Accurate Recovery of Missing Network Measurement Data With Localized Tensor Completion

Kun Xie[1,2], Xiangge Wang[1], Xin Wang[3], Yuxiang Chen[1], Gaogang Xie[4,5],
Yudian Ouyang[1], Jigang Wen[4], Jiannong Cao[6], Dafang Zhang[1]
[1] College of Computer Science and Electronics Engineering, Hunan University, Changsha, China
[2] Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen, China
[3] Department of Electrical and Computer Engineering of the State University of New York at Stony Brook, USA
[4] Computer Network Information Center, Chinese Academy of Sciences
[5] School of Computer Science and Technology, University of Chinese Academy of Sciences
[6] Department of computing, The Hong Kong Polytechnic University, Hong Kong

***Abstract*—The inference of the network traffic data from partial measurements data becomes increasingly critical for various network engineering tasks. By exploiting the multi-dimensional data structure, tensor completion is a promising technique for more accurate missing data inference. However, existing tensor completion algorithms generally have the strong assumption that the tensor data have a global low-rank structure, and try to find a single and global model to fit the data of the whole tensor. In a practical network system, a subset of data may have stronger correlation. In this work, we propose a novel localized tensor completion model (LTC) to increase the data recovery accuracy by taking advantage of the stronger local correlation of data to form and recover sub-tensors each with a lower rank. Despite that it is promising to use local tensors, the finding of correlated entries faces two challenges, the data with adjacent indexes are not ones with higher correlation and it is difficult to find the similarity of data with missing tensor entries. To conquer the challenges, we propose several novel techniques: efficiently calculating the candidate anchor points based on locality-sensitive hash (LSH), building sub-tensors around properly selected anchor points, encoding factor matrices to facilitate the finding of similarity with missing entries, and similarity-aware local tensor completion and data fusion. We have done extensive experiments using real traffic traces. Our results demonstrate that LTC is very effective in increasing the tensor recovery accuracy without depending on specific tensor completion algorithms.**

I. Introduction

Monitoring the performance of a large network would involve a high measurement cost. For a network with n nodes, the cost of monitoring the operational states of the whole network is at the order of $O(n^2)$.

Some recent studies show that network monitoring data such as end-to-end latency and flow traffic have hidden spatio-temporal correlations. This inspires the development of novel sparse network monitoring technique [1]. Sample-based network monitoring is applied in this technique, where measurements are only taken between some random node pairs or at some intervals for a given node pair, and the data of other paths are inferred leveraging the spatio-temporal correlations in network monitoring data. As only a few paths need to be probed, the measurement cost can be largely reduced. The partial measurements data may further reduce when the network suffers from unavoidable data losses upon severe communication conditions.

Despite the benefit of reducing measurement overhead, network state tracking for anomaly detection and failure recovery is highly sensitive to the missing of performance data. Accurate recovery of missing values from partially observed network measurements is an important procedure in sparse network monitoring. The quality of missing data recovery highly relies on the specific inference/recovery technique adopted. Various studies have been made to handle and infer the missing monitoring data. Designed based on purely spatial [2]–[4] or temporal [5], [6] information, the data recovery performance of most known approaches is low. To utilize both spatial and temporal information, recent studies exploit matrix completion [1], [7]–[9] to recover the missing data from a low-rank matrix. Although these approaches present good performance under low data missing ratio, when the data missing ratio is large, the recovery performance suffers as the data analysis based on two-dimensional matrix has the limitation in extracting information.

For better data recovery, it helps to represent network monitoring data as a higher dimensional array called *tensor*, a higher-order generalization of vector and matrix. Exploiting the inherent relationship among higher dimensional data, tensor-based data analysis [10] has shown that tensor models can take full advantage of the multilinear structures to provide better data understanding with higher information precision. Despite the potential, existing tensor completion algorithms generally have the strong assumption that the tensor data have a global low-rank structure, and try to find a single and global model to fit the data of the whole tensor. In many practical applications, however, data entries in a large tensor may have different levels of correlation. Taking the 3-way Internet traffic monitoring tensor as an example, its three dimensions are origin destination (OD) pairs, time slots, and days. A subset of OD pairs may have similar end-to-end traffic behaviors in some time slots (i.e., working time) in part of days (i.e., working day). Such a subset of OD pairs/time slots/days may construct a sub-tensor with a lower rank.

It is well acknowledged that the ranks of sub-tensors would be lower if the data in the sub-tensors have higher correlation. For a given number of samples, a lower rank brings more accurate recovery of the missing data. Therefore, the recovery performance using sub-tensors should be better than using global large tensor. Moreover, if we consider the data from each sub-tensor as a set, applying a global model to complete a large tensor consisting of multiple low-rank sub-tensors is akin to fitting one single model to the concatenation of all the data sets. As different data sets have different structure features, a single model can not well capture features of all data sets, which results in lower missing data recovery accuracy.

To well exploit the local correlation inherent in data along the

three dimensions of tensor (OD pair, time slot, and day) for more accurate missing data recovery, we propose a novel Localized Tensor Completion model (called LTC). In LTC, we reorganize data in the tensor to form a set of sub-tensors, each containing more similar data along the three tensor dimensions. We recover each sub-tensor and fuse the recovery data from sub-tensors to recover the data of the large tensor.

Although it is promising, traffic data in the tensor are organized logically. That is, data in OD pair domain follow a specific sequence to arrange the OD pairs. Data in the time domain are organized based on sampling slot sequence, and data in the day domain are arranged with the order of days. The entries with adjacent indexes in a tensor dimension (i.e., domain) may not be similar. For example, the OD pair with the indexes 1 and 2 may be far away in the network, while data taken from day 1 (a Monday) and day 2 (a Tuesday) may have a larger difference than the ones from day 1 and day 8 where both are Monday. Thus it is impractical to build the sub-tensor through the direct partition of the original large monitoring tensor.

In light of the constraints of practical data, we propose to build sub-tensors around some anchor points, then estimate the local sub-tensors for each neighborhood of the anchor point. This solution, however, faces three major challenges:

- *Anchor points should capture the correlation of data and distribute evenly in the whole data region.* For the accurate recovery of the whole tensor data, anchor points should distribute evenly across different data regions to well capture the correlation of data. Each anchor point should be able to represent the data of its neighbors in the sub-region.
- *Anchor point selection should consider the sample density.* Measurement samples may not be taken uniformly, especially when sampling is performed dynamically following the traffic patterns or operational needs. Randomly selecting anchor points may lead to poor performance and low efficiency. If an anchor point is taken from a region with very sparse samples, the sub-tensor formed will contain very small number of samples and can not be accurately recovered. On the other hand, if too many anchor points are taken from a region with dense samples, many sub-tensors are built in the same area and be recovered with redundant computation and low efficiency.
- *Measuring data similarity in the presence of missing entries.* A sub-tensor should be built based on similarity (or called distance) among tensor data. However, this is hard if the monitoring data are incomplete to reduce the sampling overhead.

To address the three challenges, we propose a few techniques in LTC:

- **Candidate anchor point calculation based on locality sensitive hash (LSH)**. To more accurately recover data in each sub-tensor, we propose to exploit LSH to quickly reorder data and put data with higher correlation into the same sub-tensor. With traffic data projected onto three lines, each corresponding to a dimension of OD pairs, time slots, or days, we can easily divide each line into buckets of equal width, and use the bucket center to facilitate the finding of anchor points that can well represent data mapped to the same bucket. Selecting anchor points from each bucket also helps to evenly distribute them across different regions.
- **Construction of sub-tensors around properly selected anchor points.** To reduce the computation cost and increase the data recovery accuracy, we further propose an anchor-point selection algorithm, taking into account the density of samples already taken and distances of anchors selected.
- **Novel encoding of sparse traffic data.** We propose to represent the monitoring data of OD pairs, time slots, and days with novel codes under factorized basis matrices to cope with the challenge of calculating similarity/distance along each of these dimensions in the presence of partial measurements of the tensor.
- **Similarity-sensitive local tensor completion and data fusion.** We propose a similarity-sensitive local tensor completion algorithm and a data fusion algorithm for more accurate missing data recovery, taking advantage of similarity calculation based on the encoded data.
- **Extensive experiment based on real traffic traces.** We implement the proposed LTC using two real traffic flow traces (Abilene [11] and GÈANT [12]). Compared with peer tensor completion algorithms, LTC can achieve much higher data recovery accuracy.

The rest of the paper is organized as follows. Section II presents the related work. The preliminaries of tensor are presented in Section III. We present our problem and solution overview in Section IV. We present our algorithms on the calculation of candidate anchor point, LSH table building, sub-tensor extraction and completion, and data fusion in Section V, Section VI, Section VII, and Section VIII, respectively. Finally, we implement the proposed LTC and evaluate the performance using real traffic trace data in Section IX, and conclude the work in Section X.

II. Related work

There is rapid progress of sparse representation techniques. Following the compressive sensing [13] and matrix completion [14]–[16], tensor completion [17]–[20] has attracted lots of research interests recently. To infer the unavailable/missing data, these schemes usually employ both the low rank property of data and the information of partial observed entries to provide the best estimation of the whole data. These techniques have been widely used in recommendation systems [21]–[23], social network [24]–[26], disease detection [27], [28] and network data gathering and analysis [29]–[34].

