Learning Spatiotemporal Latent Factors of Traffic via Regularized Tensor Factorization: Imputing Missing Values and Forecasting

Abdelkader Baggag, Sofiane Abbar, Ankit Sharma, Tahar Zanouda, Abdulaziz Al-Homaid, Abhiraj Mohan, and Jaideep Srivastava, *Fellow, IEEE*

Abstract—Intelligent transportation systems are a key component in smart cities, and the estimation and prediction of the spatiotemporal traffic state is critical to capture the dynamics of traffic congestion, i.e., its generation, propagation and mitigation, in order to increase operational efficiency and improve livability within smart cities. And while spatiotemporal data related to traffic is becoming common place due to the wide availability of cheap sensors and the rapid deployment of IoT platforms, the data still suffer some challenges related to sparsity, incompleteness, and noise which makes the traffic analytics difficult. In this article, we investigate the problem of missing data or noisy information in the context of real-time monitoring and forecasting of traffic congestion for road networks in a city. The road network is represented as a directed graph in which nodes are junctions (intersections) and edges are road segments. We assume that the city has deployed high-fidelity sensors for speed reading in a subset of edges; and the objective is to infer the speed readings for the remaining edges in the network; and to estimate the missing values in the segments for which sensors have stopped generating data due to technical problems (e.g., battery, network, etc.). We propose a tensor representation for the series of road network snapshots, and develop a regularized factorization method to estimate the missing values, while learning the latent factors of the network. The regularizer, which incorporates spatial properties of the road network, improves the quality of the results. The learned factors, with a graph-based temporal dependency, are then used in an autoregressive algorithm to predict the future state of the road network with a large horizon. Extensive numerical experiments with real traffic data from the cities of Doha (Qatar) and Aarhus (Denmark) demonstrate that the proposed approach is appropriate for imputing the missing data and predicting the traffic state. It is accurate and efficient and can easily be applied to other traffic datasets.

Index Terms—Tensor decomposition, regularization, traffic monitoring, traffic forecasting

1 Introduction

Rapid urbanization, GDP growth, decline in fuel prices, and increase in car ownership are all factors that contribute directly or indirectly in creating and/or worsening road congestion. Most of the cities in the world are regularly monitoring yearly traffic congestion-related KPIs that help them evaluate the road infrastructure. According to [16], the economic cost of congestion in 2016 in Qatar is estimated to be between US 1.53$B and US 1.80$B which translates to a loss of about 0.9-1.0 percent of the GDP. Thus, it is extremely important for cities to deploy required systems and applications that provide access to real-time congestion information. For city planners and operators, knowing what is going to happen in their road networks is as important as knowing the real-time situation. For example, predicted traffic congestion information is essential for anticipating the implementation of automated actions to avoid heavy congestion in strategic locations, this can help urban planners to fine-tune signal timings or strategically allocate police patrol units to regulate the traffic.

From the user perspective, it is essential to provide the fastest path from a source to a destination with an accurate estimation of the Estimated Time of Arrival (ETA). These features can only be offered if the system has the capability to forecast efficiently and effectively the traffic congestion in addition to the real-time monitoring.

With the rise of IoT technologies, smart cities became a testbed for technologies that capture spatio-temporal real-time data on citizens' mobility. This later has been enabled mainly by the wide availability of cheap sensors (e.g., closed-loop detectors, Bluetooth sensors). However, the data still suffer some major challenges related to sparsity, incompleteness, and noise which makes the traffic analytics difficult. It is for instance reported in many studies that the percentage of missing values in real traffic monitoring scenarios can be as high as 90 percent [21], [23], [24]. The obvious reason for inherent sparsity problem we observe in traffic data is that not all road segments are equipped with sensors. And even when a road segment has a sensor, it is often the case that it temporarily stops emitting due to a variety of failures (e.g., networks,

- *A. Baggag, S. Abbar, and A. Al-Homaid are with the Qatar Computing Research Institute, Hamad Bin Khalifa University, Doha, Qatar. E-mail: {abaggag, sabbar, abalhomaid}@hbku.edu.qa.*
- *A. Sharma, A. Mohan, and J. Srivastava are with the Department of Computer Science at the University of Minnesota, Minneapolis, MN 55455 USA. E-mail: {sharm170, mohan056, srivasta}@umn.edu.*
- *T. Zanouda is with Ericsson, Sweden. E-mail: zanouda@gmail.com.*

Manuscript received 11 Jan. 2018; revised 30 Sept. 2019; accepted 11 Nov. 2019. Date of publication 28 Nov. 2019; date of current version 29 Apr. 2021. (Corresponding author: Abdelkader Baggag.)
Recommended for acceptance by J. Ye.
Digital Object Identifier no. 10.1109/TKDE.2019.2954868

batteries, and low recall.) This problem makes traditional approaches based on time-series for traffic forecasting obsolete [3], [24], [27]; and even more robust decomposition-based methods such as matrix factorization–which aim to project objects into a lower-dimensional latent space–have been shown to be *NP-hard* in the presence of missing values, see e.g., [7]. In fact, handling missing values in decompositions is often via *Imputation* or *Marginalization.* In the former approach, missing values are estimated iteratively using the Expectation Maximization algorithm. This results in an easy implementation with very little changes into the model [15]. In the later approach, i.e., *Marginalization or Weighted Regression*, the missing values are simply ignored during the optimization process [22].

However, given the specific nature of traffic data such as the high temporal and spatial correlation that can be leveraged to pacify the data sparsity problem, and the periodicity in data that can be exploited for the problem of multi-step ahead forecasting, it is important to design new analytical frameworks that naturally account for the above-mentioned properties.

Motivated by these challenges, we propose a temporal regularized tensor factorization framework (TRTF) suitable for high dimensional time-series analysis. TRTF models traffic data as follows. First, the road network of a city is modeled as a directed graph in which nodes are junctions (e.g., intersections, Y-merging, round-abouts, etc.) and edges are road segments linking junctions. The adjacency matrix $\mathbf{A}_{\text{adj}}$ of size $N \times N$, where N is the number of nodes of the road network, is assumed to remain unchanged over time. Second, speed readings associated with edges in the graph are modeled into a three-way tensor $\mathcal{G}$ of size $(N \times N \times T)$, where T is the number of time points. In other words, the tensor $\mathcal{G}$ is a time-dependent sequence of T successive $\mathbf{G}_t$ matrices such that $\mathbf{G}_t^{(i,j)} = 0$ if $\mathbf{A}_{\text{adj}}^{(i,j)} = 0$ i.e., there no road segment between junctions i and j, or if $\mathbf{A}_{\text{adj}}^{(i,j)} = 1$ but there is no speed reading for road-segment $[i, j]$ at time t; $\mathbf{G}_t^{(i,j)} > 0$ otherwise. The objective then is to decompose the tensor $\mathcal{G}$ into three factor matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ minimizing the following error:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \mathcal{E}(\mathcal{G} \parallel \mathbf{A}, \mathbf{B}, \mathbf{C}) + \mathcal{R}_s(\mathbf{A}, \mathbf{B}) + \lambda_t \mathcal{R}_t(\mathbf{C}), \tag{1}$$

where the first two factors $\mathbf{A}$ and $\mathbf{B}$ capture the spatial constraints of the road network, whereas $\mathbf{C}$ captures the structure of the temporal dependencies among the temporal embeddings. The decomposition problem could be solved by an alternating minimization process over $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. It is a regularized minimization problem to avoid overfitting and to incorporate the spatial dependency through $\mathcal{R}_s$ and the temporal dependency through $\mathcal{R}_t$.

One of our main contributions in this paper is the adaptation of a novel autoregressive temporal regularizer $\mathcal{R}_t(\mathbf{C})$ for tensor decompositions. This modeling is adapted from the regularized matrix representation introduced in [25]. In that work, traffic data is represented as a matrix in which rows are road segments and columns are time points. That is, each row is a time-series of speed readings of a particular road segment. The proposed Temporal Regularized Tensor Factorization (TRTF) framework allows the learning of temporal dependencies from the data and enables multi-step ahead forecasting using tensor decomposition. This is done by learning the temporal dependencies between the latent dimensions by incorporating an autoregressive temporal regularizer into the factorization process.

Latent space modeling has attracted a lot of attention lately. Deng *et al.* [4] proposed to use matrix factorization with spatial and temporal regularizers to deal with the two problems of missing values and forecasting. Although the authors highlight the difference between latent models for social network and that for traffic prediction, they eventually propose a model similar to latent social network models. Moreover, they bring in the temporal aspect by learning a fixed transition matrix between consecutive snapshots.

One solution to cope with this problem is to further exploit the summary/stacked transition model technique of Rossi *et al.* [17] and possibly make the model learn longer temporal patterns spanning multiple snapshots. However, in our case, we decided to rather take a tensor based approach which elegantly and naturally captures the temporal dependencies in the latent space. Another interesting approach to capture time dependencies is that of Wang *et al.* in which they propose a tensor based model for segment as well as path cost estimation using GPS data [23]. However, their approach has few short-comings. First, they aggregate the historical data per time-slot for the entire history and therefore, possibly losing valuable information. Second, in order to build real-time scalable solution they only consider the very recent time-slots. But this along with the aggregation make this model not suitable for forecasting where we would like the periodic patterns in data be preserved. Third, their approach uses a lot of auxiliary information which may not be available for all data-sets. Moreover, unlike the holistic manifold smoothing used in [4], Wang *et al.* manually construct the spatial features which constitutes a serious limitation for a data-driven approach. Fourth, their approach relies on Tucker decomposition [11] which has been advised not to be used for temporal problems as it suffers from rotational freedom [5].

In this study, we use CP decomposition, which is more appropriate when time order preserving is required, see e.g., [1]. We run extensive experimentation on two real traffic datasets, from two different cities, Aarhus in Denmark and Doha in Qatar, showing different sparsity distributions. In Aarhus, the percentage of missing values is only about 4 percent, whereas Doha dataset suffers from 85 percent missing values. This, in itself, constitutes a good test-bed to evaluate against several recent algorithms designed for the same purpose such as LSM-RN [4], TRMF [25], and CP-WOPT [1].

The main contributions of this paper are:

1) We propose a novel temporal regularized tensor factorization framework (TRTF) for high-dimensional traffic data. TRTF provides a principled approach to account for both the spatial structure and the temporal dependencies.
2) We introduce a novel data-driven graph-based autoregressive model, where the weights are learned from the data. Hence, the regularizer can account for both positive and negative correlations.

3) We show that incorporating temporal embeddings into `CP-WOPT` [1] leads to accurate multi-step forecasting, compared to state of the art matrix factorization based methods.
4) We conduct extensive experiments on real traffic congestion datasets from two different cities and show the superiority of `TRTF` for both tasks of missing value completion and multi-step forecasting under different experimental settings. For instance, `TRTF` outperforms `LSM-RN` by 24 percent and `TRMF` by 29 percent.

The results show the superiority of tensor based methods in general and our framework in particular for both tasks of missing value completion and multi-step ahead forecasting under different experimental settings. Through our extensive experiments on real datasets, we show that our method outperforms `LSM-RN` with about 24 percent and `TRMF` with 29 percent.

The remainder of this paper is organized as follows. We describe our problem in Section 2 before we discuss the literature review and its limitations in Section 3. In Section 4 we introduce our temporal regularized tensor factorization framework and describe the auto regressive extension allowing for multi-steps-ahead forecasting in Section 5. We demonstrate the suitability and superiority of our framework through an extensive set of experiments on two different real traffic datasets in Section 6. We finally conclude the paper in Section 7.

