned model to either perform completion or prediction tasks in the future. For instance, Pan et. al. [17] learned time-series regression model ARIMA for each edge in advance, and then perform traffic prediction on top of the model. Zhou et. al. [32] proposed a new method to apply the GP model for sensor reading prediction. Few studies [13, 27] utilized spatiotemporal model with correlated time series based on HMM, but only for small number of time series.

Finally, from the view of scalability and other algorithmic issues, most of the existing topology-based approaches [13, 26] are not scalable to large-scale road networks; while our proposed algorithms are efficient for large networks. Furthermore, most approaches failed to utilize a nice characteristic of traffic data, that is, newly arrived data provide instant ground truth feedbacks except few notable work [25, 29]. In [25], they used feedbacks to reward one classifier versus the other, but they did not perform any dynamic update to the pre-trained model/classifier according to feedbacks; Zhang et. al. [29] focus on traffic clustering with instant feedback but their experiments are based on simulation. We design a topology-aware incremental algorithm, which adaptively updates our model with topology constraints, and is much more efficient than the global approach or the re-computing approach.

2.2 Nonnegative Matrix Factorization

Recently, many real data analytic problems such as community detection [28, 30], recommendation system [7], topic modeling [22], image clustering [5], and sentiment analysis [33], have been formulated as the problem of latent space learning. These studies assume that each vertex of the graph resides in a latent space, and vertices which are

close to each other are more likely to be in the same cluster (e.g., community or topic) or form a link. Different techniques have been proposed to learn the latent positions, where Non-Negative Matrix Factorization (NMF) is one of the most popular method because of its ease of interpretable and flexibility. The success of these studies provides an alternative way of understanding the underlying structure/pattern of graphs.

In this work, we explored the feasibility of applying dynamic NMF to traffic prediction domain. We applied dynamic NMF to not only learn the correlation among different edge weights (i.e., traffic costs) in road network with the observed structure of road network, but also learned the temporal traffic patterns across time. We also developed a global algorithm, based on the traditional multiplicative algorithm [15]. We further designed a topology-aware incremental algorithm, which adaptively updates the latent space representation for each node in the road network with topology constraints. The proposed algorithms differ from traditional online NMF algorithms such as [4], which perform single task online update independently.

Table 1: Notations and Explanations

Notations	Explanations
$\mathcal{N}, n$	road network, number of vertices of the road network
G	the adjacency matrix of a graph
U	latent space matrix
B	attribute interaction matrix
A	the transition matrix
k	the latent number
T	the number of snapshots
$span$	the time gap between two continuous snapshots
h	the prediction horizon
λ, γ	regularization parameters for graph Laplacian and transition process

3 Problem Definition

In this section, we formally define the problem of traffic predictions with missing data. Table 1 lists the notations we use throughout this paper.

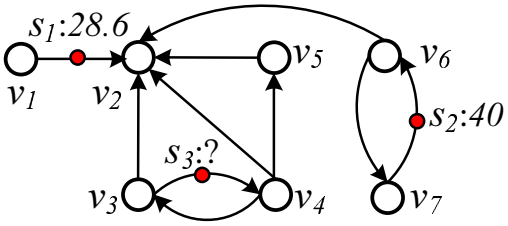
3.1 Preliminaries

Definition 1. ***Road network***. *A road network is denoted as a directed graph $\mathcal{N} = (V, E)$, where V is the set of vertices and $E \in V \times V$ is the set of edges, respectively. A vertex $v_i \in V$ models a road intersection or an end of road. An edge $e(v_i, v_j)$, that connects two vertices, represents a directed network segment. Each edge $e(v_i, v_j)$ is associated with a travel speed $c(v_i, v_j)$ (e.g., 40 miles/hour). The corresponding adjacency matrix representation of $\mathcal{N}$, is denoted as G, whose $(i, j)^{th}$ entry represents the edge weight between the i^{th} and j^{th} vertices.*

Definition 2. ***Traffic sensor***. *A traffic sensor s (i.e., a loop detector) is located at one segment of the road network $\mathcal{N}$, which provides a traffic speed reading (e.g., 40 miles/hour) per sampling rate (e.g., 30 seconds).*

We define $span$ as the temporal interval where we temporally aggregate sensor readings, which could be 5, 10 or 15 minutes. For each time span, these aggregated readings are mapped to the segments of network $\mathcal{N}$. For each edge, we aggregate all the sensor readings located at that edge, and assign a travel speed.

Therefore, after mapping sensor readings to road network, at each timestamp t, we generate a network snapshot G_t from the traffic sensors. As we already discussed, the dataset is incomplete with both missing values and missing sensors. For example, Figure 2(a) shows a simple road network with 7 vertices and 10 edges. Three sensors are located in edges (v_1, v_2), (v_3, v_4) and (v_7, v_6) respectively. Figure 2(b) shows the corresponding adjacent matrix after mapping the sensor readings to the road segments for one timestamp. Here sensor s_3 fails to provide reading, thus the edge weight of $c(v_3, v_4)$ is ? due to missing value. In addition, the edge weight of $c(v_3, v_2)$ is marked as $\times$ because of missing sensors.



(a) An Abstract Road Network $\mathcal{N}$

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	28.6	0	0	0	0	0
v_2	0	0	0	0	0	0	0
v_3	0	x	0	?	0	0	0
v_4	0	x	0	0	x	0	0
v_5	0	x	0	0	0	0	0
v_6	0	x	0	0	0	0	x
v_7	0	0	0	0	0	40	0

(b) Adjacency Matrix Representation G

Figure 2: An Example of Road Network

The high-level problem is to predict the future traffic conditions with a small number of road network snapshots, or a dynamic road network. We now first define the dynamic road network as follows:

Definition 3. ***Dynamic road network****. A dynamic road network, is a sequence of network snapshots* $(G_1, G_2, \cdots, G_T)$ *with edge weights denoting time-dependent travel cost.*

With a dynamic road network, we formally define the problem of edge traffic prediction with missing data as follows:

Problem 1. *Given a dynamic road network* $(G_1, G_2, \cdots, G_T)$ *with missing data at each timestamp, we aim to achieve the following two goals:*

- *complete the missing data (i.e., both missing value and sensor) of* G_i *, where* $1 \leq i \leq T$*;*
- *predict the future readings of* G_{T+h}*, where* h *is the prediction horizon. For example, when* $h = 1$*, we predict the traffic condition of* G_{T+1} *at the next timestamp.*

4 Dynamic Topology-Aware Temporal Traffic Model

In this section, we describe our DTT model in the context of traffic prediction. We first introduce the static topology-aware spatial traffic model (Section 4.1), and then present our DTT model that incorporates both temporal patterns and transition patterns (Section 4.2). With DTT model, we discuss how to solve the traffic prediction problem with missing data in Section 4.3.

4.1 Static Topology-Aware Traffic Model

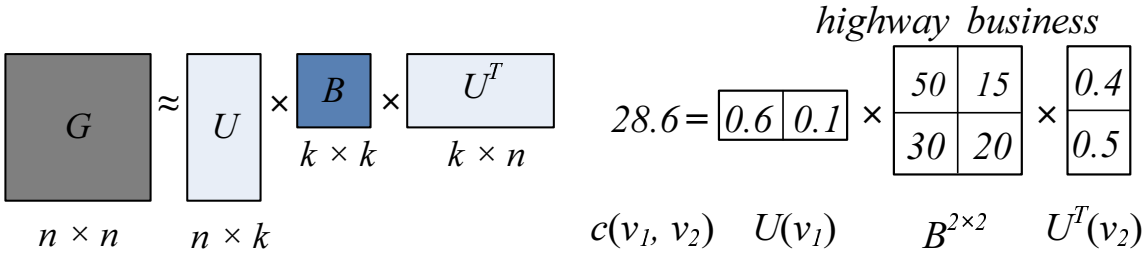
Our static spatial traffic model is built based on the latent space modeling of the observed road network, where each dimension of a latent space denotes a hidden spatial attribute of vertices in the road network. Basically, vertices of road network have different attributes (e.g., highway v.s. arterial street, business area v.s. residential area), and each vertex has an overlapping representation of attributes. Therefore, in our model, two connected nodes in a road network share similar spatial attributes. On the other hand, the attributes of vertices and how each attribute interacts with others jointly determine the underlining traffic patterns. Intuitively, if two highway vertices are connected, their corresponding interaction generates a higher travel speed than that from two vertices located at arterial streets.

Consequently, given a snapshot of road network G, we aim to learn two matrices U and B, where matrix $U \in R_+^{n \times k}$ denotes the latent attributes of vertices, and matrix $B \in R_+^{k \times k}$ denotes the attribute interaction patterns. The product of UBU^T represents the traffic speed between any two vertices, where we want to use to approximate G. Note that B

is an asymmetric matrix since the road network G is directed. Therefore, the static topology-aware traffic model could be determined by solving the following optimization problem:

$$\arg\min_{U\geq 0, B\geq 0} J = ||G - UBU^T||_F^2 \tag{1}$$

Figure 3 (a) illustrates the intuition of our static traffic model. The weights of edges in road network (i.e., traffic costs), can be approximated by the attributes of two end vertices and the interaction patterns. For example, in Figure 3, suppose we already know that each vertex is associated with two attributes (e.g., highway and business area), and the interaction pattern between two attributes encoded in matrix B, we could accurately estimate the travel speed between vertex v_1 and v_2, using their corresponding latent attributes and the matrix B.



(a) Basic model (b) Travel time of $c(v_1, v_2)$

Figure 3: An Example of Our Traffic Model.

Overcome the sparsity of G. In real applications, G is very sparse (i.e., zero entries dominate the items in G) for the following three main reasons: (1) not only the average degree of a road network is small [24], but also the maximum degree is very small. Therefore, even from a locality view, the edges of road network is far from fully connected; (2) the distribution of sensors is non-uniform, and only a small number of edges are equipped with sensors; and (3) there exists missing values (for those edges equipped with sensors) due to the failure and/or maintenance of sensors.

Learning from such sparse dataset of G yields additional challenges for traffic prediction. First, the learning process becomes extremely difficult because the estimated values would be dominated by the large number of zero entries. In addition, because a zero entry might be caused by different factors (i.e., unlinked vertices in the road network, edges without sensors, or missing readings from sensors), treating all zero entries the same introduces mystifying noises. As an example, in Figure 2, only 3 out of 10 edges have sensors, and thus we have at most 3 non-zero entries. Among all of those remaining zero entries, some of them are due to missing sensor, and the remaining of them are because of no interactions between two vertices in the road network.

Towards this end, we define our loss function only on edges with observed readings, that is, the set of edges with travel cost $c(v_i, v_j) > 0$. In addition, we also propose an in-filling method to reduce the gap between the input road network and the estimated road network. In particular, we conquer the sparsity by utilizing the graph topology, with which we estimate the traffic speed of an edge by the set of similar edges with non-zero weights. The similarity between two edges, is evaluated via similarities among their two-end vertices. Here we consider the multi-step similarity using graph random walk, or graph Laplacian dynamics, which has shown to be an effective smoothing approach for finding global structure similarity [14]. Specifically, we construct a graph Laplacian matrix L, defined as $L = D - W$, where W is a graph proximity matrix that is constructed from the network topology, and D is a diagonal matrix $D_{ii} = \sum_j (W_{ij})$. With these new constraints, the traffic model for a single snapshot of road network G is expressed as follows:

$$\arg\min_{U,B} J = ||Y \odot (G - UBU^T)||_F^2 + \lambda Tr(U^T L U) \tag{2}$$

where Y is an indication matrix for all the non-zero entries in G, i.e, $Y_{ij} = 1$ if and only if $G(i, j) > 0$; $\odot$ is the Hadamard product operator, i.e., $(X \odot Z)_{ij} = X_{ij} \times Z_{ij}$; and λ is the Laplacian regularization parameter.

4.2 Dynamic Topology-Aware Traffic Model

In our static topology-aware spatial traffic model, we do not leverage any temporal information of traffic patterns. However, time, as often perceived as the "fourth dimension", is crucial attribute in time series data. In the following, we combine the temporal information, including the time-dependent modeling of latent attributes and the transition of latent attributes, with our topology-aware spatial traffic model. In this model, each vertex is represented in a unified latent space, where each dimension of the latent space can either be a spatial attribute or a temporal attribute.