Compared with matrix-based completion algorithm, the tensor-based approach can better handle the missing data in applications. Tensor completion algorithms [17]–[20] capture the global data structure for recovering the missing data via a high-order CANDECOMP PARAFAC (CP) decomposition [35], [36] and Tucker decomposition [37]. Some typical tensor completion algorithms are CP_{nmu} [18], CP_{opt} [19], CP_{wopt} [18], CP_{als} [38], and $Tucker_{als}$ [39]. CP_{nmu} estimates the best rank-R CP model of a tensor with nonnegative constraints on the factors, using the Lee and Seung multiplicative updates from its NMF algorithm. CP_{opt} fits the CP model to incomplete data sets by solving a least-square optimization problem ([19] $Eq.(2)$) with a gradient-based optimization approach. CP_{wopt} is different from CP_{opt}, CP_{wopt} (CP Weighted Optimization) addresses the problem of fitting the CP model to incomplete data sets by solving a weighted least squares problem ([18], $Eq.(2)$). CP_{als} and $Tucker_{als}$ fit the CP model and the Tucker model to incomplete data sets by solving an alternating least-square problem, respectively. Besides CP decomposition and Tucker decomposition, some recent work [40]–[43] study the low-tubal-rank tensor model and low-tubal-rank tensor completion. Some recent studies [32], [44] have modeled the traffic matrices of different time slots/days as a tensor to recover

the missing data through tensor completion.

However, existing tensor completion algorithms commonly assume the rank of the whole tensor is low and attempt to fit a single model with data of the entire tensor through an optimization. These approaches cannot benefit from lower ranks of sub-tensors with closer data correlations for higher data recovery accuracy.

Some recent studies show that matrix data of many applications have the local properties [45]–[47] and can form some local sub-matrices with higher correlation. Well exploiting these properties can more accurately infer the missing data. For example, [47] has demonstrated that network traffic sub-matrices typically show very low effective rank if the elements of sub-matrices are highly similar. Thanks for this property, recovering missing data through sub-matrices instead of the global matrix can achieve better performance.

Inspired by these local properties hidden in the matrix, we verify that modeling traffic data as a tensor still retain local low rank attributes. Specially, we notice that a subset of OD pairs may have more similar end-to-end traffic behaviors in some time slots (i.e., office hours) and days (i.e., week day). Through experiments on real traffic data set, we demonstrate that a sub-tensor built with such a subset of OD pairs/time slots/days has a lower rank. To well utilize the local data feature for more accurate data recovery, we propose a novel localized tensor completion model where we propose several novel techniques including sub-tensor building with intelligent selection of anchor points, novel encoding of sparse traffic data, and similarity-sensitive local tensor completion and data fusion.

III. Preliminaries

In this paper, scalars are denoted by lowercase letters $(a, b, \cdots)$, vectors are written in boldface lowercase $(\mathbf{a}, \mathbf{b}, \cdots)$, and matrices are represented with boldface capitals $(\mathbf{A}, \mathbf{B}, \cdots)$. Higher-order tensors are written as calligraphic letters $(\mathcal{X}, \mathcal{Y}, \cdots)$. The elements of a tensor are denoted by its symbolic name with indexes in subscript. For example, the ith entry of a vector $\mathbf{a}$ is denoted by a_i, the element (i, j) of a matrix $\mathbf{A}$ is denoted by a_{ij}, and the element (i, j, k) of a third-order tensor $\mathcal{X}$ is denoted by x_{ijk}. Some preliminaries in this paper can be found in [10], [48], [49].

Definition 1. *A* tensor *is a multidimensional array, and is a higher-order generalization of a vector (first-order tensor) and a matrix (second-order tensor). An N-way or Nth-order tensor (denoted as $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$) is an element of the tensor product of N vector spaces, where N is the order of $\mathcal{A}$, also called way or mode.*

The element of $\mathcal{A}$ is denoted by $a_{i_1,i_2,\cdots,i_N}$, $i_n \in \{1, 2, \cdots, I_n\}$ with $1 \le n \le N$.

Definition 2. *Slices are two-dimensional sub-arrays, defined by fixing all indexes but two.*

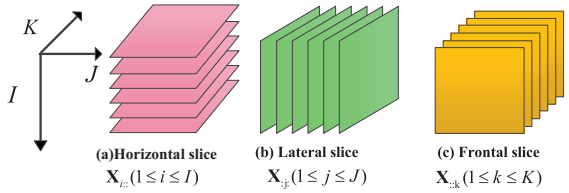


Fig. 1. A 3-way tensor $\mathcal{X}$ has horizontal, lateral and frontal slices, which are denoted by $\mathbf{X}_{i::}$, $\mathbf{X}_{:j:}$ and $\mathbf{X}_{::k}$, respectively..

Definition 3. *The outer product of two vectors $\mathbf{a} \circ \mathbf{b}$ is the matrix defined by: $(\mathbf{a} \circ \mathbf{b})_{ij} = a_i b_j$.*

Definition 4. *The outer product $\mathcal{A} \circ \mathcal{B}$ of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{N_1}}$ and a tensor $\mathcal{B} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_{N_2}}$ is the tensor of the order $N_1 + N_2$ defined by*

$$(\mathcal{A} \circ \mathcal{B})_{i_1,i_2,\cdots,i_{N_1},j_1,j_2,\cdots,j_{N_2}} = a_{i_1,i_2,\cdots,i_{N_1}} b_{j_1,j_2,\cdots,j_{N_2}} \quad (1)$$

for all values of the indexes.

Since vectors are first-order tensors, the outer product of three vectors $\mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$ is a tensor given by:

$$(\mathbf{a} \circ \mathbf{b} \circ \mathbf{c})_{ijk} = a_i b_j c_k \quad (2)$$

for all values of the indexes.

Definition 5. *A 3-way tensor $\mathcal{X}$ is a rank one tensor if it can be written as the outer product of three vectors, i.e. $\mathcal{X} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$.*

Definition 6. *The rank of a 3-way tensor is the minimal number of rank one tensors, that generate the tensor as their sum, i.e. the smallest R, such that $\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$.*

Definition 7. *The idea of CANDECOMP/PARAFAC (CP) decomposition is to express a tensor as the sum of a finite number of rank one tensors. A 3-way tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ can be expressed as*

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \quad (3)$$

with an entry calculated as

$$x_{ijk} = \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr} \quad (4)$$

where $R > 0$, a_{ir}, b_{jr}, c_{kr} are the i-th, j-th, and k-th entry of vectors $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, and $\mathbf{c}_r \in \mathbb{R}^K$, respectively.

By collecting the vectors in the rank one components, we have tensor $\mathcal{X}$'s factor matrices $\mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_R] \in \mathbb{R}^{I \times R}$, $\mathbf{B} = [\mathbf{b}_1, \cdots, \mathbf{b}_R] \in \mathbb{R}^{J \times R}$, and $\mathbf{C} = [\mathbf{c}_1, \cdots, \mathbf{c}_R] \in \mathbb{R}^{K \times R}$. Using the factor matrices, we can rewrite the CP decomposition as follows.

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \quad (5)$$

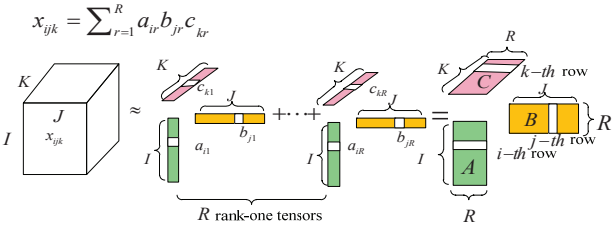


Fig. 2. CP decomposition of three-way tensor as sum of R outer products (rank one tensors). CP decomposition can be written as a triplet of factor matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, i.e, the r-th column of which contains $\mathbf{a}_r$, $\mathbf{b}_r$, and $\mathbf{c}_r$, respectively. The entry x_{ijk} can be calculated as the sum of the product of the entries of the i-th row of the matrix $\mathbf{A}$, the j-th row of the matrix $\mathbf{B}$, and the k-th row of the matrix $\mathbf{C}$.

Fig.2 illustrates the CP decomposition. In this paper, we design our LTC algorithm based on CP decomposition.

IV. Problem and Solution Overview

We first formulate the traffic data recovery problem, and then present the solution overview.

A. Problem formulation

Based on the analyses of real traffic trace, our recent work on tensor completion [29], [30], [44] reveals that the traffic data have the features of temporal stability, spatial correlation, and periodicity. To fully exploit these traffic features for accurate traffic data recovery, following [29], [30], [44], in this paper, we model the traffic data as a 3-way tensor $\mathcal{M} \in \mathbb{R}^{I \times J \times K}$ (Fig.3), where K corresponds to the number of origin and destination (OD)

pairs in the network, and there are J days to consider with each day having I time intervals. For the Abilene trace [11], $I = 288$, $J = 168$, and $K = 144$. Our LTC scheme, however, is general and does not depend on how the traffic tensor is modeled.

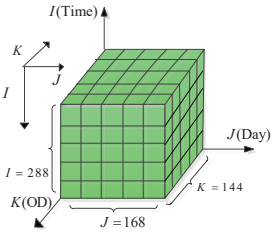


Fig. 3. 3-way traffic tensor.

As partial OD pairs are often monitored to reduce the measurement load and also there are an unavoidable data losses upon severe communication conditions, $\mathcal{M}$ is generally an incomplete tensor. If there are no traffic data between a pair of nodes in a given time interval, it leaves the corresponding entry in $\mathcal{M}$ empty. Let Ω be the set of indices of the sample entries in $\mathcal{M}$.

The traffic recovery problem is to obtain all data entries of the tensor $\mathcal{M}$ based on its measurement samples through the tensor factorization. With the factorization based on the CP decomposition, we would like to find the factor matrices $\mathbf{A} \in \mathbb{R}^{I\times R}$, $\mathbf{B} \in \mathbb{R}^{J\times R}$, and $\mathbf{C} \in \mathbb{R}^{K\times R}$ to approximate the tensor $\mathcal{M}$ with the minimum recovery loss:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \quad \sum_{(i,j,k)\in\Omega} \left([\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]_{i,j,k} - m_{i,j,k} \right)^2 \tag{6}$$

where $[\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]_{i,j,k}$ and $m_{i,j,k}$ are the recovered entry and the observed entry at (i,j,k). $[\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]_{i,j,k} - m_{i,j,k}$ is the recovery loss at (i,j,k). After finding three factor matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, the original traffic tensor can be recovered by

$$\hat{\mathcal{M}} = [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!] \tag{7}$$

B. Solution overview

The tensor completion problem in Eq.(6) is established based on the assumption that the rank of the overall tensor is low. This global low-rank assumption serves as the base for traditional tensor completion algorithms to fit a single model with the overall tensor data through the global optimization.