2 Problem Formulation

We represent the road network of a city as a directed graph consisting of N nodes and M edges. Nodes represent junctions and edges represent road segments connecting two junctions. We assume the existence of $K \ll M$ physical sensors located on K different road segments (edges) that continuously generate speed readings at a particular rate, e.g., every 1 minute. Sensors can occasionally fail to generate readings, yielding missing values. This traffic data is used to build a three-way tensor $\mathcal{G}$ of size $(N \times N \times T)$, where T is the number of time points at which speed data has been generated (from time 1 to time T). Each slice $\mathbf{G}_t$ in $\mathcal{G}$ is a $(N \times N)$ matrix of which cells $\mathbf{G}_t^{(i,j)}$ report speed values observed in segment $[i,j]$ at time t. $\mathbf{G}_t^{(i,j)}$ takes 0 if the edge $[i,j]$ does not exist or if its corresponding sensor fail to generate a speed reading at time t, see Fig. 1. The distinction between zeros of non existing edges and those of missing values is made by looking at the adjacency matrix of the road network.

Our problem is therefore:

1) to complete the (data) tensor $\mathcal{G}$ by inferring all its missing values due to absent or malfunctioning sensors; and
2) to forecast the future tensor $\mathcal{G}_{\text{new}}$ that extends the initial tensor $\mathcal{G}$ from time T to $T+h$, where h is the desired horizon.

3 Related Work

In this section, we present the literature review of previous works related to ours. We mainly focus on the two categories of work related to *traffic prediction* and the use of tensors in the new discipline of *urban computing*.

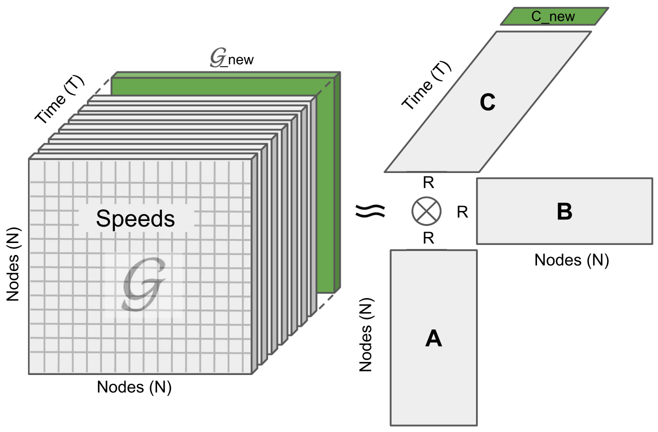


Fig. 1. Tensor factorization model for multiple temporal snapshots of speed matrices and illustration of the learning process for predictions. First the tensor $\mathcal{G}$ is decomposed into three factor matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$, then $\mathbf{C}_{\text{new}}$ is obtained from $\mathbf{C}$ via an autoregressive model in which the weights are learned, and finally the predicted $\mathcal{G}_{\text{new}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}_{\text{new}}]\!]$.

3.1 Traffic Prediction

Research in traffic prediction can be classified on the basis of the target being estimated or the data being utilized. Mainly there are two traveling costs being estimated: (1) Segment cost and (2) Path cost. Moreover, research is fragmented on the type of data that is being employed as input for the various cost estimations. Here, we have two main categories: (1) GPS-based data and (2) sensor-based data. Sensor only captures data for the road segment it is tracking but the data collection is more fine grained and continuous. GPS-data, unlike sensor-data, does not provide continuous data streams, but instead gives a more global view as the entire trip of a vehicle is recorded. Irrespective of the cost being estimated and the data being utilized, sparsity of the data whether missing sensors or un-traveled segments, is a central concern. The existence of missing data will reduce the performance sharply [2], [3], [10]. Therefore, models that treat each segment data as independent time-series fail . However, a promising feature of traffic data is that it is highly correlated both temporally as well as spatially. Therefore, several latent as well as correlated time-series based models have been proposed that exploit this nature of the traffic data. Recently, Deng *et al.* [4] proposed a latent spatio-temporal model based on traffic snapshots of the road crossing network. Inspired from social network models [9], [17] where different actors have roles that evolve over time, they similarly hypothesize that various road crossings have certain latent traffic roles which change over time. They treat the network partitions in every snapshot as roles and also learn how role transitions occur between consecutive snapshots in the form of a role transition matrix.

3.2 Tensors in Urban Computing

Given the spatial-temporal dynamics in cities, tensorial analysis attracted more attention in the field of urban informatics. Zheng *et al.* highlighted the importance of using tensors in this area in their insightful review [28] on the recent research and challenges at the convergence of city science and urban computing using human mobility data. Others used tensorial formulation to address different problems, varying from noise pollution prediction [29], and gas station recommendation

[18], to modeling urban refueling events [26] by adapting Tucker tensor decomposition to capture contextual features.

Moreover, researchers used tensor formulation to study and characterize human mobility in the city. Fan *et al.* [6] used Non-negative Tensor Factorization (NTF) with spatial regularization to model people's flow in the city. The algorithm decomposes people flow tensor into basic life pattern tensors, in order to capture real-world human mobility. The tool was used to quantify the fluctuation of urban mobility before and after the Great East Japan Earthquake in 2011. Moreover, Tan *et al.* [21] showed the efficiency of using tensor-based methods, and the authors integrated Tucker decomposition and EM algorithm to infer the missing values. Their experiments demonstrated superior performance than the state-of-the-art imputation approaches even when the missing ratio is up to 90 percent. Another research study by Takeuchi *et al.* [20] showed the use of a Non-negative Multiple Tensor Factorization algorithm to capture and reveal spatio-temporal patterns of peoples' daily routines such as leisure, drinking, and shopping activity from online review data sets. Similarly, Han *et al.* [8] proposed the use of non-negative tensor decomposition to identify spatial-temporal patterns of traffic. The study has shown the usefulness of tensorial decomposition techniques for multivariate data sequences.

4 Proposed Methodology

Let $\mathcal{G}$ be a three-way tensor of size $N \times N \times T$, and assume its rank is R. With perfect data, i.e., data with no missing values, the tensorial decomposition of $\mathcal{G}$ is defined by factor matrices $\mathbf{A} \in \mathbb{R}^{N\times R}$, $\mathbf{B} \in \mathbb{R}^{N\times R}$ and $\mathbf{C} \in \mathbb{R}^{T\times R}$ and denoted as $\mathcal{G} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$, such that

$$\mathcal{G}_{ijt} = \sum_{\ell=1}^{R} \mathbf{A}_{i\ell}\mathbf{B}_{j\ell}\mathbf{C}_{t\ell}, \tag{2}$$

for all $1 \le i, j \le N$ and $1 \le t \le T$, see Fig. 1.

This special tensorial decomposition has been discovered and named by many researchers independently, such as CANDECOMP (canonical decomposition) and PARAFAC (parallel factors), which preserves the uniqueness under some mild conditions [14].

Remark 1. The rank R of the tensor $\mathcal{G}$ is defined as the minimal number of rank-1 tensors whose linear combination yields $\mathcal{G}$, i.e.,

$$\mathcal{G} = \sum_{\ell=1}^{R} \mathbf{a}_\ell \otimes \mathbf{b}_\ell \otimes \mathbf{c}_\ell, \tag{3}$$

where $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ are column vectors of the factor matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$, respectively. Here, $\otimes$ denotes the standard outer product. Contrary to the case of matrices, the rank of a tensor is presently not well understood. It is known that the problem of computing the rank of a tensor is NP-hard.

For traffic data obtained from stationary and mobile sensors, missing values are unavoidable due to failures such as detector malfunction. Therefore, the true $\mathcal{G}$ is not observable and we cannot expect equality in (2). Instead, the CP decomposition should minimize the error function as follows:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \mathcal{E}(\mathcal{G} \parallel \mathbf{A}, \mathbf{B}, \mathbf{C}) = \sum_{i,j=1}^{N}\sum_{t=1}^{T}\left(\mathcal{G}_{ijt} - \sum_{\ell=1}^{R}\mathbf{A}_{i\ell}\mathbf{B}_{j\ell}\mathbf{C}_{t\ell}\right)^2.$$

Unlike the singular value decomposition (SVD) for matrices, PARAFAC (usually) does not impose any orthogonality constraints. The two main competitors of PARAFAC are the Tucker3 method [13], and simply the unfolding of the tensor into a matrix and then performing standard two-way methods such as PCA. The alternating least-squares framework is used to solve the CP approximation problem. It proceeds by solving a sequence of structured linear least squares problems.

Remark 2. Modern applications, such Intelligent Transportation Systems, generate massive amounts of data with multiple aspects such as sparsity and high dimensionalities; and tensors provide a natural representation for such data. Consequently, tensor decompositions have become important tools for analysis. Given a high-order large-scale tensor, how can we decompose it into latent factors in a scalable manner? In a recent work [19], the authors propose Coordinate Descent for Tensor Factorization, which is more memory efficient and applicable to more loss functions; and Subset Alternating Least Square, which converges faster to a better solution. These distributed tensor factorization methods are scalable with all aspects of data. Moreover, there have been attempts to improve the accuracy and efficiency of the decomposition by encoding prior knowledge of the application, and using additional contextual information (such as time and location), as a regularization term in the objective function.

Our approach to deal with missing values is to use a weighted version of the error function to ignore missing data and model only the known entries.

Definition 1. *We define the nonnegative binary weight index tensor $\mathcal{W}$, of the same size as $\mathcal{G}$, as a 0–1 tensor, which indicates whether entries of $\mathcal{G}$ are missing or not, i.e.,*

$$\mathcal{W}_{ijt} = \begin{cases} 1 & \text{if} \quad \mathcal{G}_{ijt} \quad \text{is known}\ , \\ 0 & \text{if} \quad \mathcal{G}_{ijt} \quad \text{is missing (or unknown)}\ . \end{cases} \tag{4}$$

Therefore the weighted version of the error is

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \mathcal{E}_{\mathcal{W}}(\mathcal{G} \parallel \mathbf{A}, \mathbf{B}, \mathbf{C}) = \|\mathcal{W} \odot (\mathcal{G} - [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!])\|^2,$$

where $\odot$ represents the Hadamard product, i.e., point-wise multiplication. By minimizing the objective function, we can get optimized $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$. Now, we can recover the missing values in $\mathcal{G}$ by

$$\mathcal{G}_{\text{recovered}} \leftarrow \hat{\mathcal{G}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]. \tag{5}$$

Remark 3. If we restrict the core-array of the Tucker model to be diagonal with values one in the diagonal elements, we arrive at the CP model which has (in general) a unique minimizer of the cost function (up to scale and permutation of the components), in contrast to the Tucker model.

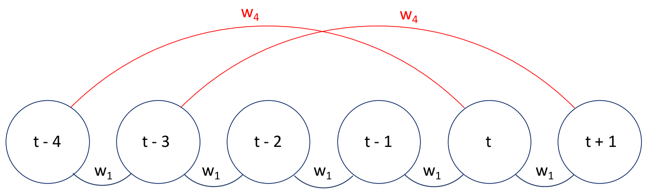


Fig. 2. Illustration of the graph-based regularization of temporal dependencies. Each node represents a column vector of $\bar{\mathbf{C}} = \mathbf{C}^T$, and an edge between two nodes represents the temporal dependency. The weights of the edges will be learned. In this example, $\mathcal{L} = \{1, 4\}$.

