Leveraging Contextual Graphs for Stochastic Weight Completion in Sparse Road Networks

Xiaolin Han* Reynold Cheng* Tobias Grubenmann† Silviu Maniu‡ Chenhao Ma*§
Xiaodong Li¶

Abstract

Road network applications, such as navigation, incident detection, and Point-of-Interest (POI) recommendation, make extensive use of network edge weights (e.g., traveling times). Some of these weights can be missing, especially in a road network where traffic data may not be available for every road. In this paper, we study the *stochastic weight completion* (SWC) problem, which computes the weight distributions of missing road edges. This is difficult, due to the intricate temporal and spatial correlations among neighboring edges. Moreover, the road network can be *sparse*, i.e., there is a lack of traveling information in a large portion of the network. To tackle these challenges, we propose the **Con**textual **G**raph **C**ompletion (ConGC). We propose to incorporate the contextual properties about the road network (e.g., speed limits, number of lanes, road types) to provide finer granularity of spatial correlations. Moreover, ConGC incorporates temporal and periodic dimensions of the road traffic. We evaluate ConGC against existing methods on three real road network datasets. They show that ConGC is more effective and efficient than state-of-the-art solutions.

1 Introduction

A road network enables Big Transportation Data applications (e.g., navigation, incident detection [28, 29], and POI recommendation). The road network can be treated as a graph with vertices and edges [24, 25, 26, 27, 30, 31]. The edge weights of the road network are extensively used by these applications. A navigation app, for instance, returns a path to a user with the smallest sum of edge weights along the path.

However, it is common for a road network to have edges whose weights are missing [10, 11, 12, 13]. For example, in a dataset that contains GPS of taxis in Hong Kong during 2010, the taxi locations are concentrated in commercial and residential districts, covering only 1.4% of all roads in Hong Kong. And roads in rural areas are seldom visited by vehicles. During late nights, most roads are traversed by few vehicles, making it difficult to collect traffic data. To tackle this issue, a few *weight completion* (WC) methods have been proposed.

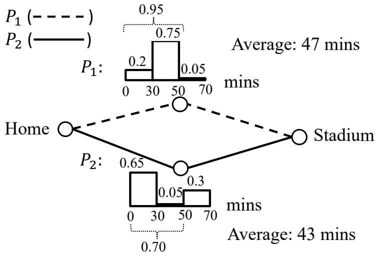


Figure 1: An example of stochastic edge weights.

In general, WC problems can be classified into two types: *deterministic* and *stochastic*. Most WC methods are targeted towards deterministic problems (e.g., [3, 5, 22]). They learn deterministic, or non-probabilistic, weights for edges (e.g., the average traveling time of Nathan Road in Hong Kong between 10-10:30 am on Sunday is 5 minutes on average). However, these methods do not perform well in stochastic settings [1]. In addition, these methods often ignore the sparsity of traffic data, which leads to inaccurate results. In contrast, very few methods are targeting the *stochastic* WC problem (e.g., A-GCWC [1]), and are optimized towards learning the *distributions* of edge weights (e.g., the traveling time of Nathan Road during Sunday 10-10:30 am is 5 minutes with 80% probability, and 10 minutes with 20% probability). Such methods, which take into account the time-varying uncertainty of real-time traffic conditions, give more precise weights than their counterparts targeting only deterministic weights.

As shown in [7, 8, 9, 17], stochastic WC improves the accuracy of path routing. To illustrate this, suppose that a person departs from her home to catch a football event in Figure 1. She needs to arrive at the stadium within 50 minutes. There are two possible paths: P_1

[*] The University of Hong Kong. {xlhan, ckcheng, chma2}@cs.hku.hk

[†] University of Bonn. grubenmann@cs.uni-bonn.de

[‡] Universite Paris-Saclay. silviu.maniu@lri.fr

[§] Chenhao Ma is the corresponding author.

[¶] National University of Singapore. xdli@nus.edu.sg and P_2. If average time is used, P_2 should be chosen, since it requires a lower cost (43 minutes). However, if the time distributions are considered, then P_1 is the preferred choice, because there is a higher chance (95%) for P_1 to be completed within 50 minutes (compared to 70% for P_2). By using stochastic weights, the chance of finding a better route can be improved.

Despite the benefits of using stochastic weights, it has not been well studied. The best solution for *stochastic weight completion* (SWC) so far is A-GCWC [1], which uses a graph convolutional neural network to propagate weights from edges whose traffic data is available to the edges with missing values. However, it does not perform well on very sparse traffic data (e.g., the Hong Kong dataset mentioned) in our experiments.

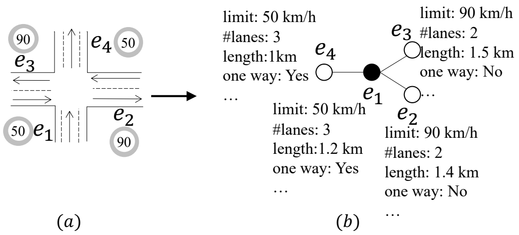


Figure 2: (a) A road network; (b) a contextual graph.

In this paper, we propose a novel SWC solution called **Con**textual **G**raph **C**ompletion (ConGC). We utilize a *contextual graph*, which describes road information such as speed limits, number of lanes, road types, to provide finer granularity of spatial correlations. Figure 2 illustrates a *contextual graph*. The intuition is that road properties depicted by a contextual graph provide informative contexts about road similarity among neighbors. As we will show, it is helpful especially when traffic data is sparse, because edge correlations captured from contextual graph enhance the performance.

A salient feature of ConGC is that it incorporates not just the spatial dimension of the traffic data, but also two aspects of the temporal dimension – recency and periodicity. The intuition is that recent and periodic traffic has a high correlation with the target time. By propagating traffic data along both space and time dimensions, more data could be provided to learn the stochastic weights for the edge concerned, which also alleviates the data sparsity problem.

To summarize, our contributions are:

- We present the ConGC, which collectively leverages contextual graph with traffic dynamics to provide finer granularity of spatial correlations.
- ConGC incorporates topological traffics, recent trends, and periodic patterns to effectively and efficiently complete stochastic weights.
- We have performed substantial experiments against exsiting methods. ConGC is more effective than both deterministic WC methods and stochstic WC methods. And it is efficient on million-scale road networks.