In network fields, a subset of OD pairs may have more close end-to-end traffic behaviors in part of time slots (i.e., working hours) of certain days (i.e., week days). Obviously, samples taken from these partial OD pairs have higher correlation and would form a sub-tensor of lower rank. Simply applying the global optimization over the whole traffic tensor can not benefit from the higher local correlation among data.

In order to better exploit the local correlations from the three corresponding domains for more accurate missing data recovery, we propose a novel localized tensor completion model, where we first reorganize data from a large tensor to build a set of sub-tensors and then complete and fuse the data recovered from sub-tensors to form the completed large tensor. Fig.4 shows the basic three steps of our solution:

1) Choose some anchor points.
2) Build local sub-tensors (each corresponding to an anchor point) by selecting samples that are within the distance d of the anchor point, where the distance is determined by the similarity of data rather than the Euclidean distance of indexes.
3) After inferring missing data in sub-tensors, we fuse the recovered data of sub-tensors to form the original large tensor data taking consideration of the similarity of data. In Fig.4, missing data entry 1 is finally recovered by fusing its recovered values from two sub-tensors which cover it. Similarly, the value of missing data entry 2 is fused by the recovered values from the three sub-tensors.

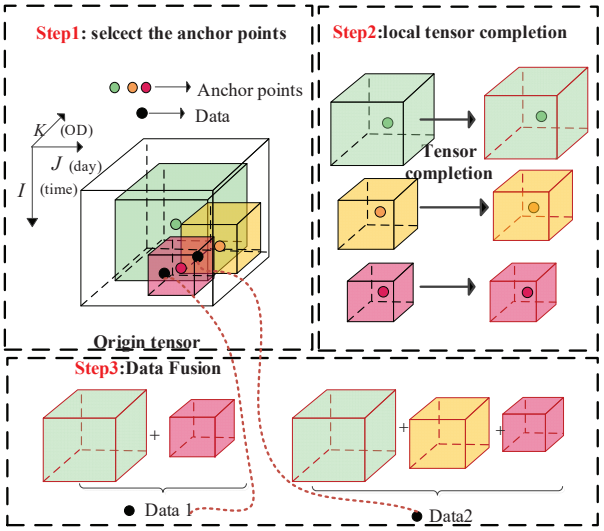


Fig. 4. Solution overview

V. Selection of Anchor Points

To build the low-rank sub-tensor, similar OD pairs from a subset of time slots and days need to be grouped together. Anchor points are selected to facilitate building the sub-tensors in our solution.

A straightforward way is to randomly select a set of OD-time-day triplets as anchor-points, then take the sample entries within a distance d of the anchor-points to construct a local sub-tensor. As discussed in introduction, measurement samples may not be taken uniformly but with different sampling rate based on the traffic patterns and operational needs. In addition, as the data in the tensor are arranged logically, even though samples are taken uniformly (as shown in Fig.5(a)), some data may have higher similarity and can be reorganized into a sub-tensor. We can regroup data to form different sub-tensors (Fig.5(b)), with data from the same sub-tensor having higher correlation and data from different ones having lower correlation. We can take advantage of higher correlation among data from sub-tensors and sub-tensor completion to more accurately recover the data.

If a sub-tensor is formed with data of higher correlation, it will improve the accuracy of missing data inference. To achieve the goal, we will reorganize data samples according to the data correlation to form different sub-tensors.

Measurement samples may not be taken uniformly, especially when the sampling is performed dynamically following the traffic patterns or operational needs. Even when measurement samples are taken uniformly, after the data reorganization based on correlation, the samples may distribute unevenly across the data region.

Fig.5(a) shows that the original sample data are uniformly distributed in a domain of monitoring. After reorganizing the data according to their correlation, the samples are distributed unevenly in Fig.5(b).

Randomly selecting anchor points may lead to uneven local sub-tensor distribution. Some regions with dense samples may not be well covered by sub-tensors while the regions with sparse samples may be over-covered. In Fig.5(c), the region B with dense samples is covered by one sub-tensor, while the region A with sparse samples is covered by a number of sub-tensors. Selecting anchor

points in the sparse area will make a sub-tensor built with very few samples and difficult to be accurately recovered. Moreover, if many anchor points are selected in a small area, it will build redundant sub-tensors, and redundant tensor recovery will result in high computational overhead.

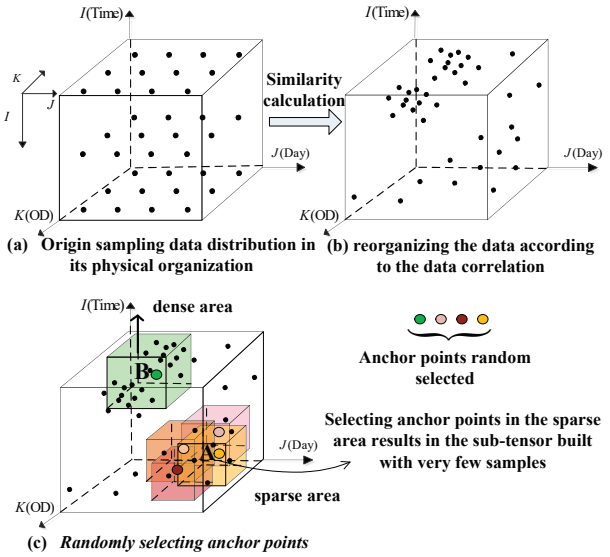


Fig. 5. Randomly selecting anchor-points results in uneven sub-tensor distribution.

To find anchor points that can well represent the correlations of data in a local tensor and distribute evenly inside the big tensor, we propose an algorithm following three steps (in Fig.6):

- Step 1: We build three LSH tables to reorder time slots, days, and OD pairs into X, Y, and Z groups based on the data correlation in each dimension. With good properties brought by our well designed LSH hash function (to be presented in Section VI), each group includes similar time slots, days, and OD pairs.
- Step 2: With group centers selected along three dimensions $\{a_1, a_2, \cdots, a_X\}$, $\{b_1, b_2, \cdots, b_Y\}$, and $\{c_1, c_2, \cdots, c_Z\}$, we can obtain the candidate anchor-points m_{a_i,b_j,c_k} by combining group centers a_i, b_j, and c_k ($1 \le i \le X, 1 \le j \le Y, 1 \le k \le Z$).
- Step 3: To reduce the computation cost and increase the data recovery accuracy, we propose an anchor-point selection algorithm (in Section VIII-A) to select appropriate anchor-points from the candidate anchor-points to form the sub-tensors, taking into account the density of samples already taken and distances of anchors selected. Our selection algorithm can avoid building multiple sub-tensors in one sub-area to reduce redundant computations.

In Fig.6 (b), candidate anchor points are found with the combination of group centers of different domains. For example, the candidate anchor-point m_{a_2,b_Y,c_2} is built by combining group centers a_2, b_Y, and c_2.

VI. Exploiting LSH to Quickly Find Candidate Anchor-Points

As shown in Section V, to find candidate anchor points, we first group similar time slots, days, and OD pairs together based on LSH tables. Although the k-means algorithm is often applied to cluster items, it runs iteratively with the need of calculating distances among items in each iteration step, which may incur a high computation cost. To reduce the computation cost, we propose

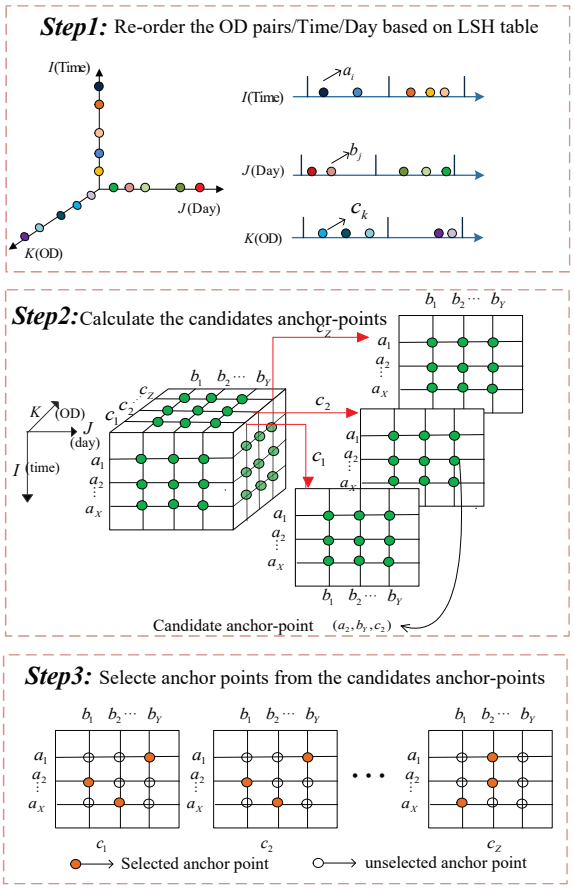


Fig. 6. Candidate anchor points

to exploit LSH function to quickly reorder the data for quickly grouping similar data. In this section, we will show that our LSH table has following good features:

- Only requiring hash calculations, our LSH tables reorder OD pairs in a fast and effective way.
- With data projected to different lines according to the dimensions of OD pair, time slot and day, we can divide projected data on each line into buckets of equal distance to ensure even anchor point selection across the whole data region.
- The LSH tables provides a new indexing method that approximates the nearest neighbor query by placing OD pairs, time slots and days with closer correlations to close-by positions. This good properties can be further utilized to form sub-tensor with similar entries, as shown in Section VII. As both the sample density and anchor distance are considered, we will not select multiple anchor points in one sub-area to reduce the redundant computation.