We can then solve

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \mathcal{E}_{\mathcal{W}}(\mathcal{G} \parallel \mathbf{A}, \mathbf{B}, \mathbf{C}) + \mathcal{R}_s(\mathbf{A}, \mathbf{B}) + \mathcal{R}_t(\mathbf{C}), \tag{6}$$

where the spatial regularizer is

$$\mathcal{R}_s(\mathbf{A}, \mathbf{B}) = \alpha_a \mathrm{Trace}(\mathbf{A}^T \mathbf{L}\ \mathbf{A}) + \frac{\beta_a}{2} \|\mathbf{A}\|_F^2 + \alpha_b \mathrm{Trace}(\mathbf{B}^T \mathbf{L}\ \mathbf{B}) + \frac{\beta_b}{2} \|\mathbf{B}\|_F^2. \tag{7}$$

Here $\mathbf{L} \in \mathbb{R}^{N\times N}$ is the graph Laplacian of the road network. This will allow embedding the spatial structure of the road network into the formulation. It is clear that the choice of the Frobenius norm only is not appropriate, as it does not take into account the dependencies among the columns of the factor matrices. Hence, it is important to impose prior knowledge of the application, such as spatial dependencies of the road network, into the learning algorithm. The Frobenius norm or Tikhonov-type regularization makes the learning algorithm numerically stable, since the sizes of the elements of the factor matrices become bounded. The penalty parameters are $\alpha_{a,b}$ and $\beta_{a,b}$. Therefore, the matrices $\mathbf{A}$ and $\mathbf{B}$ may be considered as representing the spatial latent factors, whereas the matrix $\mathbf{C}$ corresponds to the latent temporal embedding.

As for the temporal regularizer $\mathcal{R}_t(\mathbf{C})$, one of the basic and common ideas is to do a graph-based regularization for temporal dependencies, i.e., to incorporate general dependencies among temporal variables and to make the latent representations of the two entities as close as possible if there exists a relationship between them, i.e., an edge in the temporal graph, see Fig. 2. Assume that there exists a temporal graph $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$, whose adjacency matrix $\mathbf{W}_{\mathcal{G}_c} \in \mathbb{R}^{T\times T}$ encodes the relationships between the T columns of $\bar{\mathbf{C}} = \mathbf{C}^T$.

Therefore, two columns of $\bar{\mathbf{C}}$, e.g., $\bar{\mathbf{c}}_s$ and $\bar{\mathbf{c}}_t$ of size R, which are connected by an edge in this temporal graph, are said to be "close" to each other in terms of the euclidean distance, and hence temporally dependent, see Fig. 3.

Remark 4. We use $\bar{\mathbf{C}} = \mathbf{C}^T$ for clarity purposes. This does not affect the mathematical formalism. In so doing, a row in $\bar{\mathbf{C}}$, i.e., $\mathbf{c}_i^T$ for $i = 1, \ldots, R$, represents a time-series.

In the context of graph-based embedding, temporal regularization is achieved by including the following regularizer as part of the objective function

$$\mathcal{R}_c(\mathbf{C}) = \frac{\alpha_c}{2} \sum_{s\sim t} w_{st} \|\bar{\mathbf{c}}_s - \bar{\mathbf{c}}_t\|^2 + \frac{\eta}{2} \|\mathbf{C}\|_F^2, = \alpha_c \mathrm{Trace}(\mathbf{C}^T \mathbf{Lap}_{\mathcal{G}_c} \mathbf{C}) + \frac{\eta}{2} \|\mathbf{C}\|_F^2, \tag{8}$$

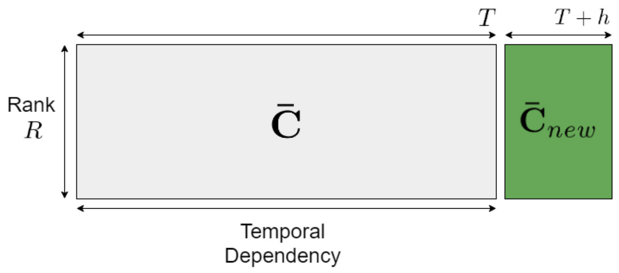


Fig. 3. The matrix $\bar{\mathbf{C}}$ captures the latent temporal variables. Each row represents a time-series. The forecasted $\bar{\mathbf{C}}_{\mathrm{new}}$ is obtained via an AR algorithm using a temporal graph where the weights are learned.

where $s \sim t$ denotes an edge between the sth node and the tth node, whose weight is w_{st}, and the second term is used to guarantee a strong convexity. This regularizer makes the temporal variable $\mathbf{C}$ to be faithful to the underlying temporal graph structure. The matrix $\mathbf{Lap}_{\mathcal{G}_c} \in \mathbb{R}^{T\times T}$ is the graph Laplacian associated with $\mathcal{G}_c$ and is defined as

$$\mathbf{Lap}_{\mathcal{G}_c} = \mathrm{diag}(\mathbf{W}_{\mathcal{G}_c} \mathbb{1}) - \mathbf{W}_{\mathcal{G}_c}, \tag{9}$$

where $\mathbb{1}$ is the vector of all ones of size T.

Lemma 1. *The objective function in (8) is convex.*

To apply graph-based regularizers to temporal dependencies, we need to specify the repeating dependency pattern by a lag set $\mathcal{L}$ and a weight vector $\mathbf{w}$ such that all the edges $s \sim t$ of distance ℓ, i.e., $|s - t| = \ell$, share the same weight w_ℓ. Therefore, given $\mathcal{L}$ and $\mathbf{w}$, the temporal regularizer (8) becomes

$$\mathcal{R}_c(\mathbf{C}) = \frac{\alpha_c}{2} \sum_{\ell\in\mathcal{L}} \sum_{\substack{t \\ t>l}} w_\ell \|\bar{\mathbf{c}}_t - \bar{\mathbf{c}}_{t-\ell}\|^2 + \frac{\eta}{2} \|\mathbf{C}\|_F^2. \tag{10}$$

While this graph-based regularization approach is intuitive, the explicit temporal dependency structure is usually not readily available. Therefore, one could try to learn this structure, i.e., the unknown weights w_l. This leads to the following optimization problem

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{w}\geq\mathbf{0}} \mathcal{E}_{\mathcal{W}}(\mathcal{G} \parallel \mathbf{A}, \mathbf{B}, \mathbf{C}) + \mathcal{R}_s(\mathbf{A}, \mathbf{B}) + \frac{\alpha_c}{2} \sum_{\ell\in\mathcal{L}} \sum_{\substack{t \\ t>\ell}} w_\ell \|\bar{\mathbf{c}}_t - \bar{\mathbf{c}}_{t-\ell}\|^2 + \frac{\eta}{2} \|\mathbf{C}\|_F^2. \tag{11}$$

While this formulation is common and widely used, it does have some issues. First, the method does not handle cases where negative correlation dependencies exist between two time points. Second, the method assumes that the structure of the temporal dependency is available, which is not the case in real-life applications. Indeed, most often, the explicit temporal dependency needs to be inferred. Moreover, a simple derivation of Equation (11) with respect to $\mathbf{w}$ shows that the above optimization yields the trivial all-zero solution for $\mathbf{w}$. Even, by adding an additional constraint, such as a simplex constraint on $\mathbf{w}$, there is a trivial solution. Therefore, learning the weights for the temporal dependencies is a challenging problem, and cannot be obtained simply by a mere addition of a regularizer into the error minimization.

Therefore, we use well-studied time-series models to describe temporal dependencies among the columns $\{\bar{\mathbf{c}}_t\}$ explicitly. Such models are of the form, see e.g., [25]

$$\bar{\mathbf{c}}_t = \mathcal{M}_{\boldsymbol{\Theta}}(\{\bar{\mathbf{c}}_t : \ell \in \mathcal{L}\}) + \boldsymbol{\varepsilon}_t, \tag{12}$$

where $\boldsymbol{\varepsilon}_t$ is a noise vector, assumed to be Gaussian here, and $\mathcal{M}_{\boldsymbol{\Theta}}$ is the time-series model which is parameterized by the set $\mathcal{L}$ containing the lag indices ℓ, denoting a dependency between time-points $\bar{\mathbf{c}}_t$ and $\bar{\mathbf{c}}_{t-\ell}$, and $\boldsymbol{\Theta}$ which captures the weight information of temporal dependencies, similar to a transition matrix in autoregressive models.

To incorporate the temporal dependencies in (6), we use a new (temporal) regularizer $\mathcal{T}_{\mathcal{M}}(\bar{\mathbf{C}} \mid \boldsymbol{\Theta})$ which encourages the structure induced by the time-series model $\mathcal{M}_{\boldsymbol{\Theta}}$. This regularizer is obtained from the general problem of maximizing the likelihood of parameters, given the data. Therefore, to estimate its parameters, we run a grid search over possible values to find the set of values for which the observed sample was most likely. That is, we find the set of parameter values that, given the model $\mathcal{M}_{\boldsymbol{\Theta}}$, were most likely to have given us the observed data. In other words, we want to know the distribution of the unknown parameters conditional on the observed data, i.e., $p(\text{Model} \mid \text{Data})$, which is known as the inverse probability problem. This cannot be calculated, so we go around it through the concept of the likelihood. And to get a convex function, we use the negative log likelihood, i.e.,

$$\mathcal{T}_{\mathcal{M}}(\bar{\mathbf{C}} \mid \boldsymbol{\Theta}) = -\log \mathbb{P}(\bar{\mathbf{c}}_1, \bar{\mathbf{c}}_2, \ldots, \bar{\mathbf{c}}_T \mid \boldsymbol{\Theta}). \tag{13}$$

The question is how can we estimate the parameters when we cannot maximize the likelihood analytically? To do so, we need to (1) be able to evaluate the likelihood function for a given set of parameters; and (2) find a way to evaluate a sequence of likelihoods conditional on difference parameter vectors so that we can feel confident that we have found the parameter vector that maximizes the likelihood. In this case, Grid Search, i.e., to divide the range of parameters into a grid and evaluate all possible combinations, is a method which guarantees finding the global optimum. With a fine enough grid, Grid Search always finds the global optimum, if the parameter space is bounded. However, it may be computationally infeasible for models with large number of parameters. Among the methods to solve this issue, one can cite the steepest ascent method, which is feasible for models with a large number of parameters; but can be hard to calibrate even for simple models to achieve the right rate of convergence. Another method is Newton-Raphson, which is similar to steepest ascent, but also computes the step-size which depends on the second derivative. It usually converges faster than steepest ascent, but it requires concavity, and so it is less robust when the shape of the likelihood function is unknown.

When $\boldsymbol{\Theta}$ is known, we can use $\mathcal{R}_c(\mathbf{C}) = \mathcal{T}_{\mathcal{M}}(\bar{\mathbf{C}} \mid \boldsymbol{\Theta})$ to encourage $\{\bar{\mathbf{c}}_t\}$ to follow the temporal dependency induced by $\mathcal{M}_{\boldsymbol{\Theta}}$; and when $\boldsymbol{\Theta}$ is unknown, we learn it by treating it as another set of variables and introduce another regularizer $\mathcal{R}_\theta(\boldsymbol{\Theta})$ into (6), that is

$$\begin{aligned}\min_{\mathbf{A},\mathbf{B},\mathbf{C},\boldsymbol{\Theta}} \; & \mathcal{E}_{\mathcal{W}}(\mathcal{G} \parallel \mathbf{A},\mathbf{B},\mathbf{C}) + \mathcal{R}_s(\mathbf{A},\mathbf{B}) + \lambda_c \mathcal{T}_{\mathcal{M}}(\bar{\mathbf{C}} \mid \boldsymbol{\Theta}) \\ & + \lambda_\theta \mathcal{R}_\theta(\boldsymbol{\Theta}),\end{aligned} \tag{14}$$

which can be solved by an alternating minimization procedure over $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and $\boldsymbol{\Theta}$.