4.2.1 Temporal Effect of Latent Attributes

In practice, the travel speeds of most edges remain similar during a short time interval (e.g., 5 mins), and the whole network tends to stay steady during this period. Despite this steadiness of the overall traffic condition, vertices exhibit different attributes at different times of the day. For instance, the behavior of a vertex that is similar to that of a highway vertex during normal traffic condition, maybe similar to that of arterial street node during congestion hours. Because the behavior of each vertex can change over time, it is required to employ a time-dependent modeling for attributes of vertices and then to perform real-time travel speed estimation using those attributes.

Therefore, we add the time-dependent effect of attributes into our traffic model. Specifically, for each $t \leq T$, we aim to learn a corresponding time-dependent latent attribute representation U_t. Although the latent attribute matrix U_t is time-dependent, we assume that the attribute interaction matrix B is an inherent property, and thus we opt to fix B for all timestamps. By incorporating this temporal effect, we obtain our topology-aware temporal traffic model based on the following optimization problem:

$$\arg\min_{U_t,B} J = \sum_{t=1}^{T} ||Y_t \odot (G_t - U_t B U_t^T)||_F^2 + \sum_{t=1}^{T} \lambda Tr(U_t L U_t^T) \tag{3}$$

4.2.2 Transition Matrix

Due to the dynamics of traffic condition, we not only want to learn the time-dependent latent attributes, but also learn a transition model to capture the evolving behavior from one snapshot to the next. The transition should be able to capture both periodic evolving patterns (e.g., morning/afternoon rush hours) and non-recurring patterns caused by traffic incidents (e.g., accidents, road construction, or work zone closures). For example, during the interval of an accident, a vertex transition from normal state to congested at the beginning, then become normal again after the accident is cleared.

We thus assume a global process to capture the state transitions. Specifically, we use a matrix A that approximates the changes of U between time $t-1$ to time t, i.e., $U_t = U_{t-1}A$, where $U \in R_+^{n \times k}$, $A \in R_+^{k \times k}$. The transition matrix A represents how likely a vertex is to transition from attribute i to attribute j for that particular time interval.

Final objective formulation. Considering all the above, the final objective function for our DTT model is defined as follows:

$$\begin{aligned}\arg\min_{U_t,B,A} J = &\sum_{t=1}^{T} ||Y_t \odot (G_t - U_t B U_t^T)||_F^2 + \sum_{t=1}^{T} \lambda Tr(U_t L U_t^T) + \\ &\sum_{t=2}^{T} \gamma ||U_t - U_{t-1}A||_F^2\end{aligned} \tag{4}$$

where λ and γ are the regularization parameters.

4.3 Edge Traffic Prediction with Missing Data

Suppose by solving Equation 4, we obtain the learned matrices of U_t, B and A from our DTT model. Consequently, the task of both missing value and sensor completion can be accomplished by the following:

$$G_t = U_t B U_t^T, \qquad \text{when } 1 \le t \le T \tag{5}$$

Subsequently, the edge traffic for snapshot G_{T+h}, where h is the number of future time spans, can be predicted as follows:

$$G_{T+h} = (U_T A^h) B (U_T A^h)^T \tag{6}$$

5 Learning the DTT Model

In this section, we first propose a global multiplicative algorithm, and then discuss a fast incremental algorithm that scales to large road networks.

5.1 Global Learning Algorithm

We develop an iterative update algorithm to solve Equation 4, which belongs to the category of traditional multiplicative update algorithm [15]. In addition, our approach follows the popular (inexact) alternative least square update framework [12], where we optimize the objective function by updating one variable while fixing the other variables.

5.1.1 Update Rule of U_t

We first consider updating variable U_t while fixing all the other variables. The part of objective function in Equation 4 that is related to U_t can be rewritten as follows:

$$J = \sum_{t=1}^{T} Tr\Big(\big(Y_t \odot (G_t - U_t B U_t^T)\big) \big(Y_t \odot (G_t - U_t B U_t^T)\big)^T \Big)$$
$$+ \sum_{t=1}^{T} \lambda Tr(U_t (D - W) U_t^T) + \sum_{t=2}^{T} Tr\Big(\gamma (U_t - U_{t-1} A)(U_t - U_{t-1} A)^T \Big)$$

Because we have the non-negative constraint of U_t, following the standard constrained optimization theory, we introduce the Lagrangian multiplier $(\psi_t) \in R^{n \times k}$ and minimize the Lagrangian function L:

$$L = J + \sum_{t=1}^{T} Tr(\psi_t U_t^T) \tag{7}$$

Take the derivation of L with respect to U_t, we have the following expression. (The detail of this solution is described in Appendix 9.1)

$$\begin{aligned} \frac{\partial L}{\partial U_t} &= -2(Y_t \odot G_t)(U_t B^T + U_t B) + 2(Y_t \odot U_t B U_t^T)(U_t B^T + U_t B) \\ &+ 2\lambda (D - W) U_t + 2\gamma (U_t - U_{t-1} A) + 2\gamma (U_t A A^T - U_{t+1} A^T) + \psi_t \end{aligned} \tag{8}$$

By setting $\frac{\partial L}{\partial U_t} = 0$, and using the KKT conditions $(\psi_t)_{ij} (U_t)_{ij} = 0$, we obtain the following equations for $(U_t)_{ij}$:

$$\begin{aligned} &\big[-(Y_t \odot G_t)(U_t B^T + U_t B) + (Y_t \odot U_t B U_t^T)(U_t B^T + U_t B) \\ &+ \lambda L U_t + \gamma (U_\tau - U_{\tau-1} A) + \gamma (U_\tau A A^T - U_{\tau+1} A^T) \big]_{ij} (U_t)_{ij} = 0 \end{aligned} \tag{9}$$

Following the updating rules proposed and proved in [15], we derive the following update rule of U_t:

$$(U_t) \leftarrow (U_t) \odot \quad \Big(\frac{(Y_t \odot G)(U_t B^T + U_t B) + \lambda W U_t + \gamma (U_{t-1} A + U_{t+1} A^T)}{(Y_t \odot U_t B U_t^T)(U_t B^T + U_t B) + \lambda D U_t + \gamma (U_t + U_t A A^T)} \Big)^{\frac{1}{4}} \tag{10}$$

Algorithm 1 Global-learning($G_1, G_2, \cdots, G_T$)

Input: graph matrix $G_1, G_2, \cdots, G_T$.
Output: U_t ($1 \leq t \leq T$), A and B.

1: Initialize U_t, B and A
2: **while** Not Convergent **do**
3: **for** $t = 1$ to T **do**
4: update U_t according to equation 10
5: **end for**
6: update B according to Equation 11
7: update A according to Equation 12
8: **end while**

5.1.2 Update Rule of B and A

The updating rules for B and A could be derived as follows in a similar way (see Appendix 9.2 for detailed calculation):

$$B \leftarrow B \odot \Big(\frac{\sum_{t=1}^{T} U_t^T (Y_t \odot G_t) U_t}{\sum_{t=1}^{T} U_t^T (Y_t \odot (U_t B U_t^T)) U_t} \Big) \tag{11}$$

$$A \leftarrow A \odot \Big(\frac{\sum_{t=1}^{T} U_{t-1}^T U_t}{\sum_{t=1}^{T} U_{t-1}^T U_{t-1} A} \Big) \tag{12}$$

Algorithm 1 outlines the process of updating each matrix using aforementioned multiplicative rules to optimize Eq. 4. The general idea is to jointly infer and cyclically update all the latent attribute matrices U_t, B and A. In particular, we first jointly learn the latent attributes for each time t from all the graph snapshots (Lines 2–4). Based on the sequence of time-dependent latent attributes $< U_1, U_2, \cdots, U_T >$, we then learn the global attribute interaction pattern B and the transition matrix A (Lines 6–7).

From Algorithm 1, we now explain how our DTT model jointly learns the spatial and temporal properties. Specifically, when we update the latent attribute of one vertex $U_t i$, the spatial property is preserved by (1) considering the latent positions of its adjacent vertices ($Y_t \odot G_t$), and (2) incorporating the local graph Laplacian constraint (i.e., matrix W and D). Moreover, the temporal property of one vertex is then captured by leveraging its latent attribute in the previous and next timestamps (i.e., $U_{t-1}(i)$ and $U_{t+1}(i)$), as well as the transition matrix.

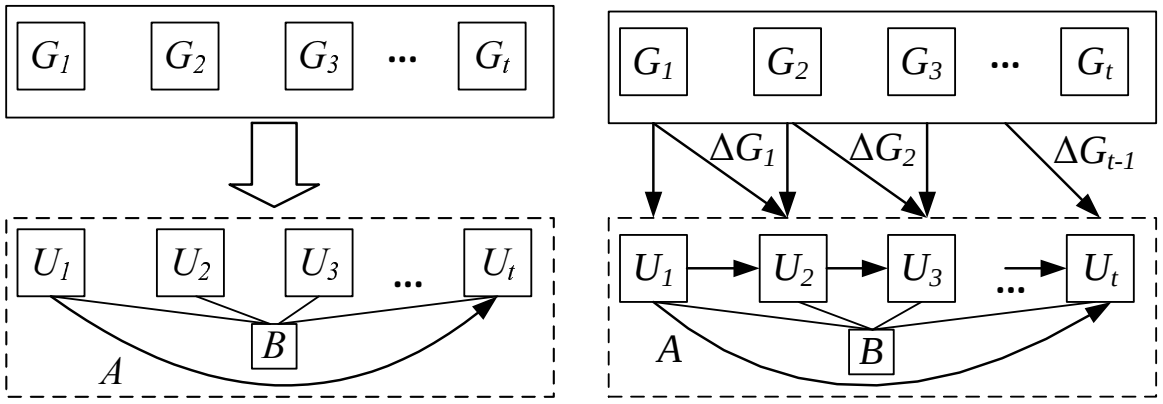
In the following, we present the detailed analysis of the time complexity and convergence property of our global multiplicative algorithm.

Complexity analysis. In each iteration, the computational complexity is dominated by matrix multiplication operations: the matrix multiplication between a $n \times n$ matrix and a $n \times k$ matrix, and another matrix multiplicative operator between a $n \times k$ matrix and a $k \times k$ matrix. Therefore, the worst case time complexity per iteration is dominated by $O(T(nk^2 + n^2 k))$. However, since in the traffic data, all the matrices are very sparse, and thus the complexity of matrix multiplication with two $n \times k$ sparse matrix, is much smaller than $O(n^2 k)$.

Convergence analysis. Algorithm 1 converges into a local minima. In addition, with Algorithm 1, the objective value is non-increasing in each iteration. This conclusion can be derived by following the proof shown in previous works [5] [15] [33].

5.2 Incremental Learning Algorithm

Although global multiplicative algorithm accurately captures the latent attribute, it is computationally expensive. As illustrated in Figure 4 (a), the latent attributes are jointly inferred from the entire set of graph snapshots and cyclically updated until they become stable. Unfortunately, this joint and cyclic inference is very time consuming. To improve the efficiency as well as preserve the topology and temporal properties, we propose an incremental algorithm.



(a) Global learning (b) Incremental learning

Figure 4: Illustration of Algorithms

As depicted in Figure 4 (b), our incremental algorithm sequentially and adaptively learns the latent attributes from the temporal graph changes.

5.2.1 Framework of Incremental Algorithm

The intuition of our incremental algorithm is based on the following observation: during a short time interval (e.g., 5 minutes), the overall traffic condition of the whole network tends to stay steady, and the travel costs of most edges change at a slow pace. For those edges with minor travel speed variations, their corresponding positions in the latent space do not change much either. Nevertheless, we still need to identify vertices with obvious variations in terms of their travel speeds, and adjust their corresponding latent attributes. For example, some edges could be in a transition state from non-rush hour to rush hour, and thus the travel speed reduces significantly. Therefore, instead of recomputing the latent attribute of each vertex from scratch at every time stamp, we perform "lazy" adjustment, utilizing the latent attributes we have already learned in the previous snapshot. As shown in Figure 4, to learn the latent attribute of U_t, the incremental algorithm utilizes the latent attributes we already learned in the previous snapshot (i.e., U_{t-1}) and the dynamism of traffic condition.

Algorithm 2 presents the pseudo-code of incremental learning algorithm. Initially, we learn the latent space of U_1 from our global multiplicative algorithm (Line 1). With the learned latent matrix U_{t-1}, at each time stamp t between 2 and T, we incrementally update the latent space of U_t from U_{t-1} according to the observed graph changes (Lines 2-4). After that, we learn the global transition matrix A (Line 5).

Algorithm 2 Incremental-Learning($G_1, G_2, \cdots, G_T$)

Input: graph matrix $G_1, G_2, \cdots, G_T$.
Output: U_t ($1 \leq t \leq T$), A and B.