A. Challenge in distance calculation

Although the solution in Section V can help find candidate anchor-points, the LSH based reordering algorithm (in Section VI-C) and further the sub-tensor building algorihm (in Section VII) depend on the distance calculation. In order to reduce the measurement overhead, the traffic tensors are often sparse with only a subset of entries having sample data. This makes the distance calculation a challenge.

As shown in Fig.7, a frontal slice, a lateral slice, and a horizontal slice of the traffic tensor $\mathcal{X}$ record respectively the data of an OD pair, the data in a day, and the data at a time slot. Intuitively, calculating distances in the domains of OD pairs, days, and time

slots can leverage frontal slices, lateral slices, and horizontal slices, respectively. For example, in Fig.7, to calculate the distance between time slots i and i', we can compare slices $X_{i::}$ and $X_{i'::}$ directly. However, as the traffic tensor is sparse with incomplete data, this straightforward way may fail. To address the challenge, we propose a novel strategy based on the encoding of slices.

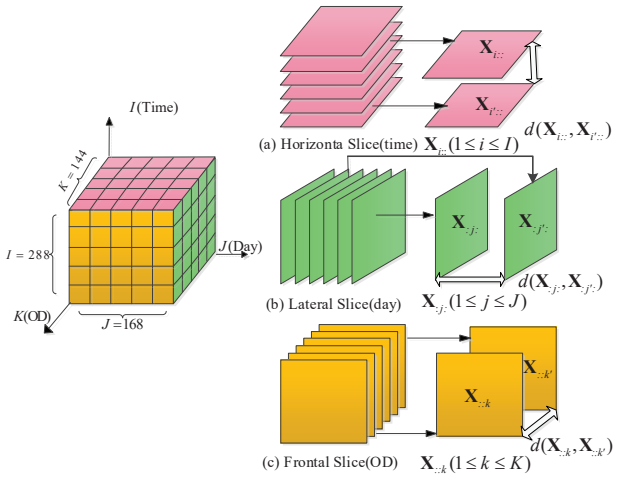


Fig. 7. Directly calculate distance using slices

B. Encoding tensor slices under the bases of factorized matrices

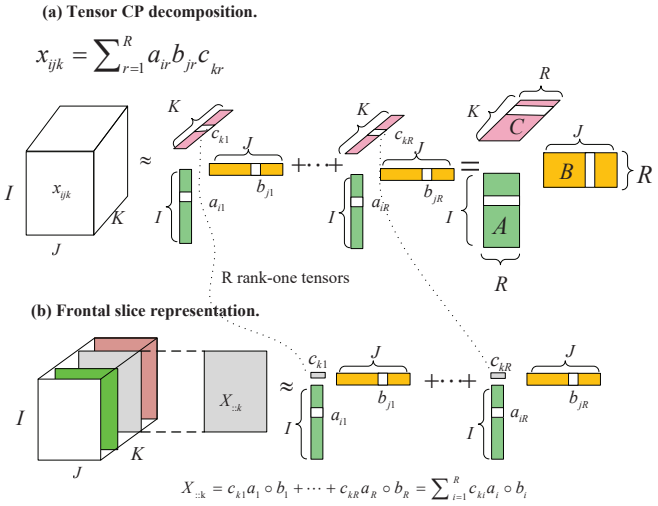


Fig. 8. The relationship between CP decomposition and frontal slice representation.

In Fig.8(a), according to (5), the CP decomposition of a 3-way tensor $\mathcal{X}$ can be written as follows.

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \tag{8}$$

where matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$, $\mathbf{B} \in \mathbb{R}^{J \times R}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$ are the factor matrices. In Fig.8(b), a frontal slice $\mathbf{X}_{::k}$ can be written as

$$\mathbf{X}_{::k} = c_{k1}\mathbf{a}_1 \circ \mathbf{b}_1 + \cdots + c_{kR}\mathbf{a}_R \circ \mathbf{b}_R = \sum_{i=1}^{R} c_{ki}\mathbf{a}_i \circ \mathbf{b}_i. \tag{9}$$

where $c_{k1}, c_{k2}, \cdots, c_{kR}$ are the entries of the k-th row of the factor matrix $\mathbf{C}$. Similarly, a lateral slice $\mathbf{X}_{:j:}$ and a horizontal slice $\mathbf{X}_{i::}$ can be written in (10) and (11).

$$\mathbf{X}_{:j:} = b_{j1}\mathbf{a}_1 \circ \mathbf{c}_1 + \cdots + b_{kR}\mathbf{a}_R \circ \mathbf{c}_R = \sum_{i=1}^{R} b_{ji}\mathbf{a}_i \circ \mathbf{c}_i. \tag{10}$$

where $b_{j1}, b_{j2}, \cdots, b_{jR}$ are the entries of the j-th row of the factor matrix $\mathbf{B}$.

$$\mathbf{X}_{i::} = a_{i1}\mathbf{b}_1 \circ \mathbf{c}_1 + \cdots + a_{iR}\mathbf{b}_R \circ \mathbf{c}_R = \sum_{j=1}^{R} a_{ij}\mathbf{b}_j \circ \mathbf{c}_j. \tag{11}$$

where $a_{i1}, a_{i2}, \cdots, a_{iR}$ are the entries of the i-th row of the factor matrix $\mathbf{A}$.

Equation (11) shows that each horizontal slice $\mathbf{X}_{i::}$ can be expressed as a superposition of R rank-1 matrices $\mathbf{b}_i \circ \mathbf{c}_i$ $(1 \le i \le R)$. That is, the traffic data $\mathbf{X}_{i::}$ at a time slot i is represented by the linear combination of R rank-1 matrices $\mathbf{b}_i \circ \mathbf{c}_i$.

These rank-1 matrices can thus be called horizontal basis matrices, and parameters $a_{i1}, a_{i2}, \cdots, a_{iR}$ are coordinates of the horizontal slice $\mathbf{X}_{i::}$ under this new basis and together they are considered as a code to represent the traffic data at the slot k. Given two horizontal slices $\mathbf{X}_{i::}$ and $\mathbf{X}_{i'::}$, if the traffic data at time slots i and i' are close, the codes under the horizontal basis, $a_{i1}, a_{i2}, \cdots, a_{iR}$ (i.e., $\mathbf{a}_{\mathbf{i}:}$, the i-th row of the factor matrix $\mathbf{A}$) and $a_{i'1}, a_{i'2}, \cdots, a_{i'R}$ (i.e., $\mathbf{a}_{\mathbf{i}':}$, the i'-th row of the factor matrix $\mathbf{A}$) should also be close to each other.

Directly calculating the distance between slices corresponding to time slots, days, and OD pairs requires each matrix to have complete data. Instead, we encode slices and calculate the distances among their codes. Denoting $\mathbf{a}_{\mathbf{i}:}$, $\mathbf{b}_{\mathbf{j}:}$, and $\mathbf{c}_{\mathbf{k}:}$ as the codes of the time slot i, day j, and OD pair k. Obviously, they are the rows in the factor matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. We can apply

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \quad \sum_{(i,j,k)\in\Omega} \left([\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]_{i,j,k} - m_{i,j,k} \right)^2 \tag{12}$$

to the incomplete monitoring tensor to obtain the three factor matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ by approximating the monitoring tensor through CP-decomposition with the minimum recovery loss.

As discussed in the introduction (Section I) and also verified in the evaluation (Section IX-A2), lower rank sub-tenors exist in the large global tensor. Therefore, using a completion algorithm that works based on the hypothesis of a global low rank structure may not yield good recovery performance. The missing data will be more accurately recovered through the sub-tensor taking advantage of higher correlation among data. Therefore, we don't directly recover the global tensor using the factor matrices trained by CP decomposition on the global tensor. However, to conquer the challenge of distance calculation brought by the sparse data, we still need to train these factor matrices to facilitate encoding time slot, day, and OD pair and finding the anchor-points for forming the sub-tensors with data of higher similarity.

C. Build LSH tables to reorder time, day, and OD pair

We need three LSH tables, one for time slots, one for days, and the other for OD pairs. We take the LSH table for OD pairs as an example to illustrate how our LSH algorithm re-orders the OD pairs with low computation cost.

Based on the principle of LSH [50], if two data points are close together, they will remain close after a "projection" operation. To group similar OD pairs together, we apply LSH to the code of OD pair to map OD pairs into the LSH table with the following two steps:

- ***Step 1: Projecting OD pairs to a line***. Given an OD pair k with its code being $\mathbf{c}_{\mathbf{k}:} \in \mathbb{R}^R$ $(1 \le k \le K)$, we define the LSH hash function as

$$h_{\mathbf{a}}(\mathbf{c}_{\mathbf{k}:}) = \mathbf{a} \cdot \frac{\mathbf{c}_{\mathbf{k}:}}{\|\mathbf{c}_{\mathbf{k}:}\|} \tag{13}$$

  Eq. (13) applies a scalar dot operation to project the OD pair k to a point on a straight line whose direction is identified by

$\mathbf{a}$. $\mathbf{a}$ is a vector with its components selected at random from a Gaussian distribution, for example $\mathcal{N}(0,1)$. In this paper, we call this line the *projection line.*

- ***Step 2: Building the LSH table***. We denote the first projected value and the last projected value on the line as p_s and p_e, respectively. Given the total number of groups to divide, Z, we partition the projection line between p_s and p_e into Z parts to build the hash table, with the bucket width of the table being $\frac{p_e-p_s}{Z}$.