2 Related Work

There are two kinds of WC problems: deterministic and stochastic. Most existing deterministic works target on predicting traffic based on past data. They use location data collected from static sensors installed along the freeways. Because these data are regularly obtained, they are “dense”. However, this assumption may not be valid when road sensors are not available. In such cases, we may still obtain vehicle location data by some other means (e.g., GPS); however, the traffic data so obtained is more sparse, making the WC more challenging.

Deterministic Weight Completion: Deterministic models focus on dense data. CNN based models [22] have been proposed to predict traffic speed or crowd density. DSAE [23] utilizes denoising stacked autoencoders to fill in weights. DCRNN [3] has been proposed to forecast traffic data by capturing spatial-temporal correlations. Spatial-temporal GCN [14, 15, 19, 20, 21] has been applied to forecast traffic data and citywide passenger demand by incorporating spatial-temporal correlation. Attention based spatial-temporal graph convolutional networks [5] are further proposed to pay different attention to nearby points in the graph. Only fewer model [2] considers the sparsity, which has been shown to perform worse than the existing stochastic WC model in [1]. Since most work ignores the sparsity in traffic data, the performance on sparse data is negatively affected.

Stochastic Weight Completion: A-GCWC [1] uses spectral-domain-based GCN to complete stochastic weights. It incorporates additional information, such as the time interval flag, by a Bayesian inference model. In terms of temporal dimension, the traffic behavior is assumed to be conditionally dependent on the time interval flag, which occurs periodically with a time period p. Given two time intervals T_1 and T_2 that are p time units apart, the traffic conditions of them are supposed to be the same in [1]. In fact, this is not always the case due to dynamic factors, e.g., road construction or incidents. Different from A-GCWC, our solution can capture dynamic changes, and learn the difference between T_1 and T_2. Our solution also considers recency and periodicity information, as well as contextual graph.

The research on GCN has two branches, which are the spectral [16] and node domains [6]. Most traffic studies follow the spectral domain. However, the model trained on one graph cannot be used to graphs with different structures since it is based on Laplacian

Eigenbasis [6]. Moreover, it requires high computation cost, e.g., matrix inversion. Fewer studies [3] follow the node domain. Although it has been shown its benefits on many tasks [6], it has not been well studied in sparse traffic. Therefore, we follow the node domain to study the possibility of applying it to sparse traffic.

3 Problem Definition

Definition 3.1. (Road Network) *A road network is a graph where each vertex $v \in V$ represents a road intersection. Edge $e = (v_1, v_2) \in E \subseteq V \times V$ indicates that intersections v_1 and v_2 are directly connected.*

Definition 3.2. (Stochastic Weight) *Let $e \in E$ be a road, the stochastic weight of e at time interval T_i, $h_{e,T_i} \in \mathbb{R}^{|B|}$, is its travel cost distribution. Each stochastic weight consists of a set B of buckets which describe the histogram of the distribution.*

In Definition 3.1, we capture each traveling direction of each road separately. However, we are not directly interested in the travelling direction but instead, we are only interested in whether the road directions allow a vehicle to travel from one road to the next road. We are interested if two roads are *spatial neighbors*.

Definition 3.3. (Edge Graph) *A directed road network can be transformed to an undirected edge graph $G = (E, A)$, in which A is adjacency matrix that captures the connectivity of the edges in E. $A_{e_i,e_j} = A_{e_j,e_i} = 1$, $e_i, e_j \in E$, if and only if a vehicle can travel from e_i to e_j or from e_j to e_i by passing through exactly one intersection $v \in V$, $A_{e_i,e_j} = A_{e_j,e_i} = 0$ otherwise.*

Definition 3.4. (n-th Order Spatial Neighbors) *Given $n \in \mathbb{N}$, an edge graph G, two edges $e_i, e_j \in E$, the edges e_i and e_j are n-th order spatial neighbors of each other if and only if a vehicle can travel from e_i to e_j or from e_j to e_i by passing through exactly n intersections. We denote with $N^n_{\text{spat.}}(e)$ the set of all spatial neighbors of e with order less or equal to n.*

Definition 3.5. (n-th Order Recent Neighbors) *Given an interval $T_i \in \mathcal{T}$, the n-th order recent neighbor for $n < i$ of T_i is the time interval T_{i-n}. We denote with $N^n_{\text{temp.}}(T)$ the set of all recent neighbors of time interval T with order less or equal to n.*

Definition 3.6. (n-th Order Periodic Neighbors) *Given a period $p \in \mathbb{N}$ and a time interval $T_i \in \mathcal{T}$, the n-th order periodic neighbor for $n \cdot p < i$ of T_i is the interval $T_{i-n\cdot p}$. We denote with $N^{n,p}_{\text{period.}}(T)$ the set of all periodic neighbors of T with order less or equal to n.*

Problem Definition. Given $|\mathcal{T}|$ time intervals, $|E|$ roads, and bucket size $|B|$ for the stochastic weight tensor $W \in \mathbb{R}^{|E|\times|\mathcal{T}|\times|B|}$, we denote $h_{e,T} \in \mathbb{R}^{|B|}$ as the stochastic weight vector for the edge e at time interval T. Due to the sparsity issue, W might have many missing values. Stochastic Weight Completion aims to reconstruct a tensor $\widehat{W}$ by filling missing values in W.

4 The ConGC Model

In this section, we first introduce the framework of ConGC (Section 4.1). After that, we explain the detailed steps in the model (Sections 4.2 – 4.8). Moreover, we show the time complexity of ConGC (Section 4.9).

4.1 Framework Figure 3 shows the framework. It follows the encoder-decoder structure, in which Topological Traffic Propagator (Step ①), Contextual Traffic Diffusion (Step ②), Recent Trend Aggregator (Step ③), Periodic Pattern Explorer (Step ④), and Pooling (Step ⑤) form the encoder, and Fully Connected Layer (⑥) forms the decoder. The idea is to encode the data by extracting key information from transformation, then reconstruct the actual one by decoding it.

4.2 Topological Traffic Propagator The spatial neighbors play an important role on the target edge.

Challenges: The importances of different spatial neighbors are not exactly the same. Besides, there may be some neighbors without vehicles travelling at the time interval T. Considering these neighbors with missing data may degrade the performance.