1: $(U_1, B) \leftarrow$Global-learning(G_1)
2: **for** $t = 2$ to T **do**
3: $U_t \leftarrow$ Incremental-Update(U_{t-1}, G_t) (See Section 5.2.2)
4: **end for**
5: Iteratively learn transition matrix A using Equation 12 until A converges

5.2.2 Topology-Aware Incremental Update

Given U_{t-1} and G_t, we now explain how to calculate U_t incrementally from U_{t-1} , with which we could accurately approximate G_t. The main idea is similar to an online learning process. At each round, the algorithm predicts

an outcome for the required task (i.e., predict the speed of edges). Once the algorithm makes a prediction, it receives feedback indicating the correct outcome. Then, the online algorithm can modify its prediction mechanism, presumably improving the changes of making a more accurate prediction on subsequent timestamps. In our scenario, we first use the latent attribute matrix U_{t-1} to make a prediction of G_t as if we do not know the observation, subsequently we adjust the model of U_t according to the true observation of G_t we already have in hand.

However, in our problem, we are making predictions for the entire road network, not for a single edge. When we predict for one edge, we only need to adjust the latent attributes of two vertices, whereas in our scenario we need to update the latent attributes for many correlated vertices. Therefore, the effect of adjusting the latent attribute of one vertex could potentially affects its neighboring vertices, and influences the convergence speed of the incremental algorithm. Hence, the adjustment order of vertices also matters in our incremental update.

In a nutshell, our incremental update consists of the following two components: 1) identify candidate nodes based on feedbacks; 2) update their latent attributes and propagate the adjustment from one vertex to its neighbors. As outlined in Algorithm 3, given U_{t-1} and G_t, we first make an estimation of $\widehat{G_t}$ based on U_{t-1} (Line 1), subsequently we treat G_t as the feedback information, and select the set of vertices where we make inaccurate predictions, and insert them into a candidate set $cand$ (Lines 3-9). Consequently, for each vertex i of $cand$, we adjust its latent attribute so that we could make more accurate predictions (Line 15) and then examine how this adjustment would influence the candidate task set from the following two aspects: (1) if the latent attribute of i does not change much, we remove it from the set of $cand$ (Lines 17-19); (2) if the adjustment of i also affects its neighbor j, we add vertex j to $cand$ (Lines 20-25).

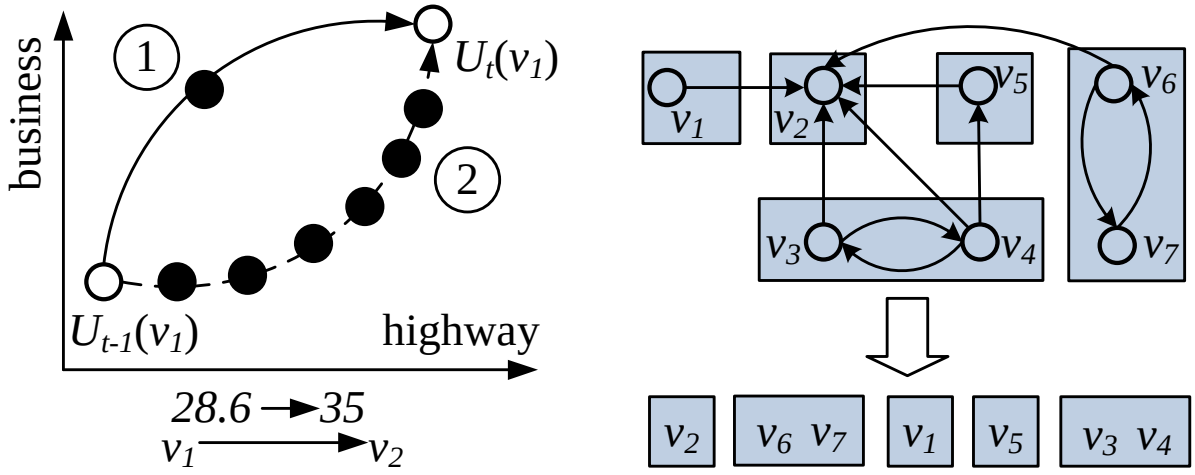
Algorithm 3 Incremental-Update(U_{t-1}, G_t)

Input: the latent matrix U_{t-1} at $t-1$, Observed graph reading G_t
Output: Updated latent space U_t.

1: $\widehat{G_t} \leftarrow U_{t-1} B U_{t-1}^T$
2: $cand \leftarrow \emptyset$
3: **for** each $i \in G$ **do**
4: **for** each $j \in out(i)$ **do**
5: **if** $|G_t(i,j) - \widehat{G_t(i,j)}| \geq \delta$ **then**
6: $cand \leftarrow cand \cup \{i, j\}$
7: **end if**
8: **end for**
9: **end for**
10: $U_t \leftarrow U_{t-1}$
11: **while** Not Convergent AND $cand \notin \emptyset$ **do**
12: **for** $i \in cand$ **do**
13: $oldu \leftarrow U_t(i)$
14: **for** each $j \in out(i)$ **do**
15: adjust $U_t(i)$ with Eq. 14
16: **end for**
17: **if** $||U_t(i) - oldu||_F^2 \leq \phi$ **then**
18: $cand \leftarrow cand \setminus \{i\}$
19: **end if**
20: **for** each $j \in out(i)$ **do**
21: $p \leftarrow U_t(i) B U_t(j)$
22: **if** $|p - G_t(i,j)| \geq \delta$ **then**
23: $cand \leftarrow cand \cup \{j\}$
24: **end if**
25: **end for**
26: **end for**
27: **end while**

The remaining questions in our Incremental-Update algorithm are how to adjust the latent position of one vertex

according to feedbacks, and how to decide the order of update. In the following, we address each of them separately.



(a) Adjustment method (b) Adjustment order

Figure 5: Two Challenges of Adjusting the Latent Attribute with Feedbacks.

Adjust the latent attribute of one vertex. To achieve high efficiency of adjusting the latent attribute, we propose to make the smallest changes of the latent space (as quick as possible) to predict the correct value. For example, as shown in Figure 5(a), suppose we already know the new latent position of v_1, then fewer step movement (Option 1) is preferable than gradual adjustment (Option 2). Note that in our problem, when we move the latent position of a vertex to a new position, the objective of this movement is to produce a correct prediction for each of its outgoing edges. Specifically, given $U_{t-1}(i)$, we want to find $U_t(i)$ which could accurately predict the weight of each edge $e(v_i, v_j)$ that is adjacent to vertex v_i. We thus formulate our problem as follows:

$$\begin{aligned} U_t(i), \xi^* = \arg \min_{U(i) \in R_+^k} & \frac{1}{2}||U(i) - U_{t-1}(i)||_F^2 + C\xi \\ \text{s.t.} \quad & |U(i)BU^T(j) - G_t(i,j)| \leq \delta + \xi \end{aligned} \tag{13}$$

Note that we have non-negativity constraint over the latent space of $U_t(i)$. To avoid over aggressive update of the latent space, we add a non-negative slack variable ξ into the optimization problem. Therefore, we adopt the approaches from [4]: When the predicted value $\widehat{y_t}$ (i.e., $U_t(i)BU_t^T(j)$) is less than the correct value y_t (i.e., $G_t(i,j)$), we use the traditional online passive-aggressive algorithm [8] because it still guarantees the non-negativity of $U(i)$; Otherwise, we update $U(i)$ by solving a quadratic optimization problem. The detailed solution is as follows:

$$U_t(i) = \max(U_{t-1}(i) + (k^* - \theta^*) \cdot BU_{t-1}(j)^T, 0) \tag{14}$$

k^* and θ^* are computed as follows:

$$\begin{cases} k^* = \alpha_t, \theta^* = 0 & \text{if } \widehat{y_t} < y_t \\ k^* = 0, \theta^* = C & \text{if } \widehat{y_t} > y_t \text{ and } f(C) \geq 0 \\ k^* = 0, \theta^* = f^{-1}(0) & \text{if } \widehat{y_t} > y_t \text{ and } f(C) < 0 \end{cases} \tag{15}$$

where

$$\alpha_t = \min\left(C, \frac{\max(|\widehat{y_t} - y_t| - \delta, 0)}{||BU_{t-1}(j)^T||^2}\right)$$

$$f_t(\theta) = \max\left(U_t(i) - \theta BU_t(j)^T, 0\right) \cdot BU_t(j)^T - G_t(i,j) - \delta$$

Updating order of $cand$**.** As we already discussed, the update order is very important because it influences the convergence speed of our incremental algorithm. Take the example of the road network shown in Figure 2, suppose our

initial $cand$ contains three vertices v_7, v_6 and v_2, where we have two edges $e(v_7, v_6)$ and $e(v_6, v_2)$. If we randomly choose the update sequence as $< v_7, v_6, v_2 >$, that is, we first adjust the latent attribute of v_7 so that $c(v_7, v_6)$ has a correct reading; subsequently we adjust the latent attribute of v_6 to correct our estimation of $c(v_6, v_2)$. Unfortunately,the adjustment of v_6 could influence the correction we have already made to v_7, thus leading to an inaccurate estimation of $c(v_7, v_6)$ again. A desirable order is to first update vertex v_6 before updating v_7.

Therefore, we propose to consider the topology of the road network when we update the latent position of each candidate vertex $v \in cand$. The general principle is that: given edge $e(v_i, v_j)$, the update of vertex v_i should be proceeded after the update of v_j, since the position of v_i is dependent on v_j. This motivates us to derive an topological order in the graph of G. Unfortunately, the road network G is not a Directed Acyclic Graph (DAG), and contains cycles. To address this issue, we first generate Strongly Connected Components (SCC) of the graph G, and contract each SCC as a super node. We then derive a topological order based on this condensed graph. For the vertex order in each SCC, we can either decide randomly, or use some heuristics to break the SCC into a DAG, thus generating an ordering of vertices inside each SCC. Figure 5(b) shows an example of ordering for the road network of Figure 2, where each rectangle represents a SCC. After generating a topological order based on the contracted graph and randomly decide the vertex order of each SCC, we obtain one final ordering $< v_2, v_6, v_7, v_1, v_5, v_4, v_3 >$. Therefore, each time when we update the latent attributes of $cand$, we follow this ordering of vertices.

Time complexity For each vertex i, the computational complexity of adjusting its latent attributes using Eq. 14 is $O(k)$, where k is number of attributes. Therefore, to compute latent attributes u, the time complexity per iteration is $O(kT(\Delta n + \Delta m))$, where Δn is number of candidate vertex in $cand$, and Δm is total number of edges incident to vertices in $cand$. In practice, $\Delta n \ll n$ and $\Delta m \ll m \ll n^2$, therefore, we conclude that the computational cost per iteration is significantly reduced using Algorithm 2 than using the global approach. The time complexity of computing the transition matrix A using Algorithm 2 is same with that of using the global approach.

6 Real-Time Forecasting

In this section, we discuss how to apply our learning algorithms to real-time traffic prediction, where the sensor reading is received in a streaming fashion. In practice, if we want to make a prediction for the current traffic, we cannot afford to apply our global learning algorithm to all the previous snapshots because it is computationally expensive. Moreover, it is not always true that more snapshots would yield a better prediction performance. The alternative method is to treat each snapshot independently: i.e., each time we only apply our learning algorithm for the most recent snapshot, and then use the learned latent attribute to predict the traffic condition. Obviously, this would yield poor prediction quality as it totally ignores the temporal transitions.

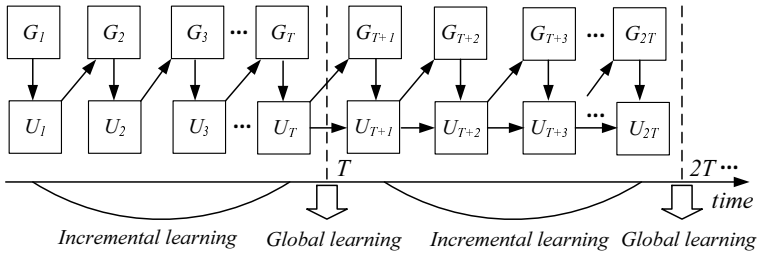


Figure 6: A Batch Recomputing Framework for Real-Time Forecasting.