Unlike the case of using directly a graph-based regularizer to embed the temporal dependencies, as in Equation (11), which leads to trivial solutions for the weights, the formulation (14) avoids this issue. When $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are fixed, the optimization problem is reduced to

$$\min_{\boldsymbol{\Theta}} \;\; \lambda_c \mathcal{T}_{\mathcal{M}}(\bar{\mathbf{C}} \mid \boldsymbol{\Theta}) \; + \; \lambda_\theta \mathcal{R}_\theta(\boldsymbol{\Theta}), \tag{15}$$

and therefore the set of parameters $\boldsymbol{\Theta}$ can be learned automatically from the data. This is a data-driven approach to learn the temporal dependency, and obtain non-trivial $\boldsymbol{\Theta}$, as shown in [25].

We show in what follows how the general temporal regularizer $\mathcal{T}_{\mathcal{M}}(\bar{\mathbf{C}} \mid \boldsymbol{\Theta})$, which incorporates dependencies specified by the time-series model $\mathcal{M}_{\boldsymbol{\Theta}}$, could be instantiated to the more specific case of an autoregressive (AR) model, which is parametrized by a lag set $\mathcal{L}$ and the weights $\mathcal{W} = \{\mathbf{W}^{(\ell)} \in \mathbb{R}^{R \times R} \;:\; \ell \in \mathcal{L}\}$. Assume that $\bar{\mathbf{c}}_t \in \mathbb{R}^R$ is a noisy linear combination of some previous points, i.e.,

$$\bar{\mathbf{c}}_t = \sum_{\ell \in \mathcal{L}} \mathbf{W}^{(\ell)} \bar{\mathbf{c}}_{t-\ell} + \boldsymbol{\varepsilon}_t, \tag{16}$$

where $\boldsymbol{\varepsilon}_t$ is assumed to be a Gaussian noise, see Equation (12). Here, the main challenge is to learn the weight matrix $\mathbf{W}^{(\ell)}$ associated with each $\ell \in \mathcal{L}$. To avoid overfitting, $\mathbf{W}^{(\ell)}$ is assumed to be a diagonal matrix. Algorithm 1 details the procedure.

Algorithm 1. LearnWeight

```
1: $\bar{\mathbf{C}} \leftarrow \texttt{Transpose}(\mathbf{C})$                    ▷ $\bar{\mathbf{C}} \in \mathbb{R}^{R\times T}$
2: procedure $[\mathbf{W}] = \text{LearnWeight}(\bar{\mathbf{C}}, \mathcal{L})$
3:     $T = \text{size}(\bar{\mathbf{C}}, 2)$
4:     $m = \max\{\mathcal{L}\}$
5:     $\texttt{idx} = (m+1) : T$                                        ▷ interval
6:     $\mathbf{W} = [\;\,]$                     ▷ Initialize the matrix $\mathbf{W}$ to zeros
7:     for $r \leftarrow 1, R$ do
8:         $\mathbf{x} = \bar{\mathbf{C}}(:, r)$             ▷ $r$th column of the matrix $\bar{\mathbf{C}}$
9:         $\mathbf{y} = \mathbf{x}(\texttt{idx})$
10:        $\mathbf{X} = [\;\,]$                 ▷ Initialize the matrix $\mathbf{X}$ to zeros
11:        for $i \leftarrow 1, \text{size}(\mathcal{L})$ do
12:            $\ell = \mathcal{L}(i)$
13:            $\mathbf{X}(:, i) = \mathbf{x}(\texttt{idx} - \ell)$
14:        end for
15:        $\mathbf{w} = (\mathbf{X}^T\mathbf{X} + \lambda_w \mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$              ▷ $r$th column of $\mathbf{W}$
16:        $\mathbf{W} = [\mathbf{W}; \mathbf{w}]$
17:    end for
18: end procedure
```

This algorithm learns the weights of the temporal graph, given the matrix $\mathbf{C}$ and the lag set $\mathcal{L}$. We use ridge regression to avoid potential singularity of $(\mathbf{X}^T\mathbf{X})$. We can also use LASSO, or Elastic Net to learn the weights. However, numerical experiments show that it is sufficient to use Ridge Regression. This algorithm is very efficient and its cost is essentially the cost of a ridge regression per iteration, i.e., R times the cost of a ridge regression. In Matlab, the inversion is simply done by the backslash operator.

The choice of the lag set $\mathcal{L}$ is flexible, and since the weights are learned, $\mathcal{L}$ can be chosen to be discontinuous (if domain knowledge is included such as periodicity and/or seasonality); or very large to account for long range dependencies. The model is explainable, and the temporal graph dependency gets sparsified when using LASSO.

5 Prediction

In this section, we present the autoregressive procedure used to allow multi-step forecasting using time regularized tensor decomposition (see Algorithm 2). The main idea is to learn the weights of the lag set in the graph-based temporal dependencies by using Algorithm 1, and then use them in the prediction procedure. The (vector) autoregressive model is one of the most successful, flexible, and easy to use models for the analysis of multivariate time-series.

Algorithm 2. Forecast

```
1: $\bar{\mathbf{C}} \leftarrow \texttt{Transpose}(\mathbf{C})$
2: procedure $[\bar{\mathbf{C}}_{\text{new}}] = \text{Forecast}(\bar{\mathbf{C}}, \mathcal{L}, \mathbf{W}, \texttt{hope})$
3:     $T \leftarrow \texttt{size}(\bar{\mathbf{C}}, 2)$
4:     $\bar{\mathbf{X}} \leftarrow [\bar{\mathbf{C}}; \mathbf{0}]$                    $\triangleright\ \mathbf{0} \in \mathbb{R}^{R\times \texttt{hope}}$
5:     for $t \leftarrow (T+1), (T+\texttt{hope})$ do
6:         $\bar{\mathbf{X}}(:,t) \leftarrow \sum_{\ell\in\mathcal{L}} \mathbf{W}(:,\ell) \odot \bar{\mathbf{X}}(:,t-\ell)$
7:     end for
8:     $\bar{\mathbf{C}}_{\text{new}} \leftarrow \bar{\mathbf{X}}(:,(T+1):end)$
9: end procedure
10: $\mathbf{C}_{\text{new}} \leftarrow \texttt{Transpose}(\bar{\mathbf{C}}_{\text{new}})$
11: $\mathcal{G}_{\text{new}} = [\![\mathbf{A},\ \mathbf{B},\ \mathbf{C}_{\text{new}}]\!]$
```

Remark 5. Therefore, the main ideas are: (1) get the factors matrices **A**, **B** and **C** from the regularized tensor factorization of $\mathcal{G}$ (regularized CP-WOPT); (2) learn the weights of the temporal graph (data-driven approach); (3) get $\mathbf{C}_{\text{new}}$ from **C** by auto-regression (using the weighted temporal graph); and finally (4) forecast: $\mathcal{G}_{\text{new}} = [\![\mathbf{A},\ \mathbf{B},\ \mathbf{C}_{\text{new}}]\!]$.

Since the weight matrix $\mathbf{W}^{(\ell)}$ is diagonal, the matrix-vector product $\mathbf{W}^{(\ell)}\bar{\mathbf{c}}_{t-\ell}$ is efficiently computed by using the Hadamard product, represented by the symbol $\odot$.

6 Experiments

In this section, we describe the experimentation setting and discuss results achieved by three different algorithms against our TRTF (Temporal Regularized Tensor Factorization). Two of the baseline algorithms are based on matrix factorization that use different ways to represent speed information. LSM-RN (Latent Space Road Network) captures speed data into T snapshots of $N \times N$ matrices where each cell $s_t(i,j)$ reports the speed observed in the road segment linking i to j at time t [4]. TRMF (Temporal Regularized Matrix Factorization) uses a matrix in which rows are road segments and columns are timestamps, i.e, each row is the time-series of speeds of a particular road segment [25]. The third baseline is CP-WOPT (Weighted Optimization), which is the state of the art method for tensor decomposition with missing data [1].

We evaluate the four algorithms on two real datasets of speed readings observed in road networks of two cities. The evaluation concerns two tasks: (1) missing value completion and (2) forecasting of future values. In addition to their general accuracy, we are interested in how different algorithms perform in the two scenarios of rush-hour (e.g., 7am–8am) versus non-rush hour (e.g., 10am–11am). All reported results are averages of several runs on four weeks of data.

6.1 Traffic Datasets

We use real traffic data from two different cities, namely (1) Aarhus in Denmark as a mature and developed Northern European city, and (2) Doha in Qatar as one of the most fast growing cities in the world that suffers a huge congestion problem due to its extra-ordinary yearly population growth ($\approx 10\%$).

6.1.1 *Aarhus*

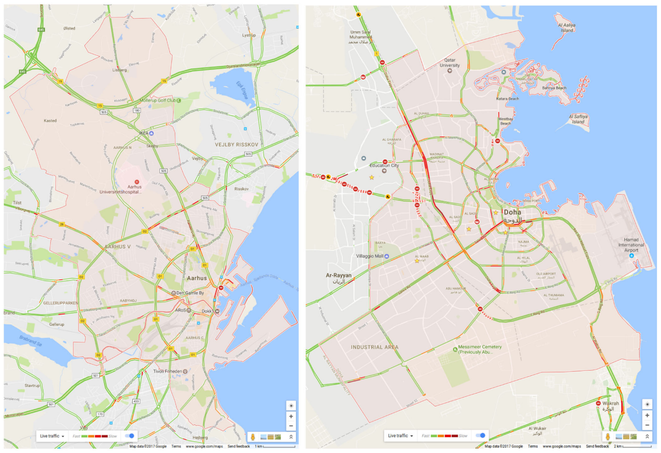
This data is part of Aarhus smart city project and has been made available for download via CityPulse Project Portal,[1] see [12]. The data is used for various traffic analytics such as day-to-day developments of roadworks, major changes, and abnormal events. Data is collected from 126 measuring points using Bluetooth sensors on selected stretches in the municipality of Aarhus. By looking at the time it takes for a car to drive from one measuring point to the next, its speed is computed based on the distance and travel time. Speed time-series are then aggregated at five-minutes granularity. In this study, we use the chunk of data captured between February 2014 and March 2014. The fraction of Aarhus road network covered by the data is modeled as a directed graph consisting of 136 nodes (junctions) and 443 edges (road segments) in two periods, the first period spans between February 13th, 2014 to June 09th, 2014 and the second period spans between August 1st, 2014 to September 30th, 2014 see Fig. 4a.

The number of speed readings captured in this dataset is 20,402,174. Only 2,055,404 speed readings are missing from 5 minutes-granularity time-series of all road segments. This represents approximately 10 percent of missing values.

6.1.2 *Doha*

The data is provided by the Qatar Mobility Innovation Center, a local company that deployed different types of traffic sensors and mobile applications to monitor the traffic and mobility in the city. The data consist of two files: one for the details of the road network such as IDs of road segments, their start and end points as geographic coordinates, and length of the segments; and the second file contains timely speed readings for a subset of segments. As one would expect, the data is far from being complete. Many road segments lack readings, and there is a high variability in the completeness of the speed time-series of different road segments. That is, the percentage of missing values varies a lot from a road segment to another, depending on its importance in the city traffic, its proximity to the city center, the presence of sensors, the network coverage, and many other parameters. Given the nature of the data sources used by the company (few Bluetooth sensors), it is often the case that there is not enough information to calculate speeds for all road segments. Thus, we observe a very high rate of missing values reaching up to 99 percent in some cases. The part of Doha's road network covered by the dataset

1. http://iot.ee.surrey.ac.uk:8080/datasets.html



(a) Aarhus map (b) Doha map

Fig. 4. Google Maps screenshots of the cities of Aarhus (left panel) and Doha (right panel). We highlight the main roads with their associated congestion level at the time of the creation of the images. The scale is provided in the bottom-right of the two images.

consists of 979 road segments out of which only 512 have at least one speed reading in the period spanning January 1st, 2015 00:00 to January 31st, 2015 23:59. The total number of speed readings is 3,574,497 which represent 15.6 percent of all possible 22,855,680 readings of the complete minute-level time-series for all road segments. The other major challenge of this dataset is the low connectedness of segments that have speed readings, i.e., the sparsity of the dataset see Fig. 4b.