Design: Topological Traffic Propagator learns the importance scores of the edge's informative spatial neighbors, and updates the edge by aggregating these neighbors. The intuition is that among the edge's spatial neighbors, there are some neighbors following the similar traffic condition as the target edge, but some may have different traffic conditions. For example, in Figure 3 (a), road e_1 should pay more attention to road e_2 since vehicles can turn right when the red light is on. But road e_3 and e_4 have less correlation with road e_1 compared to road e_2. Inspired by attention mechanism [6] which was originally proposed to learn attention for neighbors in the standard graph, we propose to exploit hidden correlations in sparse traffic data which contain complex information with missing data in neighbors.

The stochastic weight $h_{e_i,T} \in \mathbb{R}^{|B|}$ is transformed to latent embedding to allow deep expression:

$$H_{e_i,T} = U\ h_{e_i,T}\ , \tag{4.1}$$

where $U \in \mathbb{R}^{B'\times|B|}$ is the learnable parameter, and $H_{e_i,T} \in \mathbb{R}^{B'}$ is the latent representation of edge e_i at T.

The importance of edge e_j to edge e_i at the time

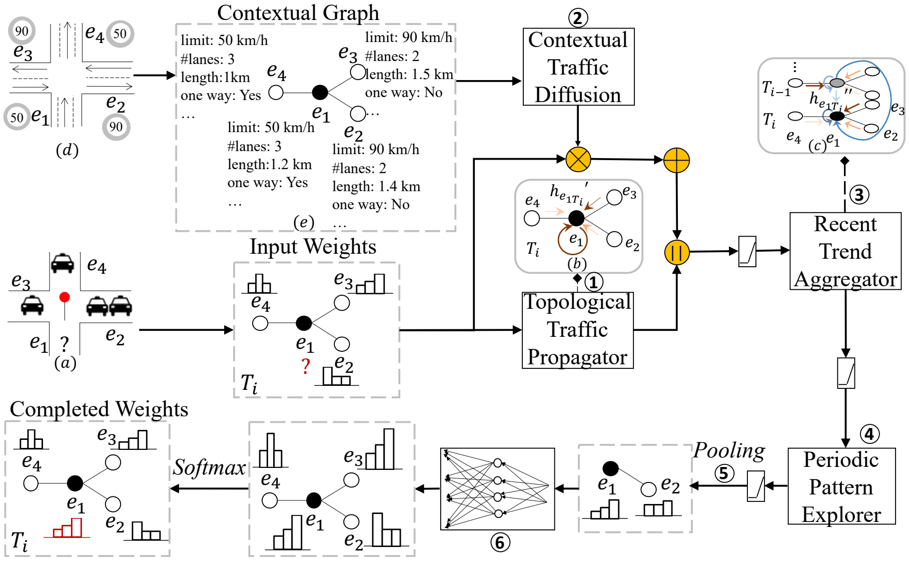


Figure 3: The overall framework of the ConGC model. Details of each step are in Sections 4.2 to 4.7.

interval T is:

$$\eta^{\text{spat.}}_{e_i,e_j,T} = \text{ReLU}(a^{\text{tr}} \cdot [H_{e_i,T}||H_{e_j,T}]) \ , \tag{4.2}$$

where ReLU is the activation function, a^{tr} is the transpose of the trainable parameter vector a, $\|$ is the concatenation operator, $e_j \in N^n_{\text{spat.}}(e_i)$ is the at most n-th order spatial neighbors of e_i.

The spatial traffic attention can be calculated as:

$$\alpha^{\text{spat.}}_{e_i,e_j,T} = \frac{\exp(\eta^{\text{spat.}}_{e_i,e_j,T})}{\sum_{e_k \in \text{Mask}_T(N^n_{\text{spat.}}(e_i))} \exp(\eta^{\text{spat.}}_{e_i,e_k,T})} \ , \tag{4.3}$$

where $\text{Mask}_T(N^n_{\text{spat.}}(e_i)) = \{e_j \mid h_{e_j,T} \text{ is valid}, e_j \in N^n_{\text{spat.}}(e_i)\}$ is the masked set of spatial neighbors that only contain edges with traffic data at time interval T.

The spatial graph convolution is calculated as:

$$h^{\text{spat.}}_{e_i,T} = \sum_{e_j \in \text{Mask}_T N^n_{\text{spat.}}(e_i)} \alpha^{\text{spat.}}_{e_i,e_j,T} \cdot H_{e_j,T} \ , \tag{4.4}$$

where $h^{\text{spat.}}_{e_i,T} \in \mathbb{R}^{B'}$ is the updated embedding of e_i at T after spatial attentional graph convolution.

4.3 Contextual Traffic Diffusion Roads in the road network have properties, e.g., speed limit, the number of lanes, length, road type, and one way flag. These properties in the road network form a contextual graph (Figure 3 (e)). The insight is that they can provide useful information about road similarity in nature, which is important especially under data sparsity scenarios.

Challenges: The dynamic graph depicted in Figure 3 (b) (Section 4.2) contains missing values as there exist edges having no vehicles traversed at some time intervals. It makes weights update based on neighbor pairs' attention, i.e., Equations 4.2 and 4.3, inaccurate. Take Figure 3 (a) as an example, edge e_1 contains missing values, then attention scores between e_1 and its neighbors become imprecise. In contrast, contextual graph is a static graph that provides properties of roads. It is different from the dynamic graph, which models dynamic weights changing when traffic is updating. Hence, contextual graph is essential since it contains road similarity information to supplement imprecise attention scores. However, the integration of static and dynamic graphs is non-trivial.

Design: The ConGC updates based on road properties in contextual graph and traffic condition in dynamic graph collectively. The road properties can be transformed as contextual embeddings. For categorical features, e.g., the number of lanes nl, one way flag ow, road type rt, we apply one-hot encoding to transform them as f_{nl}, f_{ow} and f_{rt}. For continuous features, e.g., speed limit sl and road length rl, we use Binning strategy to transform them into discrete ones as f_{sl} and f_{rl}. Then we concatenate them as $f_c \in \mathbb{R}^{|E| \times nf}$:

$$f_c = f_{nl} \ || \ f_{ow} \ || \ f_{rt} \ || \ f_{sl} \ || \ f_{rl} \ , \tag{4.5}$$

where nf is the dimension of feature f_c of edge e_i.