To achieve a good trade-off between the above two methods, we propose to adapt a sliding window-based setting for the learning of our DTT model, where we only run our global learning algorithm at the end of one time window. As shown in Figure 6, we first apply our global learning at timestamps T (i.e., the end of the first time window), which learns the time-dependent latent attributes for the previous T timestamps. Subsequently, for each timestamp $T + i$ between [T, 2T], we apply our incremental algorithm to adjust the latent attribute and make further predictions: i.e., we use U_{T+i} to predict the traffic of G_{T+i+1}. Once we receive the true observation of G_{T+i+1}, we calculate U_{T+i+1} via the incremental update from Algorithm 3. The latent attributes will be re-computed at timestamp $2T$. The same

process continues in the next time window.

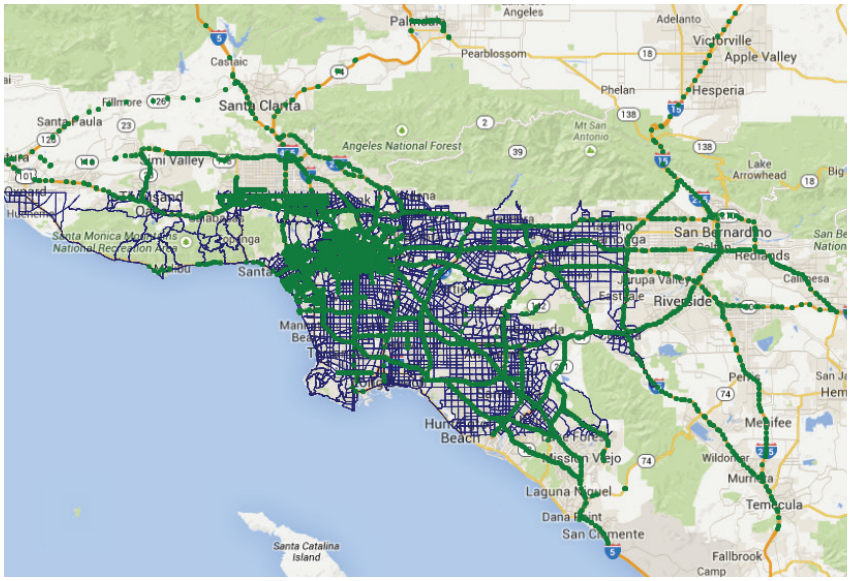


Figure 7: Sensor Distribution and Los Angles Road Network.

7 Experiment

7.1 Dataset

We used a large-scale and high resolution (both spatial and temporal) traffic sensor (loop detector) dataset collected from Los Angeles County highways and arterial streets. This dataset includes both inventory and real-time data for 15000 traffic sensors covering approximately 3420 miles. The sampling rate of the streaming data, which provides speed, volume (number of cars passing from sensor locations) and occupancy, is 1 reading/sensor/min. This sensor dataset have been continuously collected and archived since 2010.

We chose two months of sensor dataset (i.e., March and April in 2014) for our experiments, which include more than 60 million records of readings. As for the road network, we used Los Angeles road network which was obtained from HERE Map dataset [10]. We constructed two subgraphs of Los Angeles road network, termed as SMALL and LARGE. The SMALL network contains 5984 vertices and 12538 edges, and LARGE contains 8242 vertices and 19986 edges. As described in Section 3, the sensor data is mapped to the road network edges. Figure 7 shows sensors locations and road network segments, where the green lines depict the sensors, and blue lines represent the road network segments. After mapping the sensor data, we have two months of network snapshots for both SMALL and LARGE.

7.2 Experimental Setting

7.2.1 Algorithms

Our methods are termed as **DTT-All** (i.e., global learning algorithm) and **DTT-Inc** (i.e., incremental learning algorithm).

We first consider the task of missing-value and missing sensor completion. For the task of missing-value completion, we compare our algorithms with two methods: (1) KNN [9], which uses the average values of the nearby edges in Euclidean distance as the imputed value, (2) DTT-One, a local version of DTT-All, which applies Global-learning on each snapshot independently without considering the temporal factors, then uses the latent attributes of vertices to approximate the edge readings. We also implemented the Tensor method [1, 3] for missing value completion. However,

it cannot address the sparsity problem of our dataset and thus produce meaningless results (i.e., most of the completed values are close to 0). Although our framework is very general and supports missing sensor completion, we do not evaluate it through our experiments since we do not have ground truth values to verify.

For edge traffic prediction, we compare our algorithms with two representative time series prediction methods: a linear model (i.e., ARIMA [17]) and a non-linear model (i.e., SVR [20]). We train each model independently for each time series with historical data. In addition, because these methods will be affected negatively due to the missing values during the prediction stages (i.e, some of the input readings for ARIMA and SVR could be zero), for fair comparison we consider ARIMA-Sp and SVR-Sp, which use the completed readings from our global learning algorithm. We also compare with DTT-One, which always uses the most recent latent attribute as the prediction.

To evaluate the performance of online prediction, we consider the scenario of a batch-window setting described in Section 6. Considering a time window $[0, 2T]$, we sequentially predict the traffic condition for the timestamps during $[T+1, 2T]$, with the latent attributes of U_T and transition matrix A learned from the previous batch computing. Each time when we make a prediction, we receive the true observations as the feedback. We compare our Incremental algorithm (Inc), with three baseline algorithms: Old, Mini-batch and Full-batch. Specifically, in order to predict G_{T+i}, DTT-Inc utilizes the feedback of G_{T+i-1} to adjust the time-dependent latent attributes of U_{T+i-1}, whereas Old does not consider the feedback, and always uses latent attributes U_T and transition matrix A from the previous time window. On the other hand, Mini-batch ignores the previous snapshots, and only applies the global learning algorithm to the most recent snapshot G_{T+i-1}. Finally, Full-batch applies the global learning algorithm consistently to all historical snapshots (i.e., G_1 to G_{T+i-1}) and then makes a prediction.

Table 2: Experiment Parameters

Parameters	Value range
T	$2, 4, 6, 8, \mathbf{10}, 12$
$span$	$\mathbf{5}, 10, 15, 20, 25, 30$
k	$5, 10, 15, \mathbf{20}, 25, 30$
λ	$2^{-7}, 2^{-5}, 2^{-3}, 2^{-1}, 2^{1}, \mathbf{2^{3}}, 2^{5}$
γ	$2^{-7}, \mathbf{2^{-5}}, 2^{-3}, 2^{-1}, 2^{1}, 2^{3}, 2^{5}$

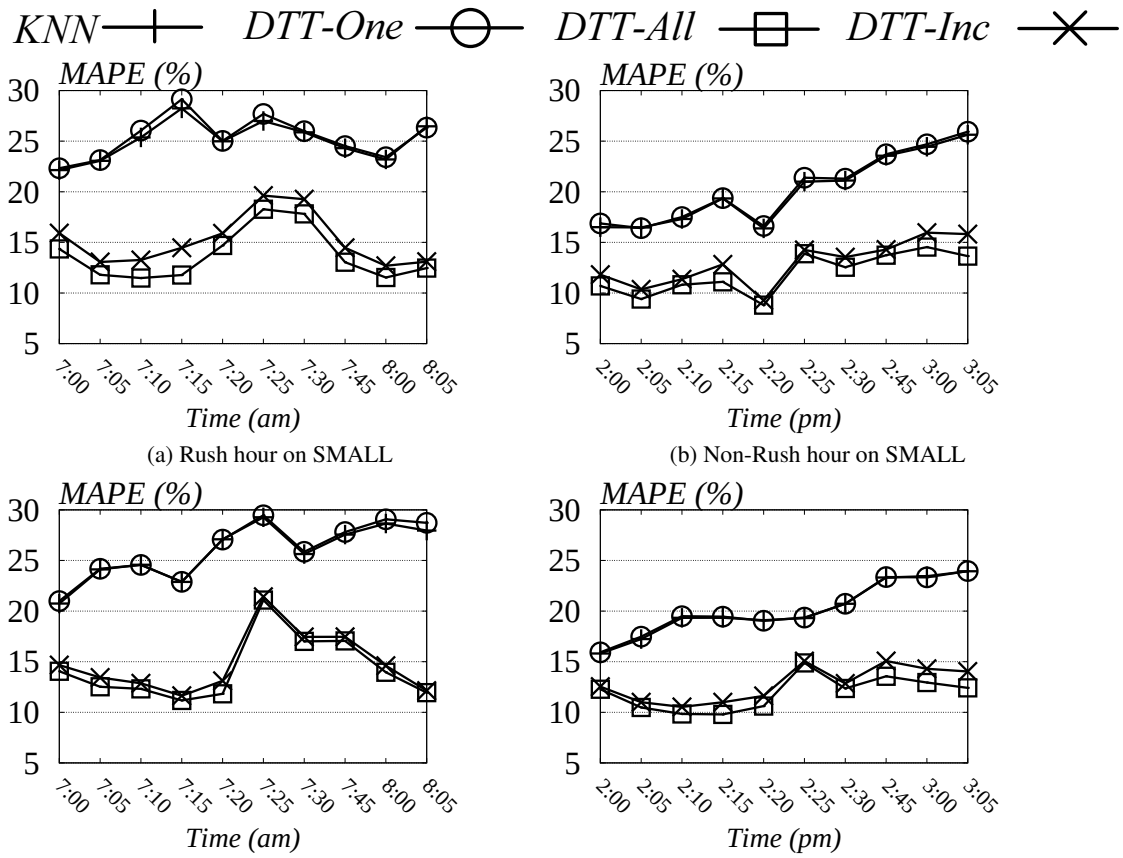
7.2.2 Configurations and Measures.

With our missing value completion and edge traffic prediction experiments, we selected two different time ranges that represent rush hour (i.e., 7am-8am) and non-rush hour (i.e., 2pm-3pm) respectively. For the task of missing value completion, during each timestamps of one time range (e.g., rush hour), we randomly selected 20% of values as unobserved and manipulated them as missing, with the objective of completing those missing values. For each traffic prediction task at one particular timestamp (e.g., 7:30 am), we also randomly selected 20% of the values as unknown and use them as ground-truth values.

We varied the parameters T and $span$: where T is the number of snapshots, and $span$ is time gap between two continuous snapshots. We also varied k, λ, and γ, which are parameters of our model. The default settings (shown with bold font) of the experiment parameter are listed in Table 2. Because of space limitations, the results of varying γ are not reported, which are similar to result of varying λ. We use Mean Absolute Percentage Error (MAPE) and Root Mean Square Error (RMSE) to measure the accuracy. In the following we only report the experiment results based on MAPE, the experiment results based on RMSE are reported in the Appendix. Specifically, MAPE is defined as follows:

$$MAPE = (\frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y_i}|}{y_i})$$

With ARIMA and SVR, we use the dataset of March to train a model for each edge, and use 5-fold cross-validation to choose the best parameters. All the tasks of missing value completion and edge traffic prediction tasks are conducted on April data. We conducted our experimentswith C++ on a Linux PC with i5-2400 CPU @ 3.10G HZ and 24GB memory.



(a) Rush hour on SMALL

(b) Non-Rush hour on SMALL

(c) Rush hour on LARGE

(d) Non-Rush hour on LARGE

Figure 8: Missing Value Completion Mean Average Percentage Error (MAPE)

7.3 Comparison for Missing Value Completion

In this set of experiments, we evaluate the completion accuracy of different methods. The experiment results on SMALL are shown in Figure 8 (a) and (b). We observe that both DTT-All and DTT-Inc achieve much lower errors than that of other methods. This is because DTT-All and DTT-Inc capture both spatial and temporal relationships, while DTT-One and KNN only use spatial property. DTT-All performs better than DTT-Inc by jointly infer all the latent attributes. On the other hand, we note that DTT-One and KNN have similar performances, which shows that the effect of utilizing either Euclidean or Topology proximity are similar for the task of missing value completion. This also indicates that utilizing both spatial and temporal property yields a large gain than only utilizing spatial property.