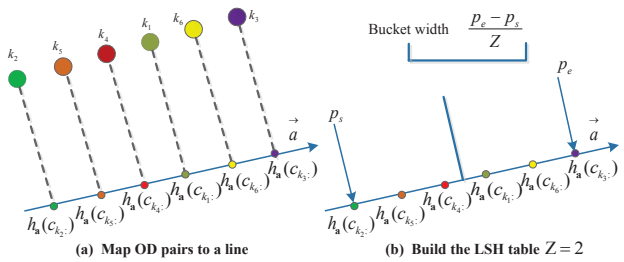


Fig. 9. An example to illustrate how to build the LSH table

Fig. 9(a) shows geometrically the projection process of Step 1. After applying the LSH hash function (13) to OD pairs k_1, k_2, k_3, k_4, k_5, k_6, these OD pairs are projected along a line. In Fig.9(b), the first projected value and the last projected value are $p_s = h_{\mathbf{a}}(\mathbf{c_{k_2}}:)$ and $p_e = h_{\mathbf{a}}(\mathbf{c_{k_3}}:)$, respectively. If we assume $Z = 2$, all projected values are clustered into two hash buckets (groups), with the width of each bucket being $\frac{p_e-p_s}{2}$.

D. Good property of the bucket center

We denote the center of a hash bucket in the OD pair LSH table as y_0, which may be projected from $\mathbf{c_{k_0}}$ that exists either physically or virtually. If $p_s = p_e$, all $\mathbf{c_{k}}: \in \mathbb{R}^R\,(1 \le k \le K)$ are colinear and we can hardly build the hash table, otherwise, the following Theorem show that $\mathbf{c_{k_0}}:$ has good properties to represent OD pairs in the hash bucket.

Theorem 1. *The group center $\mathbf{c_{k_0}}:$ has the following two properties:*

- **Property 1**. *For any an OD pair k_j that is close to the OD pair k_0, there is a high probability P_1 that these two OD pairs fall into the same bucket. That is, if $||\frac{\mathbf{c_{k_0}}:}{\|\mathbf{c_{k_0}}:\|}, \frac{\mathbf{c_{k_j}}:}{\|\mathbf{c_{k_j}}:\|}||_2 \le r$, $Pr\left[\left|h_{\mathbf{a}}(\mathbf{c_{k_j}}:) - h_{\mathbf{a}}(\mathbf{c_{k_0}}:)\right| \le \frac{p_e-p_s}{2Z}\right] \ge P_1$.*
- **Property 2**. *For any an OD pair k_j that is far from OD pair k_0, there is a low probability $P_2 < P_1$ that they fall into the same bucket. That is, if $||\frac{\mathbf{c_{k_0}}:}{\|\mathbf{c_{k_0}}:\|}, \frac{\mathbf{c_{k_j}}:}{\|\mathbf{c_{k_j}}:\|}||_2 \ge cr$, $Pr\left[\left|h_{\mathbf{a}}(\mathbf{c_{k_j}}:) - h_{\mathbf{a}}(\mathbf{c_{k_0}}:)\right| \le \frac{p_e-p_s}{2Z}\right] \le P_2$.*

Above, r is a search radius, c $(c > 1)$ is a scaling factor, $||\frac{\mathbf{c_{k_0}}:}{\|\mathbf{c_{k_0}}:\|}, \frac{\mathbf{c_{k_j}}:}{\|\mathbf{c_{k_j}}:\|}||_2$ is the Euclidean distance between OD pairs k_j and k_0, and $Pr\left[\left|h_{\mathbf{a}}(\mathbf{c_{k_j}}:) - h_{\mathbf{a}}(\mathbf{c_{k_0}}:)\right| \le \frac{p_e-p_s}{2Z}\right]$ is the collision probability of hashing OD pair k_j and k_0 to the same bucket.

Proof. In the domain of hash function, given two OD pairs k_j and k_0, if the pairs are mapped to the same bucket, there is a collision in hashing. We analyze the properties of the hash table through the calculation of the collision probability, i.e., $Pr\left[\left|h_{\mathbf{a}}(\mathbf{c_{k_j}}:) - h_{\mathbf{a}}(\mathbf{c_{k_0}}:)\right| \le \frac{p_e-p_s}{2Z}\right]$.

According to [51], we know that Gaussian distribution $\mathcal{N}(0,1)$ is 2-stable distribution. Thus the distribution of $\mathbf{a} \cdot \frac{\mathbf{c_{k_j}}:}{\|\mathbf{c_{k_j}}:\|} - \mathbf{a} \cdot \frac{\mathbf{c_{k_0}}:}{\|\mathbf{c_{k_0}}:\|}$ follows that of dX, where $d = ||\frac{\mathbf{c_{k_0}}:}{\|\mathbf{c_{k_0}}:\|}, \frac{\mathbf{c_{k_j}}:}{\|\mathbf{c_{k_j}}:\|}||_2$ is the distance between OD pairs k_j and k_0 and X is a random variable drawn from $\mathcal{N}(0,1)$. Let $f(x)$ be the probability density function (PDF) of $\mathcal{N}(0,1)$, i.e., $f(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2}$. The probability that the OD pairs k_j and k_0 collide in our LSH hash table is computed as follows:

$$\begin{aligned} &Pr(d) = Pr\left[\left|h_{\mathbf{a}}(\mathbf{c_{k_j}}:) - h_{\mathbf{a}}(\mathbf{c_{k_0}}:)\right| \le \tfrac{p_e-p_s}{2Z}\right] \\ &= Pr\left[|dX| \le \tfrac{p_e-p_s}{2Z}\right] = Pr\left[-\tfrac{p_e-p_s}{2dZ} \le X \le \tfrac{p_e-p_s}{2dZ}\right] \\ &= \int_{-\frac{p_e-p_s}{2dZ}}^{\frac{p_e-p_s}{2dZ}} f(t)dt \end{aligned} \tag{14}$$

Obviously, we have

$$Pr(d) = 1 - 2norm(-\frac{p_e - p_s}{2dZ}) \tag{15}$$

where $norm(x) = \int_{-\infty}^{x} f(t)dt$ is simply the cumulative distribution function (CDF) of $\mathcal{N}(0,1)$, which increases monotonically as x increases. For a fixed $\frac{p_e-p_s}{Z}$, $norm(-\frac{p_e-p_s}{2dZ})$ increases monotonically as d increases. Accordingly, $Pr(d)$ decreases monotonically as d increases. In another word, $Pr(d)$ increases as d becomes smaller.

Replacing d in Eq(14) with r and cr respectively, we obtain $P_1 = Pr(r)$ and $P_2 = Pr(cr)$. As $r < cr$ when $c > 1$, we have $P_2 < P_1$. □

Finding the group center with our LSH table brings the above two good properties, and our method can map similar OD pairs with a shorter distance to the group center into the same hash bucket with a high probability. Therefore, the group center can be a representative point of the OD pairs in the bucket.

For the bucket center y_0, its corresponding $\mathbf{c_{k_0}}:$ may be an OD pair that physically exists in the tensor or a virtual point without corresponding to any OD pair. In the latter case, we set the OD pair whose projected value is the closest to y_0 as the group center.

In Eq.(13), we project an OD pair using its normalized code instead of its original code. Moreover, in Theorem 1, the distance is also calculated using the normalized code. This is because that we care more about the codes' directions (thus the angles among OD pairs) than their absolute values for more accurate tensor recovery.

According to *Property 2* in Theorem 1, with one LSH hash function, two OD pairs k_j and k_0 have up to P_2 probability of being hashed into the same bucket when their distance $||\frac{\mathbf{c_{k_0}}:}{\|\mathbf{c_{k_0}}:\|}, \frac{\mathbf{c_{k_j}}:}{\|\mathbf{c_{k_j}}:\|}||_2 \ge cr$. To help buffer similar OD pairs into the same bucket while reducing the probability of hashing uncorrelated ones to the same hash bucket to create the collision, instead of using a single hash function, we could apply multiple hash functions. With n hash functions, we have the hash index of $\{h_{\mathbf{a_1}}(\mathbf{c_{k_j}}:), h_{\mathbf{a_2}}(\mathbf{c_{k_j}}:), ..., h_{\mathbf{a_n}}(\mathbf{c_{k_j}}:)\}$. Straight forwardly, this would require n hash tables to store the OD vectors, and thus requires $O(n)$ to store the hash index and also $O(n)$ to search for a close-by neighbor. Without a physical bucket to hold all items closely-related, this storage and search scheme is difficult to find the cluster centers thus anchor-points.

To reduce the storage and query cost, and more importantly, to support the quick finding of the cluster center, inspired by MIT's E2LSH [52], we will instead use a super index which involves n hash functions:

$$H(\mathbf{c_{k_j}}:) = \sum\nolimits_{i=1}^{n} r_i h_{\mathbf{a_i}}(\mathbf{c_{k_j}}:) \tag{16}$$

where r_i is a random integer. With this super index, two distant OD k_j and k_0 have the same super index under Eq.(16) only if all n hash indexes are in the same bucket under the corresponding single hash scenario, that is, $\left|h_{\mathbf{a_i}}\left(\mathbf{c_{k_j}}:\right) - h_{\mathbf{a_i}}\left(\mathbf{c_{k_0}}:\right)\right| \le \frac{p_{e_i}-p_{s_i}}{2Z}$ where p_{s_i}

and p_{e_i} are the first projected value and the last projected value on the projection line defined by $\mathbf{a_i}$.

According to *Property 2* in Theorem 1, using a single hash function, the probability for two distant OD pairs with the distance $||\frac{\mathbf{c_{k_0:}}}{\|\mathbf{c_{k_0:}}\|}, \frac{\mathbf{c_{k_j:}}}{\|\mathbf{c_{k_j:}}\|}||_2 \geq cr$ to be hashed into the same bucket is not larger than P_2. Thus, the probability in Eq.(16) is not larger than $(P_2)^n$. Obviously, we have $(P_2)^n < P_2$ because $0 < P_2 < 1$. That is, the use of a group of hash functions helps to reduce the probability of hashing uncorrelated OD pairs to the same bucket to create the collision. In this paper, we set $n = 10$.