We apply a transformation to obtain the latent feature of graph contexts as:

$$F_{c_{e_i}} = R \ f_{c_{e_i}} \ , \tag{4.6}$$

where $f_{c_{e_i}} \in \mathbb{R}^{nf}$ is the contextual embedding of edge e_i, $R \in \mathbb{R}^{B' \times nf}$ is the learnable parameter, and $F_{c_{e_i}} \in \mathbb{R}^{B'}$ is the latent context representation of edge e_i.

The contextual importance of edge e_j to edge e_i is:

$$\eta^{\text{cont.}}_{e_i,e_j} = \text{ReLU}(d^{\text{tr}} \cdot [F_{c_{e_i}} || F_{c_{e_j}}]) \ , \tag{4.7}$$

where d is the trainable parameter vector. The contextual similarity score $\alpha^{\text{cont.}}_{e_i,e_j} \in \mathbb{R}$ can be calculated as:

$$\alpha^{\text{cont.}}_{e_i,e_j} = \frac{\exp(\eta^{\text{cont.}}_{e_i,e_j})}{\sum_{e_k \in (N^n_{\text{spat.}}(e_i))} \exp(\eta^{\text{cont.}}_{e_i,e_k})} \ , \tag{4.8}$$

We then diffuse the transformed stochastic weight embedding $H_{e_j,T} \in \mathbb{R}^{B'}$ (Equation 4.1) based on the contextual similarity score $\alpha^{\text{cont.}}_{e_i,e_j}$ as:

$$h^{\text{cont.}}_{e_i,T} = \sum_{e_j \in N^n_{\text{spat.}}(e_i)} \alpha^{\text{cont.}}_{e_i,e_j} \cdot H_{e_j,T} \ . \tag{4.9}$$

The embedding $h^{'}_{e_i,T} \in \mathbb{R}^{2B'Q}$ of edge e_i at the time interval T after contextual and spatial graph convolution is updated as:

$$h^{'}_{e_i,T} = \|^{Q}_{q=1} \text{ ReLU} \left(h^{\text{cont.}}_{e_i,T} || h^{\text{spat.}}_{e_i,T} \right) \ . \tag{4.10}$$

To make the learning robust, we concatenate the learned embedding $Q \in \mathbb{N}$ times as the multi-head attention.

4.4 Recent Trend Aggregator Temporal neighbors of edge e_i have high correlations with the target time interval T_j of e_i.

Challenges: These correlations differ among temporal neighbors. Moreover, traffic of temporal neighbors with missing data propagates low quality information via graph connections.

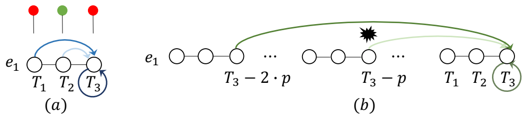


Figure 4: (a) Recent neighbors; (b) Periodic neighbors.

Design: Recent Trend Aggregator with a masked operator aims to calculate the importance scores of the edge's masked recent neighbors, then update the edge by aggregating its masked recent neighbors with importance scores. The intuition is that recent neighbors of different orders follow different degrees of traffic correlation with the target time interval T_i. For example, in Figure 4 (a), road e_1 at T_3 should pay more attention to its second-order recent neighbor at T_1 since the traffic lights are red at both T_1 and T_3. In contrast, the first-order recent neighbor T_2 has less correlation with T_3 compared to T_1, since the traffic light is green at T_2.

The $h^{'}_{e_i,T_j} \in \mathbb{R}^{2B'Q}$ is mapped to latent representation $H^{'}_{e_i,T_j} \in \mathbb{R}^{B'}$ to allow sufficient expression as:

$$H^{'}_{e_i,T_j} = O \ h^{'}_{e_i,T_j} \ , \tag{4.11}$$

where $O \in \mathbb{R}^{B' \times 2B'Q}$ is the learnable parameter.

The importance of T_k for T_j at e_i is defined as:

$$\eta^{\text{temp.}}_{T_j,T_k,e_i} = \text{ReLU}(p^{\text{tr}} \cdot [H^{'}_{e_i,T_j} || H^{'}_{e_i,T_k}]) \ , \tag{4.12}$$

where p is the trainable parameter vector, the time interval $T_k \in N^n_{\text{temp.}}(T_j)$, and $j - n \leq k \leq j$, n is the maximum order of recent neighbors.

The recent attention coefficient is defined as:

$$\alpha^{\text{temp.}}_{T_j,T_k,e_i} = \frac{\exp(\eta^{\text{temp.}}_{T_j,T_k,e_i})}{\sum_{T_l \in \text{Mask}_{e_i} N^n_{\text{temp.}}(T_j)} \exp(\eta^{\text{temp.}}_{T_j,T_l,e_i})} \ , \tag{4.13}$$

where $\text{Mask}_{e_i} N^n_{\text{temp.}}(T_j) = \{T_l \mid h_{e_i,T_l} \text{ is valid}, \ T_l \in N^n_{\text{temp.}}(T_j)\}$ is the set of masked recent neighbors of T_j at e_i that filters out neighbors with missing traffic data.

The embedding $h^{''}_{e_i,T_j} \in \mathbb{R}^{B'Q}$ is calculated as:

$$h^{''}_{e_i,T_j} = \|^{Q}_{q=1} \text{ ReLU} \left(\sum_{T_k \in \text{Mask}_{e_i} N^n_{\text{temp.}}(T_j)} \alpha^{\text{temp.}}_{T_j,T_k,e_i} H^{'}_{e_i,T_k} \right) , \tag{4.14}$$

4.5 Periodic Pattern Explorer Periodic Pattern Explorer with masked neighbors learns importance scores for informative periodic neighbors of the target edge e_i at time interval T, then integrates these periodic neighbors. The intuition is that periodic neighbors usually follow a similar correlation as the target one. However, there may be some divergence among them. For example, in Figure 4 (b), there is an incident happened at time interval $T_3 - p$ at edge e_1, where p is the period. Consequently, the correlation between T_3 and $T_3 - p$ is lower than that between T_3 and $T_3 - 2 \cdot p$. The period p could be 1-day or 1-week.

The $h^{''}_{e_i,T_j} \in \mathbb{R}^{B'Q}$ is transformed to latent embedding by parameter $P \in \mathbb{R}^{B' \times B'Q}$ as:

$$H^{''}_{e_i,T_j} = P \ h^{''}_{e_i,T_j} \ , \tag{4.15}$$

where $H^{''}_{e_i,T_j} \in \mathbb{R}^{B'}$ is the latent representation.