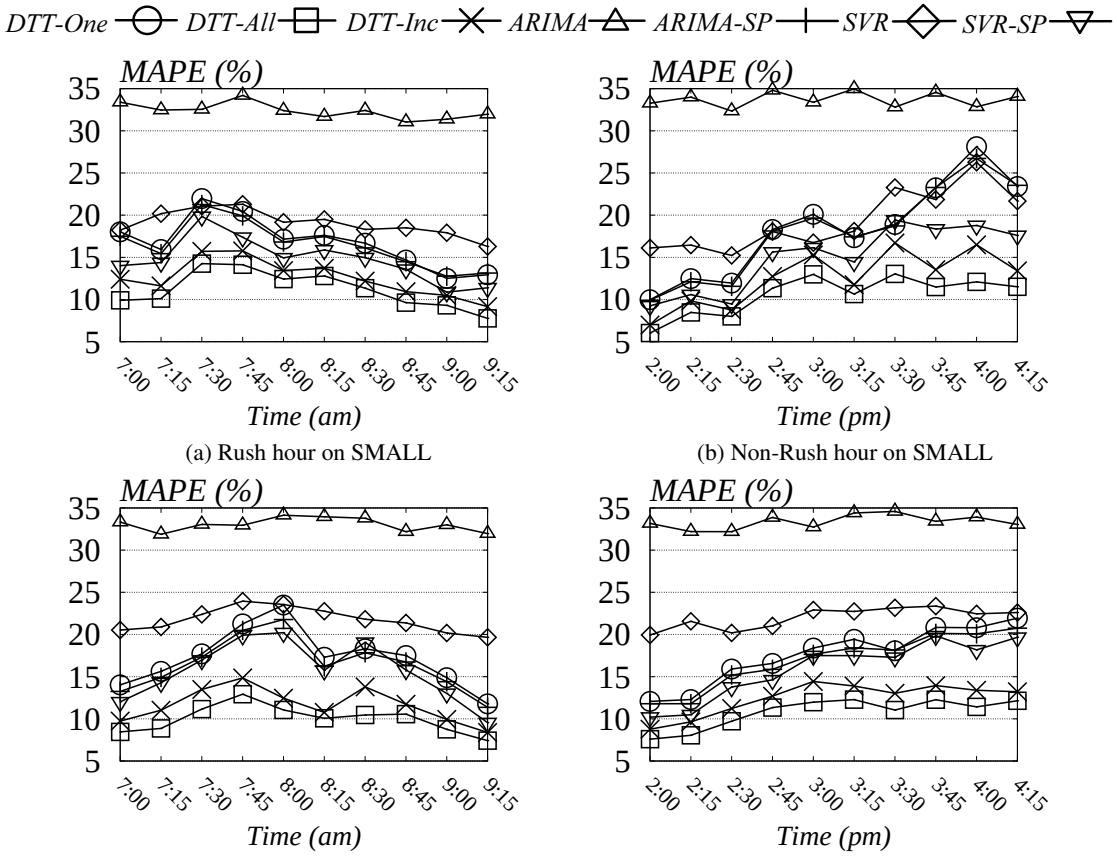
As shown in Figure 8(b), the completion performance on the non-rush hour is better as compared to on the rush hour time interval. This is because during rush hour range, the traffic condition is more dynamic, and the underlying pattern and transition changes frequently. All of these factors render worse performance during rush hour. Figure 8 (c) and (d) depict the experiment results on LARGE, which are similar on that of SMALL.

7.4 Comparison with Edge Traffic Prediction

In the following, we present the results of edge traffic prediction experiments.

7.4.1 One-Step Ahead Prediction

The experimental results of SMALL are shown in Figure 18 (a) and (b). Among all the methods, DTT-All and DTT-Inc achieve the best results, and DTT-All performs slightly better than DTT-Inc. This demonstrates that the



(a) Rush hour on SMALL

(b) Non-Rush hour on SMALL

(c) Rush hour on LARGE

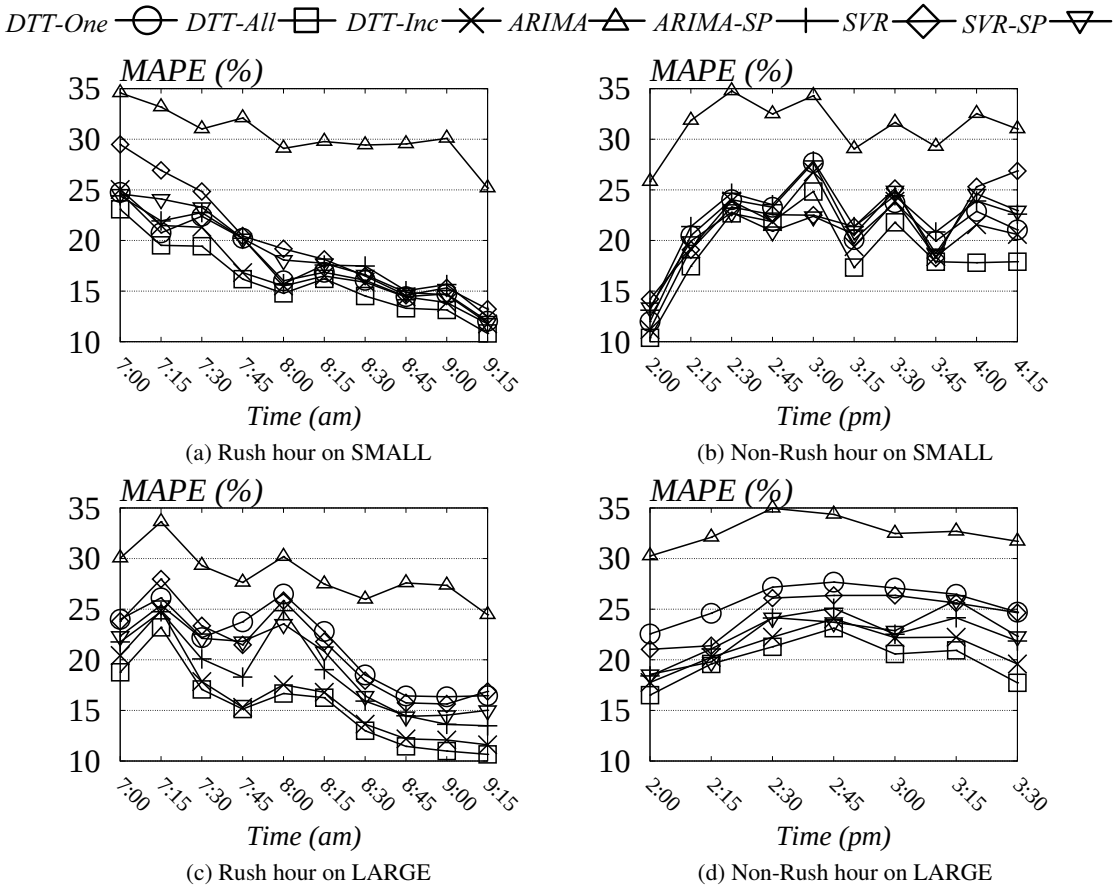
(d) Non-Rush hour on LARGE

Figure 9: One-Step Ahead Prediction MAPE

effectiveness of time-dependent latent attributes and the transition matrix. We observe that without the imputation of missing values, time series prediction techniques (i.e., ARIMA and SVR) perform much worse than DTT-ALL and DTT-Inc. Meanwhile, DTT-One, which only considers the spatial factor, cannot achieve good prediction results as compared to DTT-All and DTT-Inc. This indicates and enforces our claim that utilizing either temporal or spatial factors is not enough for making accurate predictions. We also note that even with completed readings, the accuracy of SVR-Sp and ARIMA-Sp is still worse than that of DTT-All and DTT-Inc. One reason is that simply combination of the spatial and temporal properties does not necessarily yield a better performance. Another reason is that both SVR-Sp and ARIMA-Sp also suffer from missing data during the training stage, which renders less accurate prediction. In Appendix 9.4, we also show how the ratio of missing data would influence the prediction performance. Finally, we observe that SVR is more robust than ARIMA when encountering missing value on prediction stages: i.e., ARIMA-Sp performs significantly better than ARIMA, while the improvement of SVR-Sp over SVR is not much. This is because ARIMA is a linear model which mainly uses the weighted average of the previous readings for prediction, while SVR is a non-linear model that utilizes a kernel function. Figure 9 (c) and (d) show the experiment results on LARGE, the overall results are similar to those of SMALL.

7.4.2 Multi-Steps Ahead Prediction

We now present the experiment results on multi-step ahead prediction, with which we predict the traffic conditions for next 30 minutes (i.e., $h = 6$). The prediction accuracy comparison among different methods on SMALL are shown in Figure 10 (a) and (b). Although DTT-All and DTT-Inc still outperforms other methods, the margin between our methods and the baselines is smaller. The reason is that, when we make multiple-step ahead prediction, we use



(a) Rush hour on SMALL

(b) Non-Rush hour on SMALL

(c) Rush hour on LARGE

(d) Non-Rush hour on LARGE

Figure 10: Six-Steps Ahead Prediction MAPE

the predicted values from the past for future prediction. This leads to the problem of error accumulation, i.e., errors incurred in the past are propagated into future predictions. We observe the similar trends on LARGE, from the results reported in Figure 10 (c) and (d).

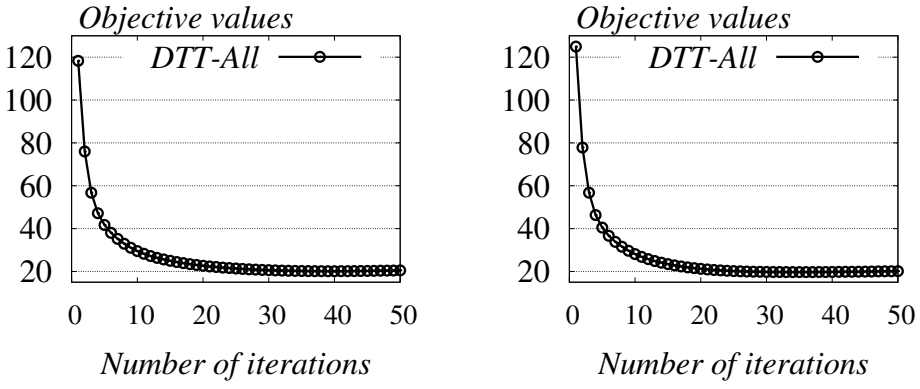
7.5 Scalability of Different Methods

Table 3 shows the running time of different methods. Although ARIMA and SVR is fast for each prediction, they have much higher training cost. Note that our methods do not require extra training data, i.e., our methods train and predict at the same time. Among them, DTT-Inc is the most efficient approach: it only takes less than 500 milliseconds to learn the time-dependent latent attributes and make predictions for all the edges of the road network. This is because our incremental learning algorithm conditionally adjusts the latent attributes of certain vertices, and utilizes the topological order that enables fast convergence. Even for the LARGE dataset, DTT-Inc only takes less than five seconds, which is acceptable considering the span between two snapshots is at least five minutes in practice. This demonstrates that DTT-Inc scales well to large road networks. Regarding DTT-All and DTT-One, they both require much longer running time than that of DTT-Inc. In addition, DTT-All is faster than DTT-One. This is because DTT-One independently runs the global learning algorithm for each snapshot T times, while DTT-All only applies global learning for the whole snapshots once.

Convergence analysis. Figure 11 (a) and (b) report the convergence rate of iterative algorithm DTT-All on both SMALL and LARGE. As shown in Figure 11, DTT-All converges very fast: when the number of iterations is around 20, our algorithm tends to converge in terms of our objective value in Equation 4.