6.1.3 *Some Remarks Regarding the Datasets*

It is important to notice that the two datasets show a completely different set of properties. First, the fraction of missing values in Doha is 96 percent whereas Aarhus suffers 10 percent missing values. This could have a significant impact on the quality of different algorithms. Second, because of the high level of data sparsity in Doha, the segments for which speed readings are available are weakly connected in the road network–modeled as directed graphs–i.e., the adjacency matrix of the dual road network graph is very sparse. This is another challenge that could presumably make it difficult to effectively learn the role of spatial properties in speed inference. In sum, we believe that having access to those two different types of real datasets is a good opportunity to understand how different methods respond under different circumstances.

6.2 Evaluation Metrics

Based on our literature review, we found that it is common to use different metrics to evaluate the quality of both missing value completion and forecasting accuracy in the context of traffic prediction [4], [25]. Thus, we report the results of the four algorithms based on the following metrics: Mean Absolute Percentage Error (MAPE), Root Mean Square Error (RMSE), Normalized RMSE, and Normalized Deviation (ND), each of which captures a different aspect of the quality and accuracy of the results. Definitions of the used metrics are given below.

$$\mathtt{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{\|\mathbf{y}_i - \hat{\mathbf{y}}_i\|}{\|\mathbf{y}_i\|} \tag{17}$$

TABLE 1
Results for Missing Value Completion Obtained Across Eighteen Hours Tensors (6 am to 12 am) Averaged Over Five Working Days of Three Weeks for Both Aarhus and Doha

	Aarhus				Doha			
Mis%	nd	nrmse	rmse	mape	nd	nrmse	rmse	mape
				LSM-RN				
10%	0.430	0.550	23.590	55.260	0.480	0.570	24.260	46.990
30%	0.550	0.680	29.360	60.720	0.530	0.630	26.730	50.970
50%	0.680	0.800	34.760	69.520	0.590	0.700	30.630	56.090
70%	0.790	0.900	39.100	78.500	0.700	0.810	35.690	65.150
90%	0.890	0.990	42.680	87.030	0.890	0.990	42.920	86.200
				TRMF				
10%	0.144	0.209	9.040	18.300	0.353	0.472	18.820	39.040
30%	0.155	0.215	9.270	19.440	0.388	0.515	20.548	42.220
50%	0.155	0.223	9.589	20.340	0.424	0.560	22.298	45.700
70%	0.159	0.229	9.829	21.060	0.463	0.603	24.049	49.340
90%	0.165	0.229	9.848	21.060	0.504	0.656	26.137	53.300
				CP-WOPT				
10%	0.143	0.210	9.423	19.838	0.190	0.254	10.091	29.217
30%	0.149	0.217	9.731	21.381	0.195	0.262	10.481	30.291
50%	0.159	0.242	10.825	23.217	0.203	0.275	10.976	30.717
70%	0.183	0.275	12.304	26.624	0.235	0.333	13.264	33.558
90%	0.242	0.359	16.062	33.875	0.724	1.063	42.391	76.682
				TRTF				
10%	0.141	0.200	8.966	19.756	0.185	0.244	9.677	28.618
30%	0.148	0.209	9.349	21.333	0.187	0.247	9.891	29.185
50%	0.154	0.219	9.805	22.662	0.192	0.255	10.181	29.492
70%	0.169	0.245	10.969	25.196	0.194	0.261	10.410	29.905
90%	0.212	0.296	13.248	32.562	0.243	0.335	13.364	34.082

$$\mathtt{ND} = \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{y}_i - \hat{\mathbf{y}}_i\| \Big/ \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{y}_i\| \tag{18}$$

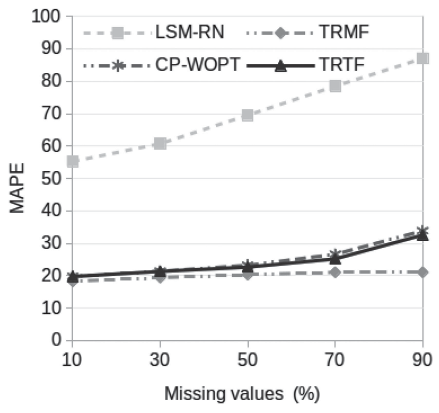
$$\mathtt{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2} \tag{19}$$

$$\mathtt{NRMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{y}_i - \hat{\mathbf{y}}_i\|^2} \Big/ \frac{1}{N}\sum_{i=1}^{N}\|\mathbf{y}_i\|. \tag{20}$$

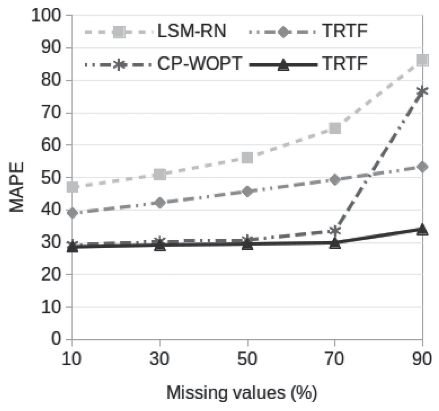
6.3 Missing Value Completion

We first report the completion accuracy of different algorithms under different percentages of missing values {10, 30, 50, 70, 90 percent}. Given the fact that the input tensors are sparse and comprise lots of missing values, the percentages we are testing with are relative to the known values.

The overall results with the best parameters for each dataset and each evaluation metric are reported in Table 1. For a close inspection, we report in Fig. 5 MAPE scores achieved by different methods in Aarhus (Fig. 5a) and Doha (Fig. 5b.) Among all algorithms, LSM-RN is the worst, especially in the cases where the data is very sparse. TRTF outperforms all methods in the case of Doha where we get a noisy and sparse data comparing to Aarhus, this later makes it difficult to other methods to correctly learn the latent space features. This is especially true in the case where only 10 percent of data is

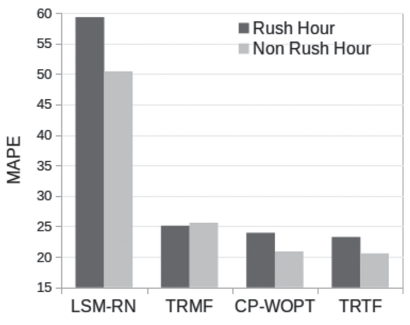


(a) Aarhus

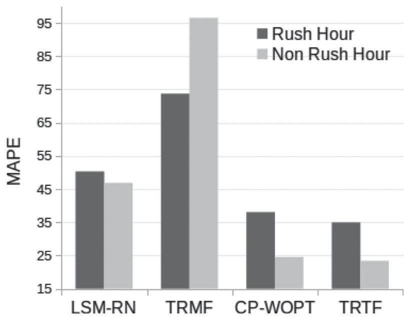


(b) Doha

Fig. 5. MAPE scores for different percentages of missing values.



(a) Aarhus



(b) Doha

Fig. 6. MAPE results for completion of (10 percent) missing values in rush versus non rush hours. Note that the scales are different.

available and in which TRTF clearly outperformed by far all other methods achieving a MAPE score of 34 percent against 53.3 percent for TRMF, 76.6 percent for CP-WOPT, and 99.9 percent for LSM-RN. The same trends can be seen for all other metrics in tables Aarhus (Table 1) and Doha (Table 1).

On the contrary, in the case of dense road network graphs and complete data, LSM-RN seems to slightly outperform our method. This indicates that in such scenarios, LSM-RN is able to accurately learn the spatial and temporal properties on the traffic phenomena.

6.3.1 Rush Hour Versus Non Rush Hour

Next, we investigate the accuracy of missing value completion in two different time windows, known to have different traffic dynamics: rush hours (congestion) and non rush hours (free flow). We choose the time window between 7 am to 8 am to represent the rush hour scenario and the one from 10 am to 11 am to represent the non rush hour scenario. Fig. 6 plots the results (MAPE scores) for the case of 10 percent missing values. The full set of results is reported in Table 2. Once again, our method TRTF, in most cases, outperforms other methods for both cities (Aarhus and Doha) and both cases (rush hour and non rush hour). While the difference between TRTF and CP-WOPT in the case of Aarhus (Fig. 6a) seems narrow in that TRTF achieves only 3 percent improvement over CP-WOPT, the difference is much significant in the case of Doha where TRTF achieves over 9 percent improvement compared to the second best method CP-WOPT. This means, that in the extreme cases of missing values and disconnected networks, the use of both temporal and spatial properties leads to lower errors compared to cases where only temporal properties are learned. Another interesting observation is that TRMF completely degenerated in completing missing values for Doha, especially in the case of non-rush hour, which could be explained by the low effectiveness of this method in case of large amount of missing values. Last, with an exception of TRMF, we see from this experiment that all three methods TRTF, CP-WOPT, and LSM-RN have had better completion scores in the case of non-rush hours compared to rush hours. The difference in gain is specifically noticeable for CP-WOPT and TRTF. This leads to the fact that traffic dynamic is more predictable in free-flow scenarios.

6.3.2 Weekly Patterns of MAPE Scores

As seen in the previous section, we demonstrated the effectiveness of TRTF method in general and in specific scenarios of rush and non rush hours. We plot in Fig. 7 the heatmaps of typical MAPE scores achieved by TRTF through-out a week. Note that we are only interested in the five working days in each city (Monday through Friday in Aarhus and Sunday through Thursday in Doha). For each day, we only report for the 18 hours in which it makes sense to monitor the traffic, i.e., from 6am to 12am. In the heat-maps, rows represent days of the week and columns represent hours of the day. Each cell reports the average level of MAPE error observed on a typical hour of that day (e.g., the typical level of error made at any Tuesday 9 am). Due to space limitation, we only plot heat-maps for our method. Unsurprisingly, the scores for Doha are noticeably higher than those for Aarhus which could be explained by the quality of the different datasets. Doha shows two distinguishable periods in which

TABLE 2
Aarhus and Doha Results in
Rush and Non Rush Hour Scenarios

Mis(%)	Aarhus				Doha			
	Rush hour		Non rush		Rush hour		Non rush	
	ND	MP	ND	MP	ND	MP	ND	MP
				`LSM-RN`				
10%	0.440	59.290	0.420	50.410	0.480	50.270	0.480	46.850
30%	0.550	60.400	0.550	62.030	0.540	56.130	0.530	51.720
50%	0.670	68.330	0.680	71.350	0.610	57.630	0.620	57.640
70%	0.780	76.850	0.790	79.810	0.740	66.940	0.740	68.090
90%	0.880	86.100	0.890	88.090	0.910	86.800	0.910	88.340
				`TRMF`				
10%	0.181	25.100	0.187	25.600	0.668	73.700	0.970	96.500
30%	0.210	28.801	0.221	31.100	0.735	83.200	0.936	99.200
50%	0.229	31.100	0.234	33.400	0.746	84.000	0.907	95.300
70%	0.269	36.800	0.268	38.500	0.704	79.700	0.862	91.600
90%	0.289	39.500	0.277	38.700	0.710	80.800	0.819	88.000
				`CP-WOPT`				
10%	0.148	23.900	0.142	20.800	0.241	38.000	0.175	24.500
30%	0.154	25.500	0.143	21.600	0.270	39.500	0.166	23.000
50%	0.167	29.000	0.150	25.600	0.269	38.800	0.181	25.100
70%	0.196	36.201	0.162	25.800	0.327	45.700	0.216	27.600
90%	0.270	46.300	0.220	35.600	0.798	86.900	0.710	72.500
				TRTF				
10%	0.146	23.200	0.140	20.500	0.236	34.900	0.165	23.300
30%	0.154	24.800	0.141	21.700	0.249	37.900	0.165	23.100
50%	0.165	28.400	0.147	24.500	0.259	38.500	0.170	23.700
70%	0.183	32.800	0.158	25.700	0.275	41.000	0.169	23.200
90%	0.238	43.900	0.190	33.400	0.314	43.000	0.216	28.100

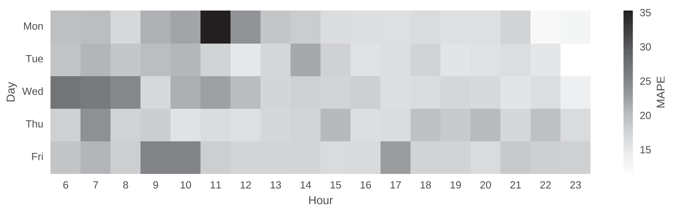
MP is used as an abbreviation for MAPE.