The importance of temporal periodic neighbor T_k for time interval T_j of edge e_i is defined as:

$$\eta^{\text{period.}}_{T_j,T_k,e_i} = \text{ReLU}(u^{\text{tr}} \cdot [H^{''}_{e_i,T_j} || H^{''}_{e_i,T_k}]) \ , \tag{4.16}$$

where u is the parameter vector, the time interval $T_k \in N^{n,p}_{\text{period.}}(T_j)$, and $j - n \cdot p \leq k \leq j$, n is the maximum order of periodic neighbors, p is the period.

The periodic traffic attention can be calculated as:

$$\alpha^{\text{period.}}_{T_j,T_k,e_i} = \frac{\exp(\eta^{\text{period.}}_{T_j,T_k,e_i})}{\sum_{T_l \in \text{Mask}_{e_i} N^{n,p}_{\text{period.}}(T_j)} \exp(\eta^{\text{period.}}_{T_j,T_l,e_i})}\ , \tag{4.17}$$

where $\text{Mask}_{e_i} N^{n,p}_{\text{period.}}(T_j) = \{T_l \mid h_{e_i,T_l} \text{ is valid, } T_l \in N^{n,p}_{\text{period.}}(T_j)\}$ is masked periodic neighbors of T_j at e_i that only contains periodic neighbors with traffic data.

The periodic graph convolution is calculated as:

$$h^{\text{updated}}_{e_i,T_j} = \|_{q=1}^{Q} \text{ ReLU}\left(\sum_{T_k \in \text{Mask}_{e_i} N^{n,p}_{\text{period.}}(T_j)} \alpha^{\text{period.}}_{T_j,T_k,e_i} H''_{e_i,T_k}\right) \tag{4.18}$$

where $h^{\text{updated}}_{e_i,T_j} \in \mathbb{R}^{B'Q}$ is the updated embedding after periodic graph convolution.

4.6 Pooling The $h^{\text{updated}}_{e_i,T_j}$ of all edges at all time intervals form the tensor W^{updated}. Then, Pooling is applied to extract key information from W^{updated} by a pooling size of k_t and k_e, where k_t is designed to extract temporal key information, and k_e is used for pooling important spatial information. Under traffic sparsity, there may still exist missing values in W^{updated} even after the operations we applied. Pooling is necessary since it only keeps key information from W^{updated} instead of missing values. After pooling, we obtain the encoded W^{PL} with a smaller size compared to W^{updated}.

Take the max Pooling on $W^{\text{updated}} \in \mathbb{R}^{|E|\times|\mathcal{T}|\times B'Q}$ in the spatial and temporal dimensions as an example:

$$W^{\text{PL}}_{e,t,bq} = \max_{i=k_e*e}^{k_e(e+1)-1} \max_{j=k_t*t}^{k_t(t+1)-1} W^{\text{updated}}_{i,j,bq}, \tag{4.19}$$

where $W^{\text{PL}} \in \mathbb{R}^{\frac{|E|}{k_e}\times\frac{|\mathcal{T}|}{k_t}\times B'Q}$ is the tensor after Pooling.

4.7 FC The Fully Connected Layers (FCs) are applied to decode key information which is encoded in W^{PL}, and restore the tensor to the original shape $|E| \times |\mathcal{T}| \times |B|$. After that, softmax is applied to make sure the sum of each stochastic weight equals one. Finally, the completed stochastic weight tensor $\widehat{W}$ is obtained.

4.8 Optimization After the encoder-decoder structure, we finally obtain the completed stochastic weight tensor $\widehat{W} \in \mathbb{R}^{|E|\times|\mathcal{T}|\times|B|}$. The goal is that $\widehat{W}$ is as close as the actual ground truth stochastic weight tensor W_G as possible, where $W_G \in \mathbb{R}^{|E|\times|\mathcal{T}|\times|B|}$ is set according to the label of model functionalities (see Section 5.1.3). Therefore, the loss function is formulated as:

$$\mathcal{L}(\widehat{W}, W_G) = \sum_{i=1}^{|\mathcal{T}|}\sum_{j=1}^{|E|} \mathbb{1}_{T_i,e_j} \cdot \text{KL}(\widehat{h}_{e_j,T_i} || h_{e_j,Ti})\ , \tag{4.20}$$

where $\text{KL}(\cdot||\cdot)$ measures the KL-divergence between the completed stochastic weight and the ground truth, and $\mathbb{1}_{T_i,e_j}$ is an indicator function that is 0 if data is missing in T_i at e_j, and 1 otherwise. The reason is that we can only establish the quality of the completed stochastic weight if the actual one is available as ground truth.

4.9 Complexity Analysis The time complexity of ConGC is $\mathcal{O}(Q \cdot |\mathcal{T}| \cdot |E| \cdot |B| \cdot B' \cdot (2 \cdot |N^{\max}_{\text{spat.}}| + |N^{n}_{\text{temp.}}| + |N^{n,p}_{\text{period.}}|))$, where $|N^{\max}_{\text{spat.}}|, |N^{n}_{\text{temp.}}|$, and $|N^{n,p}_{\text{period.}}|$ are the maximal number of spatial neighbors, the number of recent, and periodic neighbors, respectively. After removing constants, the complexity is dominated by $\mathcal{O}(|\mathcal{T}| \cdot |E|)$. Hence, ConGC has a polynomial time complexity, and is efficient as shown in Section 5.

5 Experiments

5.1 Experimental setup

5.1.1 Datasets We evaluate on three real datasets.

- The HK is a taxi GPS data set [1] in Hong Kong in 2010. It contains 35 gigabytes trajectories.
- The XN is an open GPS data set [2] in Xi'an in 2016 , which contains 137 gigabytes trajectories.
- The CD is an open GPS data set [2], which contains 196 gigabytes of GPS data in Chengdu in 2016.

We set the histogram with eight 5-m/s buckets ranging from 0m/s to 40m/s, and partition a day into 96 15-min intervals as [1].

5.1.2 Preprocessing After map matching [4], the two preprocessing steps are conducted.

Edge Graph Transformation. We choose the largest connected subgraph as [1]. In the HK, XN, and CD datasets, the numbers of selected edges are 1158, 64, and 175, respectively. We then transform the directed road network into an undirected edge graph.