Table 3: Running Time Comparisons.

data	SMALL		LARGE	
	train (s)	pred.(ms)	train (s)	pred. (ms)
DTT-One	-	1353	-	29439
DTT-All	-	869	-	14247
DTT-Inc	-	407	-	4145
ARIMA	484	0.00015	987	0.00024
SVR	47420	0.00042	86093.99	0.00051



(a) SMALL (b) LARGE

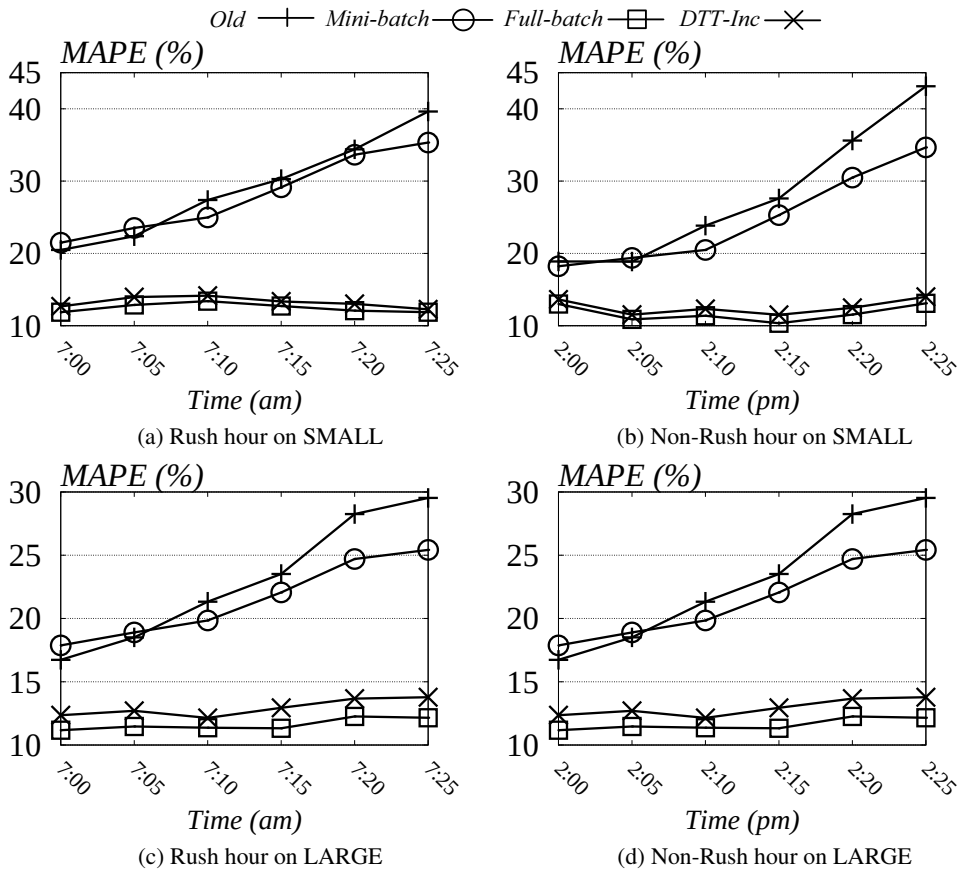
Figure 11: Converge Rate

7.6 Comparison for Real-Time Forecasting

In this set of experiments, we evaluate our online setting algorithms. As shown in Figure 12 (a) and (b), DTT-Inc achieves comparable accuracy with Full-batch. This is because DTT-Inc effectively leverages the real-time feedback information to adjust the latent attributes. We observe that DTT-Inc performs significantly better than Old and Mini-batch, which ignores either the feedback information (i.e., Old) or the previous snapshots (i.g., Mini-batch). One interesting observation is that Old performs better than Mini-batch for the initial timestamps, whereas Old surpasses Mini-batch at the later timestamps. This indicates that the latent attributes learned in the previous time-window are more reliable for predicting the near-future traffic conditions, but may not be good for multi-step ahead prediction because of the error accumulation problem. Similar results have also been observed in Figure 10 for multi-step ahead prediction. Figure 12 (c) and (d) show similar effects on LARGE.

Figure 13 (a) and (b) show the running time comparisons of different methods. One important conclusion for this experiment is that DTT-Inc is the most efficient approach, which is on average two times faster than Mini-batch and one order of magnitude faster than Full-batch. This is because DTT-Inc performs a conditional latent attribute update for vertices within a small portion of road network, whereas Mini-batch and Full-batch both recompute the latent attributes from at least one entire road network snapshot. Because Full-batch utilizes all the up-to-date snapshots and Mini-batch only considers the most recent single snapshot, Mini-batch is faster than Full-batch. Figure 13 (c) and (d) show the running time on LARGE. We observe that DTT-Inc only takes less than 1 seconds to incorporate the real-time feedback information, while Mini-batch and Full-batch take much longer.

Therefore, we conclude that DTT-Inc achieves a good trade-off between prediction accuracy and efficiency, which is applicable for real-time traffic prediction applications.



(a) Rush hour on SMALL

(b) Non-Rush hour on SMALL

(c) Rush hour on LARGE

(d) Non-Rush hour on LARGE

Figure 12: Online prediction MAPE

7.7 Varying Parameters of Our Methods

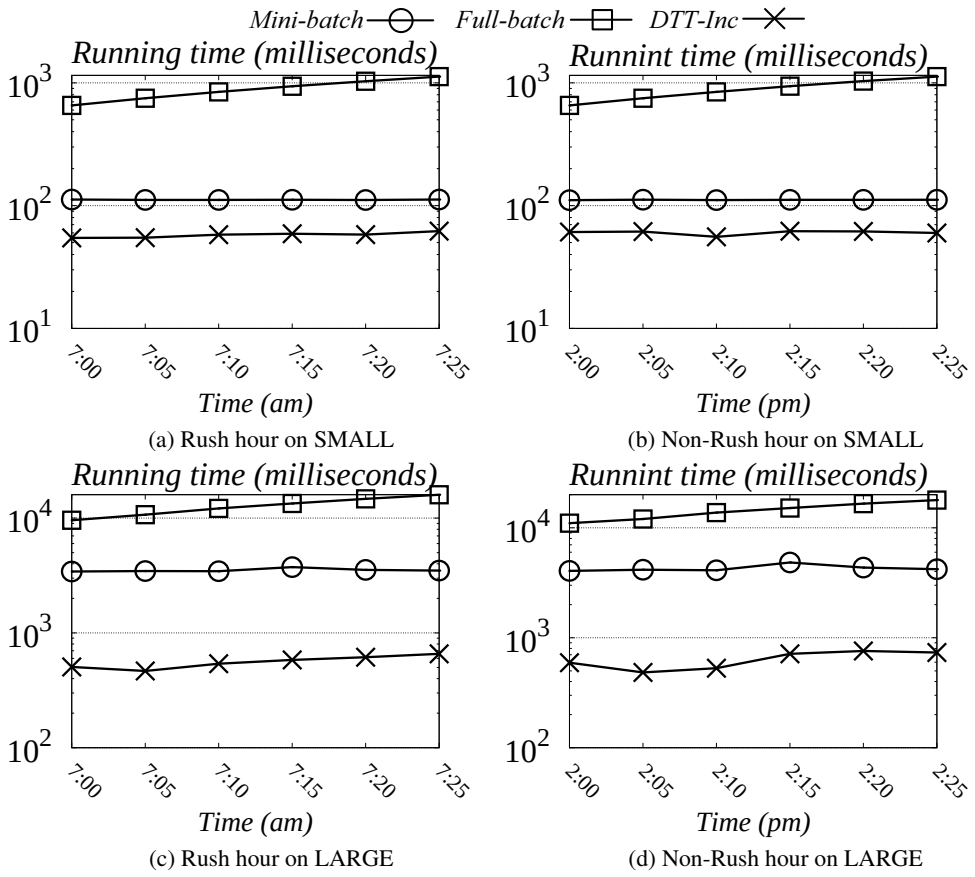
In the following, we test the performance of our methods by varying the parameters of our model. Since the effect of parameters does not have much correlation with the size of road network, we only show the experimental results on SMALL.

7.7.1 Effect of Varying T

Figure 14 (a) and Figure 14 (b) shows the prediction performance and the running time of varying T, respectively. We observe that with more number of snapshots, the prediction error decreases. In particular, when we increase T from 2 to 6, the results improve significantly. However, the performance tends to stay stable at $T \geq 6$. This indicates that smaller number of snapshots (i.e., two or less) are not enough to capture the traffic patterns and the evolving changes. On the other hand, more snapshots (i.e., more historical data) do not necessarily yield much gain, considering the running time increases when we have more number of snapshots. Therefore, to achieve a good trade-off between running time and prediction accuracy, we suggest to use at least 6 snapshots, but no more than 12 snapshots.

7.7.2 Effect of Varying Span

The results of varying $span$ are shown in Figure 15. It is clear that as the time gap between two snapshots increases, the performance declines. This is because when $span$ increases, the evolving process of underlying traffic may not stay smooth, the transition process learned in the previous snapshot are not applicable for the next prediction.



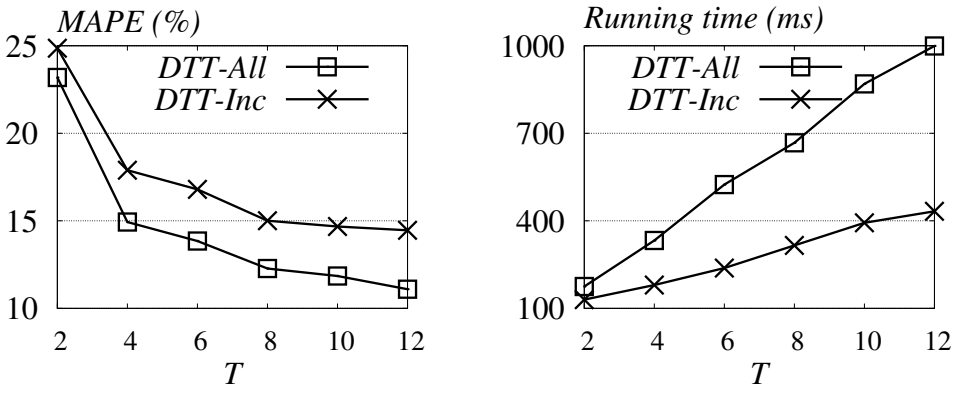
(a) Rush hour on SMALL

(b) Non-Rush hour on SMALL

(c) Rush hour on LARGE

(d) Non-Rush hour on LARGE

Figure 13: Online Prediction time

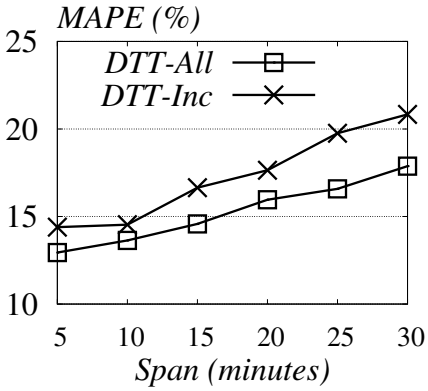


(a) Prediction error

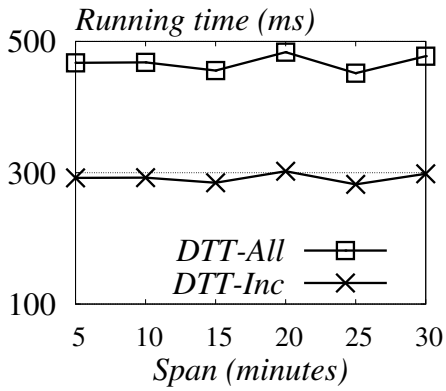
(b) Running time

Figure 14: Effect of Varying T

Fortunately the sensor dataset usually have high-resolution, therefore it is always better to use smaller span to learn the latent attributes. In addition, span does not affect the running time of both algorithms.

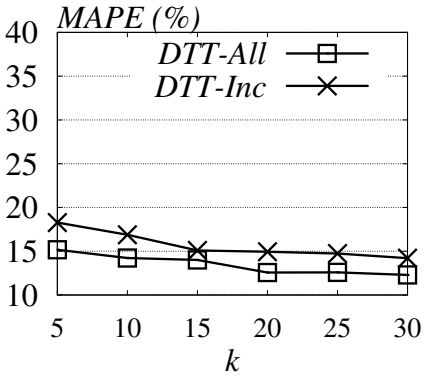


(a) Prediction error

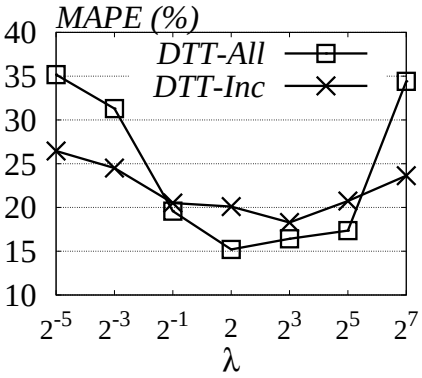


(b) Running time

Figure 15: Effect of Varying Span



(a) Prediction MAPE with k



(b) Prediction MAPE with λ

Figure 16: Effect of Varying k and λ on Prediction Accuracy.

7.7.3 Effect of Varying k and λ

Figure 16 (a) shows the effect of varying k. We have two main observations from this experiment: (1) we achieve better results with increasing number of latent attributes; (2) the performance stays stable when $k \geq 20$. This indicates a low-rank latent space representation can already capture the latent attributes of the traffic data. In addition, our results show that when the number of latent attributes is small (i.e., $k \leq 30$), the running time increased with k but does not change much when we vary k from 5 to 30. Therefore, setting k as 20 achieves a good balance between computational cost and accuracy.

Figure 16 (b) depicts the effect of varying λ, which is the regularization parameter for our graph Laplacian dynamics. We observe that the graph Laplacian has a larger impact on DTT-All algorithm than that on DTT-Inc. This is because λ controls how the global structure similarity contributes to latent attributes and DTT-All jointly learns those time-dependent latent attribute, thus λ has larger effect on DTT-ALL. In contrast, DTT-Inc adaptively updates the latent positions of a small number of changed vertices in limited localized view, and thus is less sensitive to the global structure similarity than DTT-ALL. In terms of parameters choices, $\lambda = 2$ and $\lambda = 8$ yields best results for DTT-All and DTT-Inc, respectively.