E. Scalability to handle real time data set

In our method, for each data item (can be time, day, OD pair) that needs to be reordered, we obtain their codes through CP decomposition, then apply LSH hash function on the codes to project the data on the projection lines. When new data arrive, we can apply the incremental CP proposed in our recent study [53] to obtain the new codes corresponding to the new data, without the change of the code of the history data. Therefore, to update the LSH table, we only need to apply the LSH function on the new codes to add the new data item. The projections on the old codes corresponding to the history data will not change. So the computation cost can remain low to handle real time data set with sequential incoming new data.

VII. Sub-tensor extraction and completion

For more accurate missing data recovery, a sub-tensor should contain data entries from similar OD pairs with related time slots and days. As discussed in introduction, we can not build a sub-tensor with simple tensor partition as entries with adjacent indexes in the large tensor may not be similar.

In order to find the similarity between data points m_{ijk} and $m_{i'j'k'}$ in the tensor, we define their distance as

$$d(m_{ijk}, m_{i'j'k'}) = d(\mathbf{a_{i:}}, \mathbf{a_{i':}}) \times d(\mathbf{b_{j:}}, \mathbf{b_{j':}}) \times d(\mathbf{c_{k:}}, \mathbf{c_{k':}}) \tag{17}$$

where $d(\mathbf{a_{i:}}, \mathbf{a_{i':}}) = arccos\left(\frac{\langle \mathbf{a_{i:}}, \mathbf{a_{i':}} \rangle}{\|\mathbf{a_{i:}}\| \cdot \|\mathbf{a_{i':}}\|}\right)$, $d(\mathbf{b_{j:}}, \mathbf{b_{j':}}) = arccos\left(\frac{\langle \mathbf{b_{j:}}, \mathbf{b_{j':}} \rangle}{\|\mathbf{b_{j:}}\| \cdot \|\mathbf{b_{j':}}\|}\right)$, and $d(\mathbf{c_{k:}}, \mathbf{c_{k':}}) = arccos\left(\frac{\langle \mathbf{c_{k:}}, \mathbf{c_{k':}} \rangle}{\|\mathbf{c_{k:}}\| \cdot \|\mathbf{c_{k':}}\|}\right)$ are the angular distance between two time slots i and i', two days j and j', and two OD pairs k and k'.

After calculating the distance between items in each domain, we apply following equations to normalize the angular distance to the range $[0, 1]$:

$$d(\mathbf{a_{i:}}, \mathbf{a_{i':}}) = \frac{d(\mathbf{a_{i:}}, \mathbf{a_{i':}}) - \min(d(\mathbf{a_{i:}}, \mathbf{a_{i':}}))}{\max(d(\mathbf{a_{i:}}, \mathbf{a_{i':}})) - \min(d(\mathbf{a_{i:}}, \mathbf{a_{i':}}))} \tag{18}$$

$$d(\mathbf{b_{i:}}, \mathbf{b_{i':}}) = \frac{d(\mathbf{b_{i:}}, \mathbf{b_{i':}}) - \min(d(\mathbf{b_{i:}}, \mathbf{b_{i':}}))}{\max(d(\mathbf{b_{i:}}, \mathbf{b_{i':}})) - \min(d(\mathbf{b_{i:}}, \mathbf{b_{i':}}))} \tag{19}$$

$$d(\mathbf{c_{i:}}, \mathbf{c_{i':}}) = \frac{d(\mathbf{c_{i:}}, \mathbf{c_{i':}}) - \min(d(\mathbf{c_{i:}}, \mathbf{c_{i':}}))}{\max(d(\mathbf{c_{i:}}, \mathbf{c_{i':}})) - \min(d(\mathbf{c_{i:}}, \mathbf{c_{i':}}))} \tag{20}$$

In Eq.(18), Eq.(19), and Eq.(20), $\max(d(\mathbf{a_{i:}}, \mathbf{a_{i':}}))$, $\max(d(\mathbf{b_{i:}}, \mathbf{b_{i':}}))$, and $\max(d(\mathbf{c_{i:}}, \mathbf{c_{i':}}))$, are respectively the maximum angular distance between two time slots i and i', two days j and j', and two OD pairs k and k'. $\min(d(\mathbf{a_{i:}}, \mathbf{a_{i':}}))$, $\min(d(\mathbf{b_{i:}}, \mathbf{b_{i':}}))$, and $\min(d(\mathbf{c_{i:}}, \mathbf{c_{i':}}))$, are respectively the minimum angular distance between two time slots i and i', two days j and j', and two OD pairs k and k'. Using the normalized angular value, entry distance is obviously within the range $[0, 1]$ according to Eq.(17).

Given an anchor-point m_{a_t,b_t,c_t}, where a_t, b_t, and c_t are group centers from three different domains with $a_t \in \{a_1, a_2, \cdots, a_X\}$, $b_t \in \{b_1, b_2, \cdots, b_Y\}$, and $c_t \in \{c_1, c_2, \cdots, c_Z\}$, the sub-tensor can be built by selecting the entry m_{ijk} whose distance to m_{a_t,b_t,c_t} is less than h:

$$d(m_{ijk}, m_{a_t,b_t,c_t}) < h \tag{21}$$

Fig.10 illustrates how we build the sub-tensor around the anchor-point m_{a_t,b_t,c_t}.

As we have re-ordered the time slots, days, and OD pairs in LSH table, to find the entry m_{ijk} closest to m_{a_t,b_t,c_t}, the candidate entry index i, j, k can be easily found in the bucket that is the same or adjacent to the bucket of a_t, b_t, and c_t. Facilitated by the LSH table, the calculation cost of selecting tensor entries to build the sub-tensor can be largely reduced.

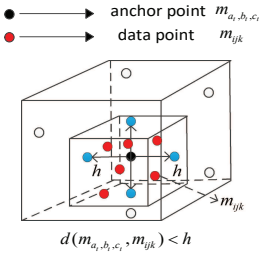


Fig. 10. The formation of a sub-tensor.

Rather than using the Euclidean distance, as the distance of two entries is defined based on the angular distances along three tensor dimensions, the sub-tensors built with the same h may have different sizes as shown in Fig.4, 5, and Fig.12. Formulating a sub-tensor this way helps to better capture the correlations hidden in the traffic data for more accurate missing data recovery.

For a sub-tensor with the anchor point m_{a_t,b_t,c_t}, the local tensor completion problem can be formulated as follows:

$$\min_{\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t} \sum_{(i,j,k) \in \Omega} K(m_{a_t,b_t,c_t}, m_{ijk})([\![\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t]\!]_{i,j,k} - m_{i,j,k})^2 \tag{22}$$

which gives a low-rank approximation for each neighborhood by minimizing the squared reconstruction error, weighted by the similarity of the sample point (m_{ijk}) to the anchor point (m_{a_t,b_t,c_t}). In (22), $\mathbf{A}_t$, $\mathbf{B}_t$, $\mathbf{C}_t$ denote the factor matrices of the sub-tensor, $K(m_{a_t,b_t,c_t}, m_{ijk})$ is a smoothing kernel that measures the similarity between the sample point m_{ijk} and the anchor point m_{a_t,b_t,c_t} as

$$K(m_{a_t,b_t,c_t}, m_{ijk}) = \begin{cases} (1 - d(m_{a_t,b_t,c_t}, m_{ijk})^2) & d(m_{a_t,b_t,c_t}, m_{ijk}) < h \\ 0 & otherwise \end{cases} \tag{23}$$

Using the normalized angular value (shown in Eq.(18), Eq.(19), and Eq.(20), the distance $d(m_{a_t,b_t,c_t}, m_{ijk})$ is within the range $[0, 1]$. Thus, the kernel value $K(m_{a_t,b_t,c_t}, m_{ijk})$ obviously is not a negative value.

We use the kernel function to convert the distance to similarity. Obviously, a larger distance will have smaller similarity thus smaller kernel value, while a smaller distance often results in a larger similarity thus larger kernel value. With the introduction of kernel function, entries with shorter distances to the anchor point will have larger kernel values thus higher weights in Eq.(22), thus can be more accurately recovered with smaller error $([\![\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t]\!]_{i,j,k} - m_{i,j,k})$.

VIII. Data fusion

After obtaining the candidate anchor points, a straightforward way of recovering the large tensor is to first recover all sub-tensors with each around an anchor point, then fuse the recovered data of all sub-tensors to obtain the data of the large tensor. Although promising, the computation cost is high. As a sample data point

may be contained in multiple sub-tensors, to reduce the computation cost, we need to carefully select anchor-points from all candidate ones.

A. Density and distance aware anchor point selection

To efficiently and accurately recover missing data, we need to select appropriate anchor-points from the candidate anchor-points to form the sub-tensors. We adopt two criteria to select the anchor-points: 1) each sub-tensor selected should contain more information to recover the missing data in the sub-tensor; 2) the sub-tensors selected should contain more information to recover the original large tensor.

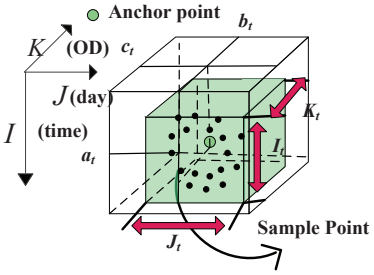


Fig. 11. Density of sub-tensor.