Input Data Preparation. We construct ground truth stochastic weight W_G from GPS data. For input weight W, we construct it by randomly removing edge weights in W_G with removal ratio rm. Then we evaluate the quality of completed $\widehat{W}$ by comparing with W_G. Since W_G may contain missing values due to sparsity, we only use available data of W_G as ground truth.

5.1.3 Model functionalities Our method is flexible to support two model functionalities.

Estimation. The input is stochastic weight $W@T_i$ at T_i with missing values. The output is the completed $\widehat{W}@T_i$ at T_i. The label is the ground truth $W_G@T_i$.

[1] The dataset HK is a confidential dataset.

[2] http://outreach.didichuxing.com/research/opendata/

Prediction. The input is $W@T_i$ at T_i with missing values. The output is the predicted $\widehat{W@T_{i+1}}$ at the next T_{i+1}. The label is the ground truth $W_G@T_{i+1}$.

5.1.4 Competitors

We compare with 8 methods.

Deterministic weight completion methods: There are two kinds of deterministic methods. The first kind contains Random Forest (**RF**), Convolutional Neural Network (**CNN**) and **DSAE** [23], which only consider *spatial data*. Another kind incorporates *temporal data*. They are **DCRNN** [3], **ASTGCN** [5] and **ST-ResNet** [22]. We complete weights of each bucket in the histogram separately for RF as [1] for stochastic setting. For other learning based methods, we adapt them by changing their output size to $|B|$. The differences between ASTGCN and ours are three folds. First, they are designed on dense data. They ignore missing values in the model, which negatively affects performance. Second, they are designed for deterministic weights. Third, they follow the spectral GCN. We follow the node domain, which enables propagation of correlations based on structure. The first two differences hold for DCRNN.

Stochastic weight completion methods: **A-GCWC** [1] is the state-of-the-art model (GCN) for stochastic weight completion. We denote **GC** as the basic version of ConGC that does not involve contexts.

5.1.5 Hyperparameter tuning

We partition datasets into 5 folds as [1], where 4 folds for training and validation, and 1 fold for testing. We run 10 times in total, and report the average. We conduct hyper-parameter tuning by Bayesian optimizer. The scopes are learning rate [0.0001, 0.1], number of spatial, recent, periodic neighbors {2, 3, 4, 5}, p {“1 day”, “1 week”}, Q {4, 8, 16, 32}, k_t, k_e {2, 4, 8}, kernel number {8, 16, 32}, B' {200, 400, 600}, kernel size {8, 16, 32}.

5.1.6 Performance metrics

We evaluate by Mean Kullback-Leibler divergence Ratio (MKLR) and Fraction of Likelihood Ratio (FLR) as [1]. Historical Average (HA) is the average of training data. The smaller the MKLR is, the better the quality is.

$$MKLR = \frac{\sum_{i=1}^{|\mathcal{T}|}\sum_{j=1}^{|E|}\mathbb{1}_{T_i,e_j}\cdot \mathrm{KL}(h^G_{e_j,T_i}||\widehat{h}_{e_j,T_i})}{\sum_{i=1}^{|\mathcal{T}|}\sum_{j=1}^{|E|}\mathbb{1}_{T_i,e_j}\cdot \mathrm{KL}(h^G_{e_j,T_i}||\mathrm{HA}_{e_j})}\ . \tag{5.21}$$

$$FLR = \frac{\sum_{i=1}^{|\mathcal{T}|}\sum_{j=1}^{|E|}\mathbb{1}_{T_i,e_j}|LR_{e_j,T_i} > 1|}{\sum_{i=1}^{|\mathcal{T}|}\sum_{j=1}^{|E|}\mathbb{1}_{T_i,e_j}}\ , \tag{5.22}$$

where $LR_{e_j,T_i} = \frac{\prod_{k=1}^{|o|}(P_{\widehat{h}}(o_k))}{\prod_{k=1}^{|o|}(P_{\mathrm{HA}}(o_k))}$, $|o|$ is the total number of ground truth records, $P_{\widehat{h}}(o_k)$ and $P_{\mathrm{HA}}(o_k)$ are the probabilities of observing o_k from $\widehat{h}$ and HA.The higher the FLR value is, the better the method is.

5.2 Effectiveness evaluation

We set rm as 0.5 – 0.8 for XN and CD datasets as [1]. In HK, we do not remove any data and only evaluate *prediction* as the sparsity is already 90% (None for rm in Tables 1, 2).

Estimation: In Tables 1, 2, the MKLR values on *estimation* task in the XN and CD datasets increase—recall that a low MKLR value is better—as rm increases. The reason is that when more edges are removed, less information can be used when propagating the correlation among edges and time intervals. And the FLR values in the XN and CD datasets decrease—recall that a high FLR value is better—as rm increases. The reason is the same as for MKLR since fewer data provide less information. ConGC achieves the best performance on all datasets. For example, its average improvements of MKLR and FLR values over state-of-the-art model A-GCWC on *estimation* are 6% and 7%.

Prediction: In Tables 1, 2, ConGC beats other methods as well on the *prediction* task. And the average improvements of MKLR and FLR values over state-of-the-art model A-GCWC are 5% and 8%.

Moreover, ConGC is clearly better than A-GCWC when rm is large. For the largest rm 0.8 on XN and CD datasets, the average improvement of ConGC is 6.12% more accurate than A-GCWC. As for HK, its average improvement is 4% more accurate than A-GCWC.

5.3 Efficiency evaluation

We use GeForce GTX 1080 Ti 11 GB GPU for evaluation.

Efficiency comparison: Figure 5 (g) and (h) show the average training and testing time for a single instance (i.e., a weight matrix at one time interval for all edges). Firstly, RF, DSAE and CNN are faster than others since they only consider *spatial* data. Second, DCRNN, A-GCWC, GC (One basic variant ignores contexts) and ConGC have comparable performance, while ASTGCN, ST-ResNet are much slower. Note that ST-ResNet, ASTGCN, GC and ConGC consider more data, i.e., *spatial*, *temporal* and *periodic* data, while DCRNN only considers *spatial* and *temporal* data. It means that ConGC have comparable performance even considering more data than other methods.