`TRTF` does not perform well. These periods correspond to the morning (7 am and 8 am) and evening (5 pm and 6 pm) commutes. In Aarhus, the general trend suggests that the method down-perform in the morning period (6 am to 12 pm), then things slowly improve in the afternoon and evening. This plot is particularly important in that it shows that completion error is never uniform through-out hours and days, especially when dealing with complex phenomenas like traffic. Thus, it is important to train different set of parameters for different relevant temporal segments.

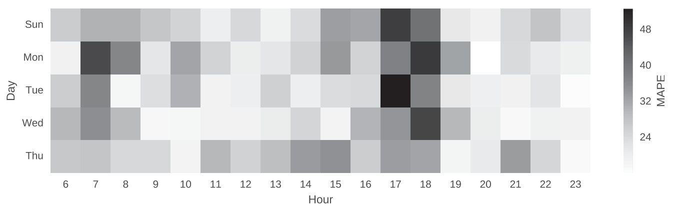
6.4 Forecasting Evaluation

In this set of experiments, we evaluate the quality of different methods in predicting future speed values of the road network in Aarhus and Doha. The two main parameters for all methods are (1) the horizon (h) which is the number of steps-ahead to predict and (2) the size of the training window. For the three methods `CP-WOPT`, `TRMF`, and `TRTF` that use auto-regression, we also vary the size of the lag set. Due to space limitation, we only report a subset of results obtained. The same trends are applicable to all other cases.

As an illustrative example, we assume a case in which we know the traffic status in each city up to 7 am (t), and try to predict the speeds beyond, i.e., at $\{t+1, t+2, \ldots t+12\}$ each step correspond to a horizon of 5 minutes. Fig. 8 reports the MAPE scores of different methods in prediction speeds at different horizons between 7:05am and 8am. The training set is fixed to 30 minutes before 7am (data from 6:30am to 7am) and the size of the lag set is 6. As one would expect, the general trend is that the error increases as the horizon increases which could be explained by the error accumulation problem.



(a) Aarhus



(b) Doha

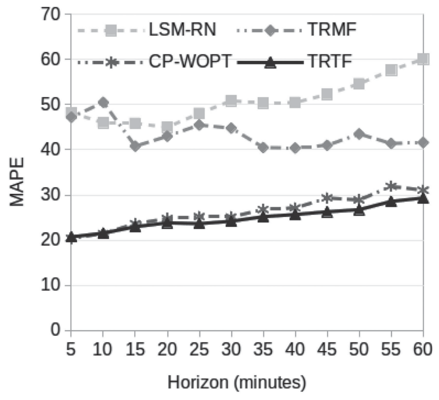
Fig. 7. Weekly heatmaps of MAPE scores achieved by `TRTF` in Aarhus and Doha in case of 10 percent missing values.

We observe that tensor-based methods (`CP-WOPT` and `TRTF`) that use autoregressive regularizer procedure outperform by far the matrix based methods `LSM-RN` & `TRMF`. This demonstrates that a good incorporation of temporal properties into a simple tensor factorization algorithm, yields remarkably good results compared to those achieved by matrix-factorization methods (i.e., `LSM-RN` and `TRMF`). For instance, in the case of lag 6 (Fig. 8a), we observe that the gain in MAPE score of our method is about 23.65 percent compared to `LSM-RN` and 28.96 percent compared to `TRMF`, which is very significant. The reason for which `TRTF` is outperforming `CP-WOPT` is that `TRTF` includes both a spatial and a temporal regularizers that are not embedded in `CP-WOPT`. These results are valid across different experimental settings as shown in (Fig. 8b) and (Fig. 8c).

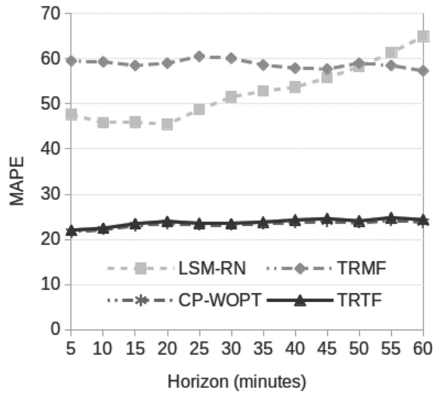
Next, we investigate the impact of the lag set size on the accuracy of `TRTF` forecasting. We report in Fig. 9 the average MAPE scores observed in horizon bins of 15 minutes. We vary the size of the lag set that we assume is continuous, taking values in {4, 6, 12}. Interestingly, we find that the accuracy of the predictions are much better in the case of Aarhus compared to Doha. This is probably due to the quality of the data and the connectedness of the road network. In the case of Aarhus (Fig. 9a), we found that a lag of 6 yields better results; whereas in Doha, the best lag size turned to be 4.

6.5 Time Performance

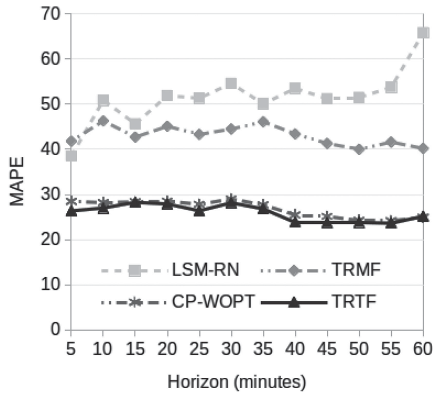
We report in Table 3 the performance results of different methods observed in the training phase of the missing value completion task. We see that `TRMF` has similar running time for both Aarhus and Doha datasets. On the other hand, `CP-WOPT` and `TRTF` are one order of magnitude faster in the case of sparse data (Doha), and become slower for bigger datasets. This behavior is observed in tensor-factorization-based methods as well. `LSM-RN` on the other side requires much more time to learn the spatio-temporal properties



(a) Aarhus (lag=6)

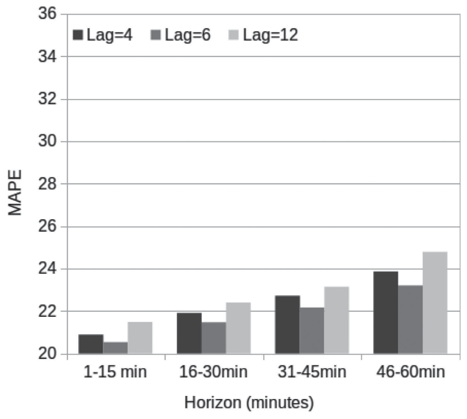


(b) Aarhus (lag=24)

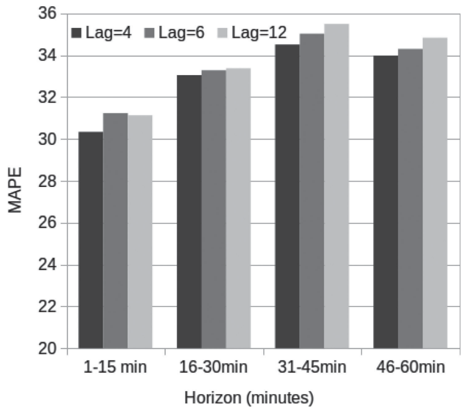


(c) Doha (lag=24)

Fig. 8. Forecasting results for predicting traffic beyond 7 am using a continuous lag set of sizes 6 for Aarhus (panels a) and 24 (panels b, c) trained in windows of size 30 minutes and 2 hours respectively.



(a) Aarhus



(b) Doha

Fig. 9. Prediction error achieved by `TRTF` with different sizes of continuous lag sets, all trained in a two hours window.

than any other method, which is mainly due to the costly phase of learning different matrices representing the attributes of the vertices of the road network.

6.5.1 Fit and Convergence

We finally discuss the convergence of our algorithm `TRTF` in its best case scenarios. Convergence is captured here in terms of number of iterations required to reach the convergence criteria. In practice, we compute the fit score at each iteration and declare convergence when the fit gets equal or greater to 0.98, i.e.,

$$\text{fit} = 1 - (||\mathcal{G} - \hat{\mathcal{G}}||/||\mathcal{G}||), \tag{21}$$

where $||\mathcal{G}||$ is the Frobenius norm of the three-way tensor of speed values. Fit curves are reported in Fig. 10. We see that our algorithm converges after 27 iterations only in the case of Doha compared to 61 iterations for Aarhus. Time wise, convergence happens within 4.30 seconds in Doha versus 112.9 seconds in Aarhus. This significant difference in time is mainly due to the fact that tensors tend to be slower when they are full.

6.5.2 New Set of Experiments

PeMS: to get access into the Freeway Performance Measurement System (PeMS) database for California State, one would need to apply for an account in the following URL: http://pems.dot.ca.gov/?dnode=apply. One may learn more about the PeMS in their FAQs section, i.e., http://pems.dot.ca.gov/?dnode=Help&content=help_faq and in the User Manual in the shared folder https://drive.google.com/open?id=1YYPsXzwktNQetdsd7C0BhTiWNDyzN5R.

Fetching real-time feeds data from PeMS can be done via FTP. We wrote a script that does the following:

Step 1 script logs in into PeMS using login() function. Users need to provide username and password in login();

Step 2 script reads station_ids.csv, where station_ids can be retrieved from 'EXPORT TEXT' button, see step2.png example for Los Angelos;

Step 3 script goes through each ID in station_ids, and generates the URL to retrieve ID's time series. The

TABLE 3
Comparing Running Times (in seconds) of the Four Methods in the two Cities of Aarhus and Doha

	Aarhus				Doha			
Mis(%)	LSM-RN	TRMF	CP-WOPT	TRTF	LSM-RN	TRMF	CP-WOPT	TRTF
10%	22.68	1.81	72.85	73.31	2.07	1.96	3.23	3.45
30%	25.08	1.47	48.33	48.74	1.93	2.04	2.52	2.37
50%	25.44	1.46	30.71	29.91	1.96	1.94	1.88	2.06
70%	26.88	1.43	11.71	11.95	2.1	1.91	1.47	1.75
90%	31.44	1.26	2.30	2.36	2.2	1.69	0.77	1.11

manual process for step 3 can be seen in step3-1.png, step3-2.png, step3-3.png. The column headers can be seen in script_result.png;

Step 4 repeat Step 3 for ID for the required number of days.

Some parameters which can be changed in prepare_response() are root, weeks_to_read and start_time. We tried different ways to obtain the adjacency matrix from the PeMS website, with no success. So we contacted PeMS support to provide the adjacency matrix for some cities in the database, but it seems road connections data is not publicly available, and hence retrieving the adjacency matrix feature is not readily available.

Therefore, it is important to find a different real traffic dataset that can provide the -required- adjacency matrix.