8 Conclusion

In this paper, we have studied the problem of real-time traffic prediction with missing values and missing sensors for large road networks. We proposed a dynamic traffic model where each vertex is associated with a set of latent attributes that captures both spatial (in network space) and temporal properties. Moreover, as these attributes are time-dependent, they also accurately estimate the traffic patterns and their evolution over time. To efficiently infer these time-dependent latent attributes, we developed both global and incremental learning algorithms on sensor data streams, enabling real-time traffic prediction under a batch window setting. Extensive experiments verified the effectiveness, flexibility and scalability of our model in identifying traffic patterns, completing missing values, and predicting future traffic conditions.

For future work, we plan to embed the current framework into real applications such as ride-sharing or vehicle routing system, to enable better navigation using accurate time-dependent traffic patterns. Another interesting direction is to incorporate other data sources (e.g., GPS, incidents) for more accurate traffic prediction.

References

[1] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, March 2011.

[2] R. Adhikari and R. Agrawal. A novel weighted ensemble technique for time series forecasting. In *Advances in Knowledge Discovery and Data Mining*, pages 38–49. Springer, 2012.

[3] B. W. Bader, T. G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, February 2015.

[4] M. Blondel, Y. Kubo, and U. Naonori. Online passive-aggressive algorithms for non-negative matrix factorization and completion. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 96–104, 2014.

[5] D. Cai, X. He, J. Han, and T. S. Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(8):1548–1560, 2011.

[6] H. Cheng and P.-N. Tan. Semi-supervised learning with data calibration for long-term time series forecasting. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–141. ACM, 2008.

[7] F. C. T. Chua, R. J. Oentaryo, and E.-P. Lim. Modeling temporal adoptions using dynamic matrix factorization. In *ICDM*, pages 91–100. IEEE, 2013.

[8] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006.

[9] J. Haworth and T. Cheng. Non-parametric regression for space–time forecasting under missing data. *Computers, Environment and Urban Systems*, 36(6):538–550, 2012.

[10] HERE. `https://company.here.com/here/`.

[11] T. Idé and M. Sugiyama. Trajectory regression on road networks. In *AAAI*, 2011.

[12] H. Kim and H. Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM Journal on Matrix Analysis and Applications*, 30(2):713–730, 2008.

[13] J. Kwon and K. Murphy. Modeling freeway traffic with coupled hmms. Technical report.

[14] R. Lambiotte, J.-C. Delvenne, and M. Barahona. Laplacian dynamics and multiscale modular structure in networks. *arXiv preprint arXiv:0812.1770*, 2008.

[15] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[16] K.-R. Müller, A. J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik. Predicting time series with support vector machines. In *Artificial Neural Networks-ICANN*, pages 999–1004. Springer, 1997.

[17] B. Pan, U. Demiryurek, and C. Shahabi. Utilizing real-world transportation data for accurate traffic prediction. ICDM '12, pages 595–604, Washington, DC, USA, 2012.

[18] B. Pan, U. Demiryurek, C. Shahabi, and C. Gupta. Forecasting spatiotemporal impact of traffic incidents on road networks. In *Data Mining (ICDM)*, pages 587–596. IEEE, 2013.

[19] L. Qu, Y. Zhang, J. Hu, L. Jia, and L. Li. A bpca based missing value imputing method for traffic flow volume data. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 985–990. IEEE, 2008.

[20] G. Ristanoski, W. Liu, and J. Bailey. Time series forecasting using distribution enhanced linear regression. In J. Pei, V. Tseng, L. Cao, H. Motoda, and G. Xu, editors, *Advances in Knowledge Discovery and Data Mining*, volume 7818 of *Lecture Notes in Computer Science*, pages 484–495. Springer Berlin Heidelberg, 2013.

[21] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*, volume 589. John Wiley & Sons, 2005.

[22] A. Saha and V. Sindhwani. Learning evolving and emerging topics in social media: a dynamic nmf approach with temporal regularization. In *WSDM*, pages 693–702. ACM, 2012.

[23] Y. Wang, Y. Zheng, and Y. Xue. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 25–34. ACM, 2014.

[24] L. Wu, X. Xiao, D. Deng, G. Cong, A. D. Zhu, and S. Zhou. Shortest path and distance queries on road networks: An experimental evaluation. *Proceedings of the VLDB Endowment*, 5(5):406–417, 2012.

[25] J. Xu, D. Deng, U. Demiryurek, C. Shahabi, and M. van der Schaar. Mining the situation: spatiotemporal traffic prediction with big data.

[26] B. Yang, C. Guo, and C. S. Jensen. Travel cost inference from sparse, spatio temporally correlated time series using markov models. *Proceedings of the VLDB Endowment*, 6(9):769–780, 2013.

[27] B. Yang, M. Kaul, and C. S. Jensen. Using incomplete information for complete weight annotation of road networks. *Knowledge and Data Engineering, IEEE Transactions on*, 26(5):1267–1279, 2014.

[28] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *WSDM*, pages 587–596. ACM, 2013.

[29] B. Zhang, K. Xing, X. Cheng, L. Huang, and R. Bie. Traffic clustering and online traffic prediction in vehicle networks: A social influence perspective. In *INFOCOM, 2012 Proceedings IEEE*, pages 495–503, March 2012.

[30] Y. Zhang and D.-Y. Yeung. Overlapping community detection via bounded nonnegative matrix tri-factorization. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 606–614. ACM, 2012.

[31] J. Zheng and L. M. Ni. Time-dependent trajectory regression on road networks via multi-task learning. In *AAAI*, 2013.

[32] J. Zhou and A. K. Tung. Smiler: A semi-lazy time series prediction system for sensors. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 1871–1886, 2015.

[33] L. Zhu, A. Galstyan, J. Cheng, and K. Lerman. Tripartite graph clustering for dynamic sentiment analysis on social media. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1531–1542. ACM, 2014.

[34] I. Žliobaitė. Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*, 2010.

9 Appendix

9.1 Derivatives of L with Respect to U_t in Equation 7.

The objective of L could be rewritten as follows:

$$
\begin{aligned}
&L = J_1 + J_2 + J_3 + \sum_{t=1}^{T} Tr(\psi_t U_t) \\
&\text{where:} \\
&J_1 = \sum_{t=1}^{T} Tr\Big(\big(Y_t \odot (G_t - U_t B U_t^T)\big)\big(Y_t \odot (G_t - U_t B U_t^T)\big)^T\Big) \\
&J_2 = \sum_{t=1}^{T} \lambda Tr(U_t L U_t^T) \\
&J_3 = \sum_{t=2}^{T} Tr\Big((U_t - U_{t-1}A)(U_t - U_{t-1}A)^T\Big)
\end{aligned}
\tag{16}
$$

J_1 could also be rewritten as follows:

$$
\begin{aligned}
J_1 =& \sum_{t=1}^{T} Tr\Big(\big(Y_t \odot (G_t - U_t B U_t^T)\big)\big(Y_t \odot (G_t - U_t B U_t^T)\big)^T\Big) \\
=& \sum_{t=1}^{T} Tr\big((Y_t \odot G_t)^T (Y_t \odot G_t) - 2(Y_t^T \odot G_t^T)(Y_t \odot U_t B U_t^T) \\
&+ (Y_t^T \odot U_t B^T U_t^T)(Y_t \odot U_t B U_t^T)\big) \\
=& \, const - 2\sum_{t=1}^{T} Tr\big((Y_t^T \odot G_t^T)(Y_t \odot U_t B U_t^T)\big) \\
&+ \sum_{t=1}^{T} Tr\big((Y_t^T \odot U_t B^T U_t^T)(Y \odot U_t B U_t^T)\big)
\end{aligned}
\tag{17}
$$

The second item of equation 17 could be transformed by:

$$\begin{aligned} O_1 &= \sum_{t=1}^{T} Tr\big((Y_t^T \odot G_t^T)(Y_t \odot U_t B U_t^T)\big) \\ &= \sum_{t=1}^{T}\sum_{k=1}^{n} \big((Y_t^T \odot G_t^T)(Y_t \odot U_t B U_t^T)\big)_{kk} \\ &= \sum_{t=1}^{T}\sum_{k=1}^{n}\sum_{i=1}^{m} (Y_t^T \odot G_t^T)_{ki}(Y_t \odot U_t B U_t^T)_{ik} \\ &= \sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{k=1}^{n} (Y_t \odot G_t \odot Y_t \odot U_t B U_t^T)_{ik} \\ &= \sum_{t=1}^{T} Tr\big((Y_t^T \odot G_t^T \odot Y_t^T) U_t B U_t^T\big) \end{aligned} \tag{18}$$

Now J_1 could be written as follows:

$$\begin{aligned} J_1 = const &- 2\sum_{t=1}^{T} Tr\big((Y_t^T \odot G_t^T \odot Y_t^T) U_t B U_t^T\big) \\ &+ \sum_{t=1}^{T} Tr\big((Y_t^T \odot U_t B^T U_t^T)(Y \odot U_t B U_t^T)\big) \end{aligned} \tag{19}$$

We now take the derivative of L in respect of U_t:

$$\frac{\partial L}{\partial U_t} = \frac{\partial J_1}{\partial U_t} + \frac{\partial J_2}{\partial U_t} + \frac{\partial J_3}{\partial U_t} + \frac{\partial \sum_{t=1}^{T} Tr(\psi_t U_t)}{\partial U_t} \tag{20}$$

The derivative of $\frac{\partial J_1}{\partial U_t}$ now could be calculated as follows:

$$\begin{aligned} \frac{\partial J_1}{\partial U_t} = &-2(Y_t \odot G_t \odot Y_t)(U_t B^T + U_t B) \\ &+ \frac{\partial \sum_{t=1}^{T} Tr\big((Y_t^T \odot U_t B^T U_t^T)(Y \odot U_t B U_t^T)\big)}{\partial U_t} \end{aligned} \tag{21}$$

Suppose $O_2 = \sum_{t=1}^{T} Tr\big((Y_t^T \odot U_t B^T U_t^T)(Y \odot U_t B U_t^T)\big)$, the derivative of O_2 could be written as:

$$\begin{aligned} \frac{\partial O_2}{\partial U_t(pq)} &= \frac{\partial \sum_{k=1}^{n}\sum_{i=1}^{m}(Y_t^T \odot U_t B^T U_t^T)_{ki}(Y \odot U_t B U_t^T)_{ik}}{\partial U_t(pq)} \\ &= \frac{\partial \sum_{i=1}^{m}\sum_{k=1}^{n}(Y_t \odot Y_t \odot U_t B U_t^T \odot U_t B U_t^T)_{ik}}{\partial U_t(pq)} \\ &= \frac{\partial \sum_{i=1}^{m}\sum_{k=1}^{n} Y_t^2(ik)(U_t B U_t^T)^2(ik)}{\partial U_t(pq)} \end{aligned} \tag{22}$$

Because only the p_{th} row of $U_t B U_t^T$ is related with $\frac{\partial O_2}{\partial U_t(pq)}$, we have the following:

$$\begin{aligned}\frac{\partial O_2}{\partial U_t(pq)} &= \frac{\partial \sum_{k=1}^{n} Y_t^2(pk)(U_t B U_t^T)^2(pk)}{\partial U_t(pq)} \\ &= 2\sum_{k=1}^{n}(Y_t^2)_{pk}(U_t B U_t^T)_{pk}\frac{\partial (U_t B U_t^T)_{pk}}{\partial (U_t)_{pq}} \\ &= 2\sum_{k=1}^{n}(Y_t^2)_{pk}(U_t B U_t^T)_{pk}(U_t B^T + U_t B)_{kq}\end{aligned} \tag{23}$$

The matrices derivation is then expressed as:

$$\begin{aligned}\frac{\partial O_2}{\partial U_t} &= 2(Y_t \odot Y_t \odot U_t B U_t^T)(U_t B^T + U_t B) \\ &= 2(Y_t \odot U_t B U_t^T)(U_t B^T + U_t B)\end{aligned} \tag{24}$$

Now the derivative of $\frac{\partial J_1}{\partial U_t}$ is as follows:

$$\begin{aligned}\frac{\partial J_1}{\partial U_t} =& -2(Y_t \odot G_t \odot Y_t)(U_t B^T + U_t B) \\ &+ 2(Y_t \odot U_t B U_t^T \odot Y_t)(U_t B^T + U_t B)\end{aligned} \tag{25}$$