Scalability w.r.t. the number of roads: We manually enlarge the dataset as [1], since large road networks with dense data are unavailable. The maximum number of edges that one GPU can process with a batch size 8 is 1600 for ConGC. We manually enlarge the road network of XN to 1536 for Figure 5 (a-b), and enlarge it to 1 million for Figure 5 (c-d), and measure the average running time for an instance (i.e., a weight matrix

Table 1: MKLR (**lower is better**) on three datasets. For each method, we report average results over 10 runs.

Datasets	Functionalities	rm	RF	CNN	DSAE	DCRNN	ASTGCN	ST-ResNet	AGCWC	GC	ConGC
XN	Estimation	0.5	0.91	0.49	0.37	1.07	0.47	0.72	0.22	0.21	**0.14**
		0.6	1.00	0.54	0.58	1.29	0.57	0.76	0.27	0.24	**0.23**
		0.7	0.95	0.54	1.13	1.16	0.52	0.60	0.39	0.37	**0.26**
		0.8	0.95	0.60	1.93	1.16	0.61	0.65	**0.59**	**0.59**	**0.59**
	Prediction	0.5	0.91	0.71	1.17	1.02	0.74	0.74	0.66	**0.65**	**0.65**
		0.6	1.04	0.71	1.30	1.25	0.74	0.73	0.66	**0.65**	**0.65**
		0.7	0.94	0.71	1.42	1.10	0.71	0.72	0.69	0.63	**0.62**
		0.8	0.95	0.71	1.63	1.07	0.70	0.75	0.69	0.62	**0.61**
CD	Estimation	0.5	0.98	0.70	0.74	0.87	0.77	0.62	0.53	0.44	**0.40**
		0.6	0.91	0.73	0.94	0.88	0.68	0.64	0.56	0.57	**0.54**
		0.7	0.99	0.77	1.19	0.89	0.70	**0.64**	0.72	0.65	**0.64**
		0.8	0.92	0.80	1.36	0.94	0.79	**0.77**	0.79	0.80	**0.77**
	Prediction	0.5	0.99	0.74	1.12	0.90	0.71	0.69	0.72	0.69	**0.68**
		0.6	0.91	0.74	1.18	0.91	0.71	0.70	0.73	0.70	**0.69**
		0.7	0.99	0.75	1.24	0.91	0.72	0.70	0.75	0.70	**0.69**
		0.8	0.93	0.76	1.30	0.92	0.72	0.70	0.76	0.71	**0.70**
HK	Prediction	None	0.92	0.82	1.14	1.27	0.88	0.77	0.80	0.80	**0.74**

Table 2: FLR (**higher is better**) on three datasets. For each method, we report average results over 10 runs.

Datasets	Functionalities	rm	RF	CNN	DSAE	DCRNN	ASTGCN	ST-ResNet	AGCWC	GC	ConGC
XN	Estimation	0.5	0.27	0.53	0.77	0.48	0.53	0.45	0.76	0.80	**0.90**
		0.6	0.31	0.53	0.74	0.40	0.51	0.43	0.76	0.78	**0.82**
		0.7	0.25	0.54	0.64	0.44	0.54	0.47	0.71	0.76	**0.82**
		0.8	0.21	0.49	0.47	0.43	0.50	0.46	0.47	0.52	**0.54**
	Prediction	0.5	0.27	0.43	0.42	0.50	0.41	0.42	0.44	0.51	**0.52**
		0.6	0.30	0.43	0.40	0.42	0.41	0.44	0.44	0.51	**0.52**
		0.7	0.25	0.40	0.38	0.44	0.39	0.41	0.36	0.49	**0.50**
		0.8	0.22	0.40	0.34	0.45	0.39	0.37	0.36	0.49	**0.50**
CD	Estimation	0.5	0.17	0.47	0.61	0.56	0.48	0.53	0.60	0.62	**0.67**
		0.6	0.15	0.46	0.52	0.56	0.50	0.52	0.57	0.56	**0.60**
		0.7	0.12	0.44	0.43	0.40	0.49	0.50	0.47	**0.51**	**0.51**
		0.8	0.09	0.42	0.35	0.36	0.44	0.47	0.43	0.46	**0.48**
	Prediction	0.5	0.17	0.44	0.40	0.51	0.49	0.54	0.49	0.54	**0.55**
		0.6	0.14	0.44	0.36	0.50	0.49	0.54	0.48	0.54	**0.55**
		0.7	0.12	0.43	0.33	0.49	0.48	**0.54**	0.47	**0.54**	**0.54**
		0.8	0.08	0.42	0.31	0.50	0.48	**0.53**	0.46	0.52	**0.53**
HK	Prediction	None	0.19	0.37	0.44	0.37	0.30	0.43	0.43	0.43	**0.45**

at one time interval for all edges). We follow the two settings in [1] and evaluate (1) the scalability of moderate road networks that fit into one GPU (Figure 5 (a-b)) and (2) the scalability of very large road networks which have to be partitioned into multiple small road networks that can be trained in sequence by batches in one GPU (Figure 5 (c-d)). We adapt methods with the state-of-the-art partitioning-based approach [18] to render very large road networks feasible. Figure 5 (a-d) shows that ConGC is scalable on moderate and vast road networks.

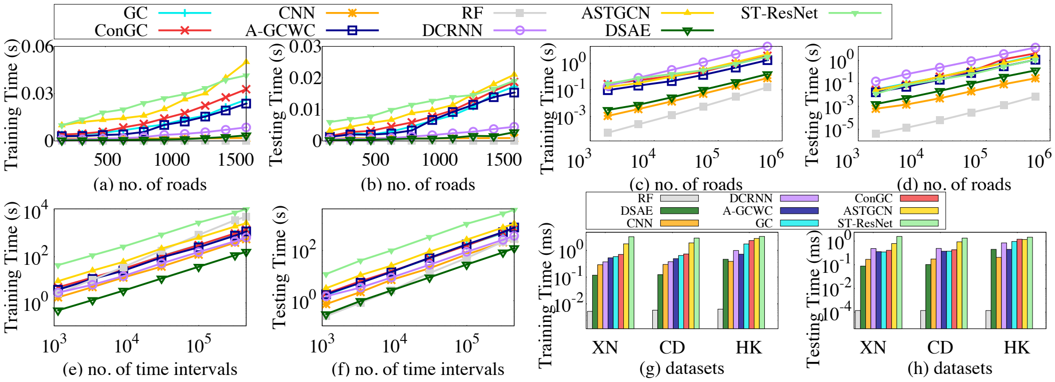
Scalability w.r.t. the number of time intervals: Similarly, we manually enlarge the dataset w.r.t. $|\mathcal{T}|$ from 1131 to 452,400, and measure the average training time for one epoch and the testing time for all testing data. Here, $|\mathcal{T}| = 452,400$ time intervals represent 12.9 years of data with 15-min intervals. The number of edges is set to 64. Figure 5 (e-f) shows that the ConGC is scalable on the time dimension.