UK Department of Transportation. UK government provides a system to obtain Traffic flow time-series data, as well as Road network in a .shp file. The time-series and the network are in two separate systems, but they share the same road IDs. We have written a code to form the adjacency matrix from the .shp file, as well as link the IDs between the two systems, so that the time-series tensor is linked to the adjacency matrix. The data and the network can be downloaded from the below URLs: (1) Traffic flow Time-series data: http://tris.highwaysengland.co.uk/detail/trafficflowdata; (2) road IDs: http://tris.highwaysengland.co.uk/ConversionTable; (3) England's road network shapefile: https://data.gov.uk/dataset/9562c512-4a0b-45ee-b6ad-afc0f99b841f/highways-england-network-journey-time-and-traffic-flow-data. One may use ArcGIS to view the .shp file. The data can be downloaded via API or manually: (1) Webtris API: http://webtris.highwaysengland.co.uk/api/swagger/ui/index#!/Areas/Areas_Get_0; and (2) Webtris R Interface: https://cran.r-project.org/web/packages/webTRISr/index.html.

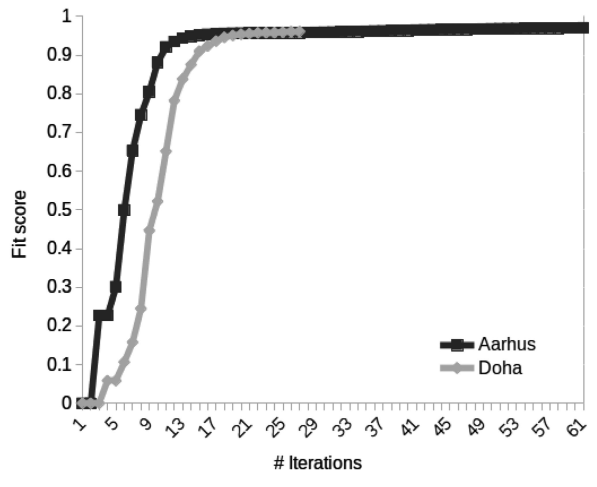


Fig. 10. Aarhus and Doha Fit curves for convergence of TRTF. Convergence happens at the 27th iteration in the case of Doha versus 61st in the case of Aarhus.

TABLE 4
Duration of Aarhus Datasets

Dataset 1	Jan. 2014 – June 2014 (6 months)
Dataset 2	Aug. 2014 – Sept. 2014 (2 months)
Dataset 3	Oct. 2014 – Nov. 2014 (2 months)
Dataset 4	Jauly 2015 – Nov. 2016 (17 months)

Remark 6. We have developed a script to extract the Adjacency matrix from England's road network; and we realized that Working on the England dataset will require a lot of time, and can be the scope of another paper. Therefore, we decided to form the adjacency matrices for Denmark Aarhus, i.e., it is the same city as the earlier experiments, but with more recent data, see Table 4. The dataset 1, i.e., for the period January-June 2014 was already done with good results; and we wanted to run experiments on the new datasets, in order to assess how the proposed methodology performs.

It is clear that the proposed methodology competes with other state-of-the-art algorithms (LSM-RN, TRMF), as well as the baseline (CP-WOPT) on three real datasets, see Figs. 11, 12, 13, 14, 15 and 16. The flexibility of the proposed algorithm TRTF can be noted when tested under different

TABLE 5
Comparison Results for the Different Methods for the City of Aarhus in Summer Time (August)

Mis(%)	Aarhus			
	MP	RMSE	NSRMSE	ND
	TRTF			
10%	26.2882	10.9853	0.23405	0.17067
30%	27	11.1041	0.23768	0.17388
50%	28.7722	11.5992	0.24789	0.18134
70%	29.8331	12.1604	0.26	0.19118
90%	35.6162	13.8344	0.29589	0.21929
	CP-WOPT			
10%	26.7094	11.1396	0.23729	0.173
30%	27.1077	11.2495	0.24083	0.17528
50%	28.1446	11.8353	0.25287	0.18243
70%	30.1609	12.6596	0.27075	0.19654
90%	36.396	18.1712	0.38771	0.25529
	CP-NMU			
10%	84.2773	42.3585	0.904	0.81786
30%	87.9959	44.5859	0.95373	0.86359
50%	92.8946	47.2519	1.0096	0.91704
70%	97.3607	49.7965	1.0641	0.96599
90%	99.6529	51.4831	1.1006	0.99476
	CP-ALS			
10%	79.6837	40.1664	0.85941	0.77166
30%	84.9157	42.8764	0.91719	0.83114
50%	90.3406	46.1207	0.98603	0.89406
70%	95.9838	49.2003	1.0528	0.95379
90%	99.5676	51.4104	1.0995	0.99387

MP is used as an abbreviation for MAPE.

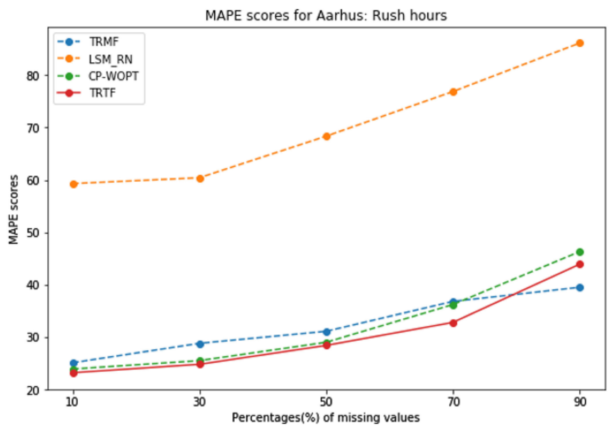


Fig. 11. MAPE scores generated from LSM-RN increase significantly past 30 percent of missing values in the tensor. LSM-RN performs the worst irrespective of the percentage of missing values. Across all values of percentages of missing values, TRTF performs the best with the lowest MAPE scores. TRMF and CP-WOPT perform optimally as well with MAPE scores very close to those of TRTF at 10, 30, 50 and 70 percent missing values. At 90 percent missing values, TRMF has a MAPE score lower than TRTF and CP-WOPT. This, however, is only observed in this instance out of all four instances.

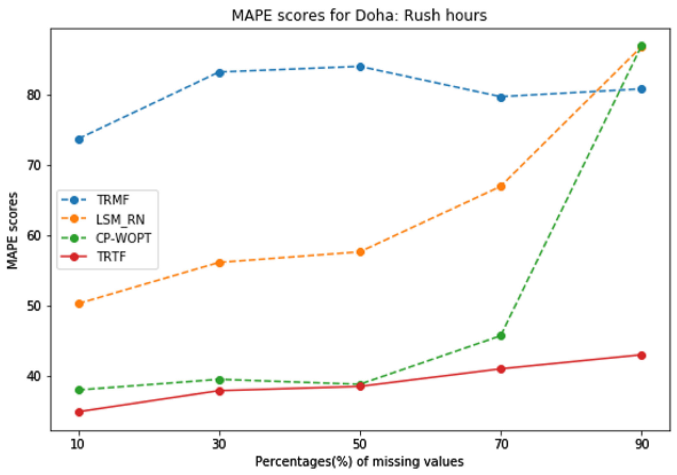


Fig. 13. Similar to the other instances, TRTF has the lowest MAPE scores for all percentages of missing values. CP-WOPT performs almost as well as TRTF but has a big jump at 90% missing values when compared with TRTF. TRMF performs the worst but has a lower MAPE score than LSM-RN and CP-WOPT at 90% missing values.

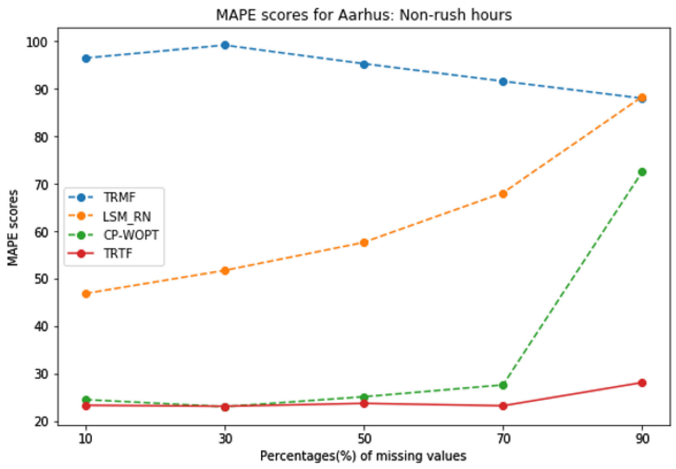


Fig. 12. TRTF performs the best our of all four algorithms. It has lowest MAPE scores at all five values for percent of missing values. All algorithms except TRMF has a big spike in MAPE scores going from 70% missing values to 90% missing values. CP-WOPT and TRTF perform significantly better than all other algorithms at all percentages of missing values. However, CP-WOPT has a much bigger change in MAPE scores at 90% compared to TRTF. LSM-RN seems to have consistent increase in MAPE scores as percent of missing values increases.

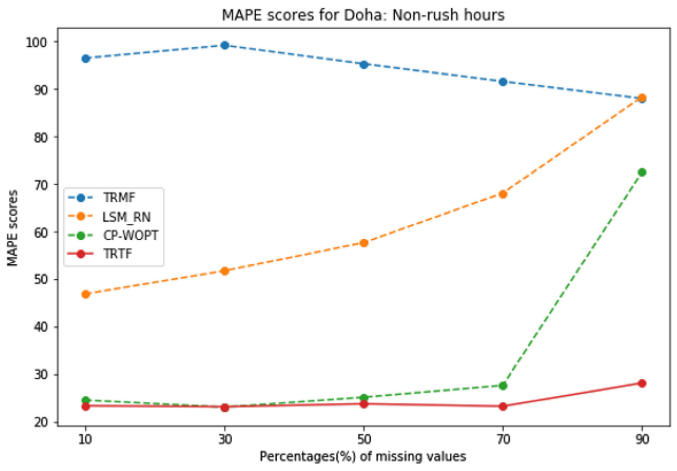


Fig. 14. The scores from all four algorithms follow a very similar trend as the instance with Rush hours in Doha. TRTF performs better than all other algorithms and has a much lower MAPE score at 90% of missing values compared to the other algorithms.

settings, as each dataset is composed of complex features which have effects on the traffic flow data. In fact, Aarhus, between February to June is accustomed to foggy weather with moderate rain as well as snowfall. These environment conditions hinders visibility and influences the chance of an accident, which toggles slower traffic flow. The warmer periods in Aarhus, between August to September, are arguably the busiest season for tourism. Routes near museums, beaches, and shopping are expected to have heavier traffic, while routes neighboring closed schools, due to vacation, may be lighter. Also, due to Doha's setting of rapid development, road expansion works can be seen every single day. Transforming roundabouts to intersections to bridges and tunnels in a fast-paced city evolution is what the citizens have experienced during recent years. Because of this unique setting, state-of-the-art web mapping services such as finding shortest travel time between two locations can be deemed unreliable and out of date. Hence, traffic flow prediction on the city of Doha can be of utmost challenge to algorithm predictions.

Except for rush hours in Aarhus, in all other instances, TRTF has lower MAPE scores than all other algorithms. During rush hours in Aarhus, MAPE scores go up for TRTF at 90 percent of missing values and TRMF performs better than TRTF. On the other hand, TRMF seems to perform not as well as other algorithms but MAPE scores go down as the percentage of missing values increase. This can lead to believe that TRMF is the least sensitive to the underlying geography of the location; whereas LSM-RN, CP-WOPT and TRTF perform worse as the percentage of missing values goes up.