According to the tensor completion theory [10], given a low rank tensor, the sampling density directly impacts the performance of missing data recovery. Given a candidate anchor-point m_{a_t,b_t,c_t}, according to (21), samples having higher correlation with m_{a_t,b_t,c_t} are selected to group into the sub-tensor. Denoting the total number of samples selected in the sub-tensor as N_t, we define the sample density as the ratio of total number of samples in the sub-tensor to its size as

$$density(m_{a_t,b_t,c_t}) = \frac{N_t}{I_t \times J_t \times K_t} \tag{24}$$

where I_t, J_t, and K_t denote the number of distinct indices of selected samples in time slot domain, day domain, and OD domain respectively, and $I_t \times J_t \times K_t$ is the size of the sub-tensor.

Obviously, a sub-tensor with a higher density will have more data points, and thus contain more information to achieve higher data recovery accuracy. Moreover, to capture more diversified information hidden in different sub-tensors for more accurate data recovery in the large tensor, the distance among anchor points selected should also be large.

Considering both the sampling density and anchor distance, we propose to select anchor-points one by one until the total number of selected anchor-points reaches q. Specially, if we have selected l anchor-points, among all the candidate anchor-points left, we will select the anchor-point m_{a_i,b_i,c_i} that maximizes the following equation

$$\alpha \times \text{density}\left(m_{a_i,b_i,c_i}\right) + \frac{1-\alpha}{l}\sum_{p=1}^{l} d\left(m_{a_i,b_i,c_i}, m_{a_p,b_p,c_p}\right) \tag{25}$$

where the second item of the equation is the average distance between the anchor-point m_{a_i,b_i,c_i} and the l anchor-points selected, α is the adjustment coefficient to balance sampling density and distance with $0 \le \alpha \le 1$. In the simulation, we will study how α impacts the recovery performance. Fig.12 is an example to illustrate our anchor-point selection algorithm, where two anchor-points m_{a_1,b_1,c_1} and m_{a_2,b_2,c_2} have already been selected. Among the left un-selected candidate anchor-points m_{a_3,b_3,c_3}, m_{a_4,b_4,c_4}, and m_{a_5,b_5,c_5}, because m_{a_4,b_4,c_4} has the largest distance to the two anchor-points selected as well as the sub-tensor corresponding to m_{a_4,b_4,c_4} has the large sampling destiny, m_{a_4,b_4,c_4} is the next selected anchor-point.

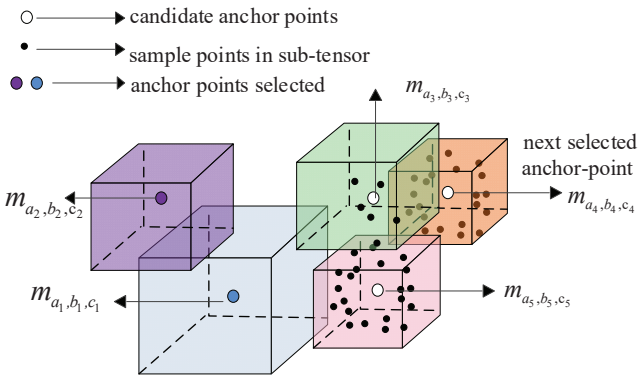


Fig. 12. Anchor-point selection.

B. Similarity sensitive data fusion

An tensor entry (i,j,k) may be contained in different sub-tensors. After recovering each sub-tensors, we need to fuse their results to form the final tensor. If we have q sub-tensors selected with the corresponding anchor points $m_{a_1b_1c_1}, m_{a_2b_2c_2}, \cdots, m_{a_qb_qc_q}$, we can find the entry of the final recovered large tensor through the following fusion operation:

$$\hat{m}_{i,j,k} = \sum_{t=1}^{q} \frac{K(m_{a_tb_tc_t}, m_{ijk})}{\sum_{s=1}^{q} K(m_{a_sb_sc_s}, m_{ijk})} [\![\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t]\!]_{i,j,k} \tag{26}$$

where $\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t$ are the factor matrices obtained in the sub-tensor corresponding to the anchor-point $m_{a_tb_tc_t}$ according to (22). $[\![\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t]\!]_{i,j,k}$ is the recovered value in the t-th sub-tensor. Actually, Eq.(26) is the weighted sum of the recovered values in the related sub-tensors. Given an entry m_{ijk} that is contained in multiple sub-tensors, our fusion in Eq.(26) gives a larger weight to the entry which has a smaller recovery error in its sub-tensor. Therefore, our fusion process can help to achieve more accurate missing data recovery in the large tensor.

If a missing entry (i,j,k) is not covered by any selected sub-tensors, we can estimate its entry value using the factor matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ obtained from Eq.(12) with $\hat{m}_{i,j,k} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]_{i,j,k}$.

IX. Performance Evaluations

We use two public traffic traces Abilene [11] and GÈANT [12], and a synthetic trace to evaluate the performance of our proposed LTC. The synthetic trace is a large trace data which is generated through following steps. We firstly generate 4 low rank tensors with $400 \times 800 \times 400$, then combine these tensors to be a large tensor with $800 \times 800 \times 800$.

In the experiment, we apply the proposed tensor completion scheme to recover the full data from part of data samples. Then, using the raw trace data as reference, we calculate the performance metrics by comparing the recovered data with the original data in the trace.

We use two relative error ratio metrics to evaluate the recovery accuracy: **Error(sampled)** $= \frac{\sqrt{\sum_{(i,j,k)\in\Omega}(m_{i,j,k}-\hat{m}_{i,j,k})^2}}{\sqrt{\sum_{(i,j,k)\in\Omega}(m_{i,j,k})^2}}$ and **Error(inferred)** $= \frac{\sqrt{\sum_{(i,j,k)\in\overline{\Omega}}(m_{i,j,k}-\hat{m}_{i,j,k})^2}}{\sqrt{\sum_{(i,j,k)\in\overline{\Omega}}(m_{i,j,k})^2}}$, where Ω and $\overline{\Omega}$ denote the sets of indices of the sample and un-sample entries, respectively. m_{ijk} and $\hat{m}_{ijk}$ denote the raw data and the recovered data at (i,j,k)-th element of $\mathcal{M}$ where $1 \le i \le I$, $1 \le j \le J$ and $1 \le k \le K$. The first metric is the relative error to evaluate the impact of tensor completion on the data elements with values observed already, and the second is error for the element locations

with the values inferred from the tensor completion. For both traces, the sampling ratio is set to 40%.

In this paper, our LTC is designed based on CP decomposition (denoted as CP-LTC). For performance comparison, we also implement the tensor completion algorithm based on standard CP decomposition (denoted by CP). We perform three categories of experiments. In the first category, we validate the good clustering performance of our LSH based algorithm and lower rank feature in the sub-tensors extracting from the real data set. In the second category of experiments, we investigate the parameters used in the CP-LTC, based on which, we provide proper parameter setting for performance studies of CP-LTC in our experiments. In the third category of experiments, we implement other tensor completion algorithm following our LTC scheme, and demonstrate that our LTC design is general and can be exploited to increase the missing data recovery accuracy regardless of the underlying tensor completion algorithms.

As all tensor completion approaches are executed iteratively to train the parameters needed, for a fair comparison, we adopt the same two stop conditions: 1) The difference in the recovery loss between two consecutive iterations is smaller than a given threshold value, set to 10^{-6} in this paper; 2) The maximum number of iterations is reached, and we set the threshold to 100 in this paper. The iteration process will continue until either of the two stop conditions is satisfied.

As LTC and CP have many random components (for example, the random initiation of the factor matrices) and the results may vary across different runs, for each parameter setting, we repeat our experiments 20 times and present the statistical results in the experiments.

A. Method validation

1) Clustering performance based on LSH

In this paper, we use LSH table to reorder time, day, and OD pair and cluster them to facilitate finding the candidate anchor points. To compare the clustering performance with LSH, we implement k-means [54] and gaussian mixture model (GMM) [55]. To make them comparable with LSH, we set their cluster numbers to be the same as those of our LSH-based algorithm, where the cluster number of three domains are set respectively to 5(OD), 3(Time), and 2(Day).

To evaluate the clustering performance, two clustering metrics (Compactness and Separation) are defined as follows. Denote s clusters as $c_1, c_2, \cdots, c_s$. The centroid of cluster c_i is calculated as $\vec{u}_i = \frac{1}{|c_i|}\sum_{\vec{p}_j \in c_i} \vec{p}_j$. The average distance between each item in cluster c_i to its centroid is $CP_i = \frac{1}{|c_i|}\sum_{\vec{p}_j \in c_i}(\vec{p}_j - \vec{u}_i)$. The Compactness (CP) is defined as $CP = \frac{1}{s}\sum_{k=1}^{s} CP_i$. Separation (SP) is defined as $\frac{2}{s^2-s}\sum_{i=1}^{s}\sum_{j=i+1}^{s}||\vec{u}_i - \vec{u}_j||_2$. Low compactness value and large separation value indicate better and more compact clusters.

Table I and II show the clustering performance under different cluster algorithms. Besides compactness and separation, the computation time under different algorithms is also listed. It takes much less time for LSH to reorder and cluster the data, as it only requires a quick hash computation to group similar time, day, and OD pairs while other two reference methods need to run iteratively and incur significantly higher computational cost. Among all the clustering algorithms, our LSH achieves the similar compactness value and separation value with much lower computation time, thus better clustering performance.

TABLE I
CLUSTERING PERFORMANCE (ABILENE).