6 Conclusions

We study the stochastic weight completion problem under data sparsity. We propose ConGC to utilize contextual graph for learning weights. The ConGC propagates correlations in spatial and temporal dimensions from edges with weights to edges with missing weights. Our evaluation results show that ConGC is more effective, and can scale to large road networks.

Acknowledgement

Reynold Cheng, Xiaolin Han, Chenhao Ma, and Xiaodong Li were supported by the University of Hong Kong (Projects 104005858, 10400599, KE award 2100048), the Innovation and Technology Commission of Hong Kong (ITF project MRP/029/18), and HKU-TCL Joint Research Center for Artificial Intelligence (Project no. 200009430). Tobias Grubenmann was supported by the Federal Ministry of Education and Research (BMBF), Germany, under Simple-



(a) no. of roads (b) no. of roads (c) no. of roads (d) no. of roads (e) no. of time intervals (f) no. of time intervals (g) datasets (h) datasets

Figure 5: Scalability on (a-b) moderate and (c-d) large road networks; (e-f) time dimension. (g-h) Efficiency.

ML (01IS18054), and the European Commission under PLATOON (872592) and Cleopatra (812997). Xiaodong Li is supported by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative.

References

[1] J. Hu, C. Guo, B. Yang, and C. S. Jensen, *Stochastic weight completion for road networks using graph convolutional networks*, IEEE ICDE, 2019.

[2] D. Deng, et al., *Latent space model for road networks to predict time-varying traffic*, SIGKDD, 2016.

[3] Y. Li, et al., *Diffusion convolutional recurrent neural network: Data-driven traffic forecasting*, ICLR, 2018.

[4] P. Newson, J. Krumm, *Hidden Markov map matching through noise and sparseness*, SIGSPATIAL, 2009.

[5] S. Guo, Y. Lin, N. Feng, C. Song, H. Wan, *Attention based spatial-temporal graph convolutional networks for traffic flow forecasting*, AAAI, 2019.

[6] P. Veličković, G. Cucurull, A. Casanova, et al., *Graph Attention Networks*, ICLR, 2018.

[7] S.A. Pedersen, B. Yang, C.S. Jensen, *Fast stochastic routing under time-varying uncertainty*, VLDBJ, 2019.

[8] B. Yang, J. Dai, C. Guo, et al., *PACE: a PAth-CEntric paradigm for stochastic path finding*, VLDBJ, 2018.

[9] S.A. Pedersen, .B Yang, C.S. Jensen, *Anytime Stochastic Routing with Hybrid Learning*, VLDB, 2020.

[10] C. Guo, B. Yang, J. Hu, C.S. Jensen, *Learning to route with sparse trajectory sets*, IEEE ICDE, 2018.

[11] Y. Wang, et al., *Real-time Traffic Pattern Analysis and Inference with Sparse Video Surveillance*, IJCAI, 2018.

[12] J. Shang, Y. Zheng, W. Tong, E. Chang, Y. Yu, *Inferring gas consumption and pollution emission of vehicles throughout a city*, SIGKDD, 2014.

[13] D. Woodard, G. Nogin, et al., *Predicting travel time reliability using mobile phone GPS data*, Transportation Research Part C: Emerging Technologies, 2017.

[14] S. Wang, M. Zhang, H. Miao, P.S. Yu, *MT-STNets: Multi-Task Spatial-Temporal Networks for Multi-Scale Traffic Prediction*, SIAM SDM, 2021.

[15] R. Dai, S. Xu and et al., *Hybrid Spatio-Temporal Graph Convolutional Network: Improving Traffic Prediction with Navigation Data*, SIGKDD, 2020.

[16] M. Defferrard, et al., *Convolutional neural networks on graphs with fast localized spectral filtering*, NIPS, 2016.

[17] A.A. Prakash, *Pruning algorithm for the least expected travel time path on stochastic and time-dependent networks*, Transportation Research Part B, 2018.

[18] W.L. Chiang, X. Liu, S. Si, Y. Li and et al., *Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks*, SIGKDD, 2019.

[19] C. Song, Y. Lin, S. Guo, H. Wan, *Spatial-Temporal Synchronous GCNs: A New Framework for Spatial-Temporal Network Data Forecasting*, AAAI, 2020.

[20] R. Huang, et al., *LSGCN: Long Short-Term Traffic Prediction with Graph Convolutional Networks*, IJCAI.

[21] X. Wang, et al., *Traffic Flow Prediction via Spatial Temporal Graph Neural Network*, TheWebConf, 2020.

[22] J. Zhang, et al., *Deep spatio-temporal residual networks for citywide crowd flows prediction*, AAAI, 2017.

[23] Y. Duan, Y. Lv and et al.,*An efficient realization of deep learning for traffic data imputation*, Transportation research part C: emerging technologies, 2016.

[24] C. Ma, et al., *Linc: a motif counting algorithm for uncertain graphs*, PVLDB, 2019.

[25] C. Ma, et al., *Efficient algorithms for densest subgraph discovery on large directed graphs*, SIGMOD, 2020.

[26] C. Ma, et al., *On Directed Densest Subgraph Discovery*, TODS, 2021.

[27] C. Ma, et al., *A Convex-Programming Approach for Efficient Directed Densest Subgraph Discovery*, SIGMOD, 2022.

[28] X. Han, et al., *Traffic incident detection: A trajectory-based approach*, ICDE, 2020.

[29] X. Han, *Traffic Incident Detection: A Deep Learning Framework*, MDM, 2019.

[30] X. Han, et al., *A Framework for Differentially-Private Knowledge Graph Embeddings*, Journal of Web Semantics, 2022.

[31] X. Li, et al., *M-Cypher: A GQL Framework Supporting Motifs*, CIKM, 2020.