Similarly, we could calculate the derivatives of $\frac{\partial J_2}{\partial U_t} = 2\lambda L U_t$,$\frac{\partial J_3}{\partial U_t} = 2(U_t - U_{t-1}A) + 2(U_t A A^T - U_{t+1}A^T)$, and $\frac{\partial \sum_{t=1}^{T} Tr(\psi_t U_t^T)}{\partial U_t} = \psi_t$, we have the following:

$$\begin{aligned}\frac{\partial L}{\partial U_t} =& -2(Y_t \odot G_t)(U_t B^T + U_t B) + 2(Y_t \odot U_t B U_t^T)(U_t B^T + U_t B) \\ &+ 2\lambda L U_t + 2(U_t - U_{t-1}A) + 2(U_t A A^T - U_{t+1}A^T) + \psi_t\end{aligned} \tag{26}$$

9.2 Update Rule of A and B

Similar with the derivation of U_t, we add Lagrangian multiplier with $\phi \in R^{k\times k}$ and $\omega \in R^{k\times k}$, and calculate the derivatives of L in respect of A and B :

$$\begin{aligned}\frac{\partial L}{\partial B} &= -2\sum_{t=1}^{T} U_t^T(Y_t \odot G)U_t + 2\sum_{t=1}^{T} U_t^T(Y_t \odot U_t B U_t^T)U_t + \phi \\ \frac{\partial L}{\partial A} &= -2\sum_{t=2}^{T} U_{t-1}^T U_t + 2\sum_{t=2}^{T} U_{t-1}^T U_{t-1} A + \omega\end{aligned} \tag{27}$$

Using the KKT conditions $\phi_{ij}B_{ij} = 0$ and $\omega_{ij}A_{ij} = 0$, we get the following equations for B_{kk}, and A_{kk}:

$$-\big(\sum_{t=1}^{T} U_t^T(Y_t \odot G)U_t\big)_{ij}B_{ij} + \big(\sum_{t=1}^{T} U_t^T(Y_t \odot U_t B U_t^T)U_t\big)_{ij}B_{ij} = 0 \tag{28}$$

$$-\big(\sum_{t=2}^{T} U_{t-1}^T U_t\big)_{ij}A_{ij} + \big(\sum_{t=2}^{T} U_{t-1}^T U_{t-1}A\big)_{ij}A_{ij} = 0 \tag{29}$$

These lead us to the following update rules:

$$B_{ij} \leftarrow B_{ij}\Big(\frac{[\sum_{t=1}^{T} U_t^T(Y_t \odot G_t)U_t]_{ij}}{[\sum_{t=1}^{T} U_t^T(Y_t \odot (U_t B U_t^T))U_t]_{ij}}\Big) \tag{30}$$

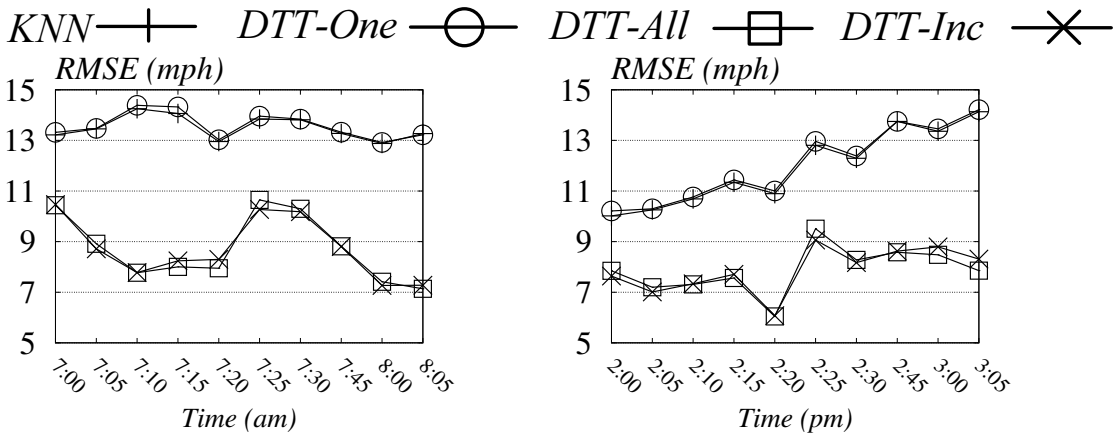
$$A_{ij} \leftarrow A_{ij}\Big(\frac{[\sum_{t=1}^{T} U_{t-1}^T U_t]_{ij}}{[\sum_{t=1}^{T} U_{t-1}^T U_{t-1} A]_{ij}}\Big) \tag{31}$$

9.3 Extra Experiment Results Based on Root Mean Square Error (RMSE)

In this set of experiments, we show the experiment results according to the measurement of RMSE, which indicates how closest predictions are to true observations. The definition of RMSE is as follows:

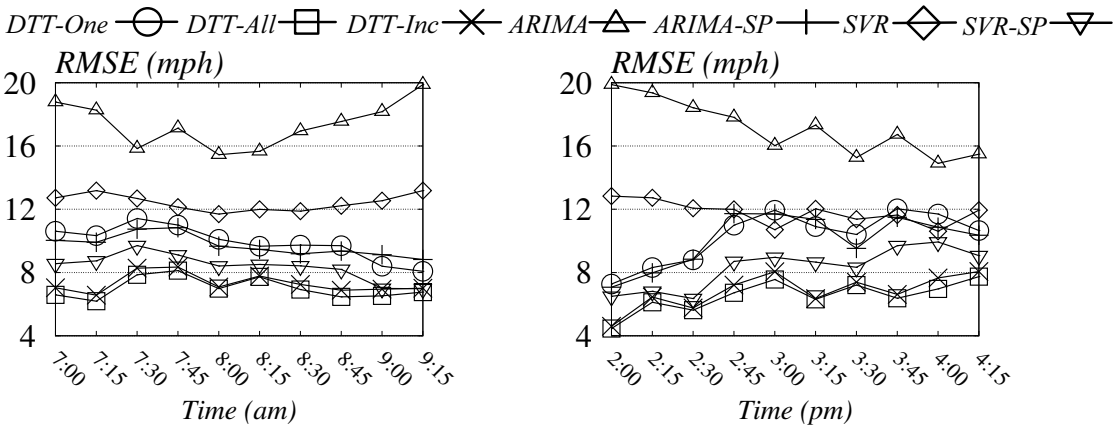
$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

Figure 17 shows the experiment results for missing value completion. The one-step ahead prediction results are shown in Figure 18, six-step ahead prediction results are depicted in Figure 19. Figure 20 shows the experiment results of online setting. The results based on RMSE are similar with those based on MAPE. We also observe that the predicated value by our methods deviate a small range (e.g., 4 to 11 mph) compared with the ground truth value.



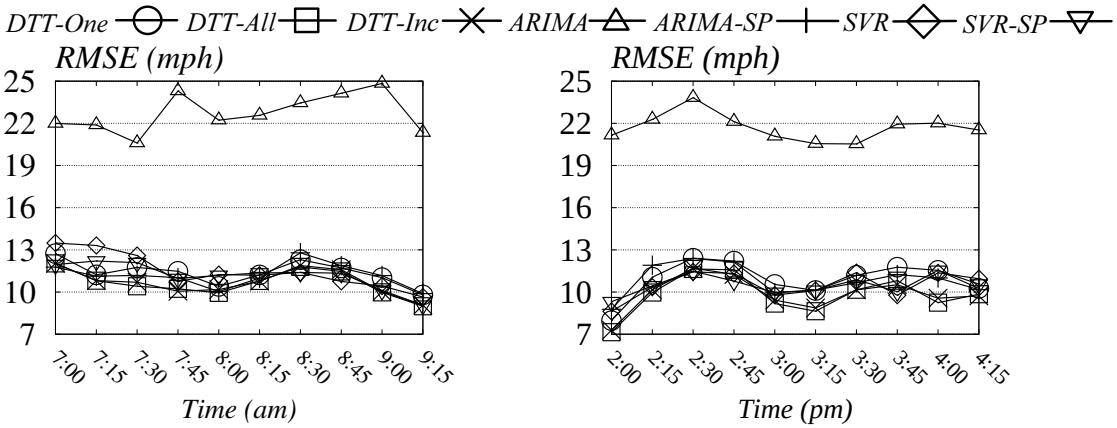
(a) Rush hour (b) Non-Rush hour

Figure 17: Missing Value Completion RMSE on SMALL



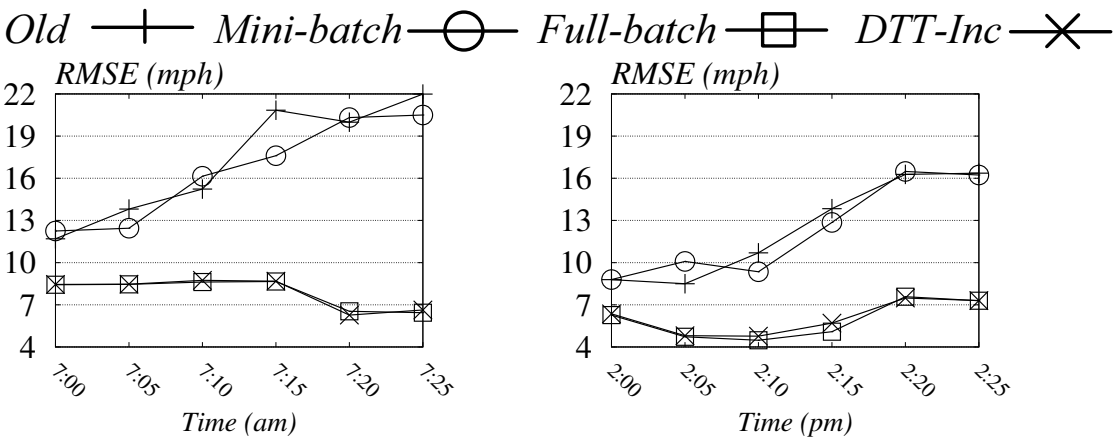
(a) Rush hour (b) Non-Rush hour

Figure 18: Edge Traffic Prediction RMSE One Step on SMALL



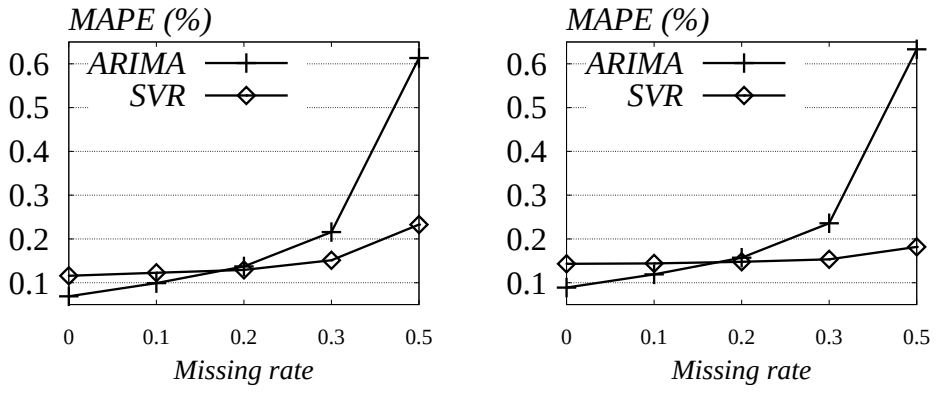
(a) Rush hour (b) Non-Rush hour

Figure 19: Edge Traffic Prediction RMSE Six Steps Ahead on SMALL



(a) Rush hour (b) Non-Rush hour

Figure 20: Online Prediction RMSE on SMALL



(a) SMALL (b) LARGE

Figure 21: Missing Rate During Training Stages for Support Vector Regression (SVR) and Auto-Regressive Integrated Moving Average (ARIMA)

9.4 Effect of Missing Data

In this set of experiment, we analyze the effect of missing data on the training dataset for the time series prediction techniques (i.e., ARIMA and SVR). The results are shown in Figure 21. As shown in Figure 21 (a) and (b), the prediction error for both approaches increases with more number of noise. Similar to the effect of missing value on the prediction stages shown in Figure 9, ARIMA is less robust than SVR because of its linear model. One interesting observation is that ARIMA performs better than SVR if the missing ratio is less than 10%, this indicates ARIMA is a good candidate for accurate traffic condition under the presence of complete data, this also conforms with the experiment results on [17]. However, ARIMA is sensitive to the missing values during both training and prediction stages, which renders poor performance with incomplete dataset.