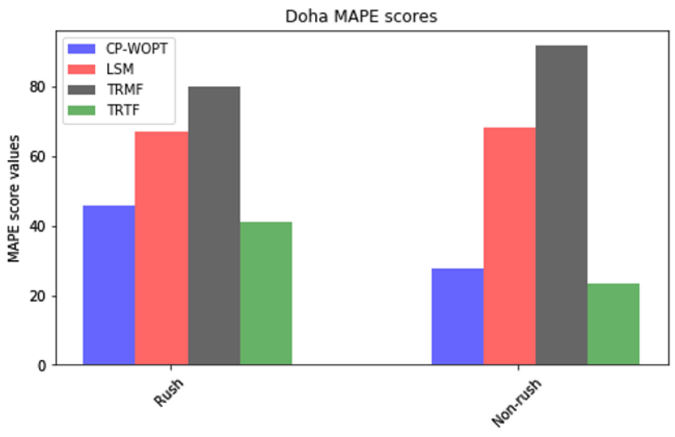


Fig. 15. During Rush hours: TRTF has the lowest MAPE score and thus, performs better than all other algorithms. During non-rush hours: TRTF seems to perform the best here as well. TRMF seems to worse during non-rush hours, which is different from the other three algorithms which seem to perform better during non-rush hours.

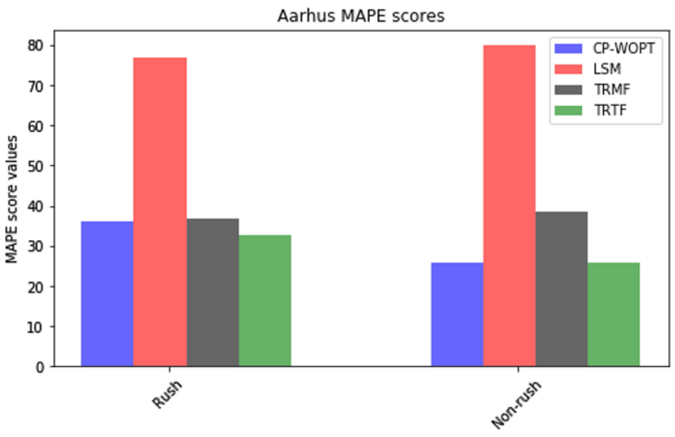


Fig. 16. Results from Aarhus seem to be different as compared to Doha. TRMF performed worse for Doha than it did for Aarhus. TRMF seems to perform well but it performs worse during non-rush hours which seems like a frequent pattern across all instances.

7 Conclusion

We presented in this paper TRTF, an algorithm for temporal regularized tensor decomposition. We show how the algorithm can be used for several traffic related tasks such as missing value completion and forecasting. Our algorithms incorporates both spatial and temporal properties into the tensor decomposition procedure, thus learning better factors. We also, extended TRTF with an autoregressive procedure to allow for multi step-ahead forecasting of future values. We compare our method to recently developed algorithms that deal with the same type of problems using regularized matrix factorization, and show that under many circumstances, TRTF does provide better results. This is particularly true in cases where the data suffers from high proportions of missing values, which is common in the traffic context. For instance, TRTF achieves a 20 percent gain in MAPE score compared to the second best algorithm (CP-WOPT) in completing missing values in the case of extreme sparsity observed in Doha.

As future work, we will first focus on adding non-negativity constraints to TRTF, although the highest fraction of negative values generated by our method throughout all the experiments did not exceed 0.7 percent. Our second focus will be to optimize TRTF training phase in order to increase its scalability to handle large dense tensors, and to implement it on a parallel environment.

Acknowledgments

The authors would like to acknowledge and thank Dr. Fethi Filali from the Qatar Mobility Innovation Center for providing the data for Doha. They also thank the anonymous reviewers for their careful reading of the manuscript and their many insightful comments and suggestions. This work was conducted while A. Sharma, T. Zanouda and J. Srivastava were in residence at the Qatar Computing Research Institute.

References

[1] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, "Scalable tensor factorizations with missing data," in *Proc. SIAM Int. Conf. Data Mining*, 2010, pp. 701–712.
[2] C. Chen, J. Kwon, J. Rice, A. Skabardonis, and P. Varaiya, "Detecting errors and imputing missing data for single-loop surveillance systems," *Transp. Res. Rec.: J. Transp. Res. Board*, vol. 1855, pp. 160–167, 2003.
[3] C. Chen, Y. Wang, L. Li, J. Hu, and Z. Zhang, "The retrieval of intra-day trend and its influence on traffic prediction," *Transp. Res. Part C: Emerg. Technol.*, vol. 22, pp. 103–118, 2012.
[4] D. Deng, C. Shahabi, U. Demiryurek, L. Zhu, R. Yu, and Y. Liu, "Latent space model for road networks to predict time-varying traffic," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1525–1534.
[5] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *ACM Trans. Knowl. Discovery Data*, vol. 5, no. 2, 2011, Art. no. 10.
[6] Z. Fan, X. Song, and R. Shibasaki, "CitySpectrum: A non-negative tensor factorization approach," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2014, pp. 213–223.
[7] N. Gillis and F. Glineur, "Low-rank matrix approximation with weights or missing data is NP-hard," *SIAM J. Matrix Anal. Appl.*, vol. 32, no. 4, pp. 1149–1165, 2011.
[8] Y. Han and F. Moutarde, "Analysis of large-scale traffic dynamics in an urban transportation network using non-negative tensor factorization," *Int. J. Intell. Transp. Syst. Res.*, vol. 14, no. 1, pp. 36–49, 2016.
[9] K. Henderson *et al.*, "RolX: Structural role extraction & mining in large graphs," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 1231–1239.
[10] M. G. Karlaftis and E. I. Vlahogianni, "Statistical methods versus neural networks in transportation research: Differences, similarities and some insights," *Transp. Res. Part C: Emerg. Technol.*, vol. 19, no. 3, pp. 387–399, 2011.
[11] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.
[12] S. Kolozali, M. Bermudez-Edo, D. Puschmann, F. Ganz, and P. Barnaghi, "A knowledge-based approach for real-time iot data stream annotation and processing," in *Proc. IEEE Int. Conf. Internet Things, and IEEE Green Comput. Commun., and IEEE Cyber Phys. Social Comput.*, 2014, pp. 215–222,
[13] P. M. Kroonenberg, *Three-Mode Principal Component Analysis: Theory and Applications*, vol. 2. Leiden, Netherlands: DSWO Press, 1983.
[14] B. Kulis, M. Sustik, and I. Dhillon, "Learning low-rank kernel matrices," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 505–512.
[15] D. Ni, J. D. Leonard, A. Guin, and C. Feng, "Multiple imputation scheme for overcoming the missing values and variability issues in its data," *J. Transp. Eng.*, vol. 131, no. 12, pp. 931–938, 2005.
[16] QMIC, "Qatar traffic report for 2016," Technical report, Qatar Mobility Innovations Center, 2016. [Online]. Available: http://www.qmic.com/news/qmic-launches-the-qatar-traffic-report-for-201 6
[17] R. A. Rossi, B. Gallagher, J. Neville, and K. Henderson, "Modeling dynamic behavior in large evolving graphs," in *Proc. 6th ACM Int. Conf. Web Search Data Mining*, 2013, pp. 667–676.

[18] J. Shang, Y. Zheng, W. Tong, E. Chang, and Y. Yu, "Inferring gas consumption and pollution emission of vehicles throughout a city," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 1027–1036.
[19] K. Shin, L. Sael, and U. Kang, "Fully scalable methods for distributed tensor factorization," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 100–113, Jan. 2017.
[20] K. Takeuchi, R. Tomioka, K. Ishiguro, A. Kimura, and H. Sawada, "Non-negative multiple tensor factorization," in *Proc. IEEE 13th Int. Conf. Data Mining*, 2013, pp. 1199–1204.
[21] H. Tan, G. Feng, J. Feng, W. Wang, Y.-J. Zhang, and F. Li, "A tensor-based method for missing traffic data completion," *Transp. Res. Part C: Emerg. Technol.*, vol. 28, pp. 15–27, 2013.
[22] G. Tomasi and R. Bro, "Parafac and missing values," *Chemometrics Intell. Laboratory Syst.*, vol. 75, no. 2, pp. 163–180, 2005.
[23] Y. Wang, Y. Zheng, and Y. Xue, "Travel time estimation of a path using sparse trajectories," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 25–34.
[24] J.-R. Xu, X.-Y. Li, and H.-J. Shi, "Short-term traffic flow forecasting model under missing data," *J. Comput. Appl.*, vol. 30, no. 4, pp. 1117–1120, 2010.
[25] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 847–855.
[26] F. Zhang, N. J. Yuan, D. Wilkie, Y. Zheng, and X. Xie, "Sensing the pulse of urban refueling behavior: A perspective from taxi mobility," *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, 2015, Art. no. 37.
[27] H.-S. Zhang, Y. Zhang, Z.-H. Li, and D.-C. Hu, "Spatial-temporal traffic data analysis based on global data management using MAS," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 267–275, Dec. 2004.
[28] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: Concepts, methodologies, and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 3, 2014, Art. no. 38.
[29] Y. Zheng, T. Liu, Y. Wang, Y. Zhu, Y. Liu, and E. Chang, "Diagnosing new york city's noises with ubiquitous data," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2014, pp. 715–725.

Abdelkader Baggag is a senior scientist with the Qatar Computing Research Institute, and an associate professor of data science at Hamad Bin Khalifa University, where he teaches advanced Machine Learning. His research focuses on developing data-driven models for finding patterns in complex data (mobility and health data) and implementing these methods in high-performance solutions, in particular multidimensional data and sequence of states data to support domain experts in traffic using sensors data, and eHealth for analyzing large-scale wearable sensor signals. His expertise is machine learning, representation learning, temporal causal modeling, artificial intelligence for health and mobility analytics, and missing data imputation.

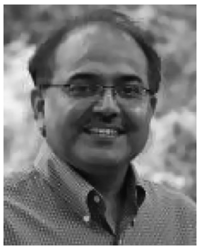
Sofiane Abbar is a researcher and senior software engineer with the Social Computing Department, Qatar Computing Research Institute, Hamad Bin Khalifa University. His research interests include social computing, machine learning, and data-driven approaches for urban analytics.

Ankit Sharma is currently working toward the PhD degree in the Computer Science Department, University of Minnesota, Minneapolis. He was a research associate with the Social Computing Department, Qatar Computing Research Institute, Hamad Bin Khalifa University, when this work was conducted.

Tahar Zanouda is a data scientist at Ericsson, Sweden. His research interest lies at the intersection of machine learning and urban mobility. He was a research associate in the Social Computing department at Qatar Computing Research Institute, Hamad Bin Khalifa University, at the time of this work.

Abdulaziz Al-Homaid is a research associate with the Qatar Computing Research Institute in Doha. He worked on improving access to Education data across response organizations during a fellowship with The United Nations Office for the Coordination of Humanitarian Affairs (UN OCHA). His research is on applying deep learning methods in Smart City domains such as city-wide traffic prediction and e-Health.

Abhiraj Mohan received the BS degree in computer science and engineering from the University of Minnesota at Twin Cities. His interests include machine learning and data science. He has worked on detecting behavior patterns for users from wearable technology, and on imputing missing values from traffic data. He has been a recipient of the UROP Scholarship at Minnesota, and has been on the Dean's list multiple times during his studies.



Jaideep Srivastava is a professor with the Department of Computer Science and Engineering, University of Minnesota, Twin Cities. He was the chief scientist with the Qatar Computing Research Institute, Hamad Bin Khalifa University, when this work was conducted. He does research in databases, computing in social sciences, arts and humanities, data mining, and works on the use of advanced machine learning for many applications such as traffic analytics and health analytics. He is an IEEE fellow.