Abilene (OD)						
Method	Compactness		Separation		Computation Time	
	Mean	Variance	Mean	Variance	Mean	Variance
LSH	0.0350	4.1e-5	0.0790	4.21e-04	0.0019s	9.0e-08
K-means	0.1278	5.13e-5	0.0225	4.32e-04	0.0451s	1.9e-05
GMM	0.0516	1.73e-6	0.0517	6.67e-04	0.0088s	5.5e-06
Abilene (Time)						
Method	Compactness		Separation		Compuation Time	
	mean	Variance	Mean	Variance	Mean	Variance
LSH	0.0694	3.1e-04	0.4771	3.0e-03	0.0017s	2.1e-07
K-means	0.1230	1.7e-04	0.2425	3.8e-03	0.0406s	2.2e-06
GMM	0.0991	4.3e-04	0.1031	1.3e-03	0.085s	5.1e-06
Abilene (Day)						
Method	Compactness		Separation		Computation Time	
	mean	Variance	Mean	Variance	Mean	Variance
LSH	0.0369	3.8e-5	0.0882	0.0011	0.0017s	2.0e-07
K-means	0.0684	4.4e-4	0.0048	0.0019	0.0455s	5.0e-06
GMM	0.0557	2.3e-5	0.0529	0.00018	0.0105s	2.1e-05

TABLE II
CLUSTERING PERFORMANCE (GÈANT).

GÈANT (OD)						
Method	Compactness		Separation		Computation Time	
	Mean	Variance	Mean	Variance	Mean	Variance
LSH	0.0495	1.1e-4	0.0673	2.1e-04	0.0023s	2.1e-07
K-means	0.1090	2.7e-5	0.0118	5.9e-07	0.0570s	7.5e-05
GMM	0.0696	6.6e-5	0.0333	4.1e-04	0.0085s	7.25e-06
GÈANT (Time)						
Method	Compactness		Separation		Computation Time	
	mean	Variance	Mean	Variance	Mean	Variance
LSH	0.0433	1.9e-05	0.1515	2.6e-05	0.0021s	1.8e-05
K-means	0.0529	3.4e-05	0.0064	1.4e-07	0.0382s	1.1e-06
GMM	0.0563	1.5e-06	0.0980	3.0e-04	0.0122s	4.9e-07
GÈANT (Day)						
Method	Compactness		Separation		Compuation Time	
	mean	Variance	Mean	Variance	Mean	Variance
LSH	0.2545	3.3e-4	0.9874	4.5e-03	0.0015s	2.5e-07
K-means	0.3061	6.7e-5	0.4960	1.3e-01	0.0381s	1.7e-06
GMM	0.3383	4.2e-3	0.8419	2.4e-02	0.0115s	2.0e-05

2) Lower-rank verification of the sub-tensors

In our paper, the data with higher correlations are extracted to form the sub-tensors. To validate that the ranks of sub-tensors are lower than that of the original large tensor, we perform the following experiments. We randomly select 5 sub-tensors to compare with the whole large tensor. For each tested tensor, we use a low rank tensor to approximate the selected tensor through CP decompositions, and calculate the error on the entries in the tensor. In the tensor approximation process, CP rank is the important parameter. In Fig.13, we can see that the error rate decreases with the increase of CP rank, as an under-estimated rank R makes the CP decomposition far from capturing the full structure of the traffic tensor. For the sub-tensors, after their ranks reach $R = 3$, the error is very small. While using the same rank setting for the large tensor, the error rate is large. A small rank R can capture the structure in the sub-tensors, but is insufficient to capture the structure in large tensor. As all sub-tensors are extracted from the real data sets with higher correlation data, we can conclude that lower rank tensors exist in the data of real applications, which is the base of our techniques.

B. Parameter study

1) Impact of h

In our LTC, the entries with the distance to the anchor point less than h are selected into the sub-tensors. h directly impacts the size of the sub-tensor and correlation level among data in the sub-tensor. In our LTC, data correlation is calculated through angular distance. Besides, we also implement our LTC with Euclidean distance for

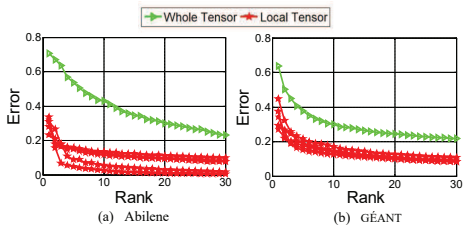


Fig. 13. Low-rank verification of sub-tensor.

performance comparison.

For each parameter setting, we repeat our experiments 20 times. To show the statistic results, a box plot (a.k.a. box and whisker diagram) is a good way of displaying the distribution of data based on the five number summary: minimum, first quartile, median, third quartile, and maximum. As a result, we use the box plot to draw the statistical results of our CP-LTC impacted by different h. Moreover, in Fig.16 (b)(c), Fig.14 (b) (c) and Fig.15 (b)(c), we also draw the curve to denote the average result.

Fig.16 (b)(c), Fig.14 (b) (c) and Fig.15 (b)(c) show how h impacts the missing data recovery performance. With the increase of h (both angular distance and Euclidean distance), the error rate decreases initially, but starts to increase when h exceeds a certain value. On the one hand, the increase of h allows more correlated entries to be included in the sub-tensors, which helps to increase the recovery accuracy. On the other hand, a larger h brings entries with lower correlation into the sub-tensor, which reduces the recovery accuracy. As CP does not have parameter h, we just show the average and variation of the experiment results. Obviously, compared with CP, our LTC achieves better recovery performance with smaller error rate.

By comparing Fig.16 (b) with Fig.16 (c), Fig.14 (b) with Fig.14 (c), and Fig.15 (b) with Fig.15 (c), the recovery performance under angular distance is better than that under Euclidean distance. This may be because that angular distance can better capture similar data trend instead of data value, and consequently bring higher recovery gain to the tensor completion algorithm. Therefore, instead of Euclidean distance, our LTC uses angular distance for sub-tensor building. From Fig.16, Fig.14 and Fig.15, we find the best performance is achieved when angular distance h = 0.6 for synthetic data, h = 0.5 for trace Abilene and GÈANT.

2) Impact of α

In Section VIII-A, the parameter α control the proportion of the distance and the density thus their tradeoff when we select anchor points from the candidate anchor points. The density ratio will be increased and the distance ratio will be reduced when α becomes larger. With different α, the anchor points selected will be changed, and thus change the recovery performance. Fig.19, Fig.17, and Fig.18 show the recovery performance by varying α. The curves in Fig.19, Fig.17, and Fig.18 are the quadratic fitting the average value. From the experiment results, we can conclude that our CP-LTC can achieve the best recovery performance when $\alpha = 0.5$ in the three traces. Therefore, we set $\alpha = 0.5$ in the rest of experiments.

3) Impact of q

Fig.22, Fig.20, and Fig.21 show the recovery performance as a function of the number anchor points (i.e., q) selected. For the comparison of different algorithms, we only draw the curves of their average results from many rounds. Besides the anchor point selection algorithm in Section VIII-A, two other anchor point selection algorithms are implemented. The first selects the anchor point randomly in the large tensor and is denoted by RAN. The second scheme (denoted by RAN-LSH) selects anchor points randomly from the group centers calculated in Section V.

Among all the local tensor completion schemes (CP-LTC, CP, RAN, and RAN-LSH), our CP-LTC achieves the best performance if the number of anchor points are the same. Obviously, when q is less than 5 in Fig.22, q is less than 8 in Fig.20, and q is less than 10 in Fig.21, the recovery errors of all the local tensor completion schemes (CP-LTC, RAN, and RAN-LSH) decline fast. When the sub-tensor number close to 4 (synthetic data) and 10 (Abilene and GÈANT), our CP-LTC reaches the convergent result with stable recovery errors. However, RAN-LSH converges when the sub-tensor number is close to 10 (synthetic data), 27 (Abilene) and 20 (GÈANT). Thus, the efficiency of our CP-LTC for local low-rank tensor approximation doubles that of RAN-LSH. According to the result, we set $q = 4$ (synthetic data) and $q = 10$ (Abilene and GÈANT) in our rest experiments.

C. Effectiveness of localized tensor completion scheme

1) Running time

By inserting the timer into the programs of CP and the CP-LTC, we compare the computation time. The computation time under our CP-LTC includes: (a) encoding, the time to train factor matrix to encode the data of three domains (time, day, OD pair); (b) sub-tensor building and selection: consisting of the time of applying LSH function to reorder the data of the three domains, the time of building sub-tensors, and the time of selecting sub-tensors considering density of samples; (c) completing sub-tensor: the time to complete sub-tensors; (d) fusion: the time to obtain the final data through fusion.

In Fig.26, obviously, the computation time under CP-LTC is slightly smaller than that under CP. However, our CP-LTC can more accurately recover the missing data in Fig. 25, Fig.23, and Fig.24. According to the stop condition described at the beginning of the experiment setting in Section IX, although the maximum iteration number is set to 100 for CP to train the factor matrices for data recovery, in our encoding phase, we only run 10 iterations to obtain the rough codes of the data in three domains for data clustering. From results in Fig. 25, Fig.23, and Fig.24, we find that such encoding works well to cluster data for sub-tensor completion. Therefore, the encoding time is small in our CP-LTC. Moreover, although we have multiple sub-tensors to complete, the completion tasks in different sub-tensors are independently and can be executed in parallel. Therefore, the total time in completing all sub-tensors is also not large.

2) Accuracy

Fig. 25, Fig.23, and Fig.24 compare the accuracy of tensor completion with the facilitation of our localized tensor factorization scheme. Similar to Fig.22, Fig.20, and Fig.21, we only show the average results of many rounds of different algorithms in Fig. 25, Fig.23, and Fig.24.

We implement three tensor completion algorithms CP, CP_{opt} [19], and CP_{nmu} [18] under our scheme. Following our scheme, these algorithms are first applied to the large incomplete monitoring tensor to obtain the codes of data in each domain, which are then used to determine the candidate anchor points using the algorithm in Section V. After well selecting the sub-tensors and recovering the missing sub-tensors, the final results are fused by combining the recovery results of sub-tensors. For performance comparison, we also implement the tensor completion algorithms directly using

CP			
Metrics	Mean	Variance	Times
Error (sampled)	0.6331	1.63e-06	40
Error (inferred)	0.6363	2.46e-06	40

(a) CP Error

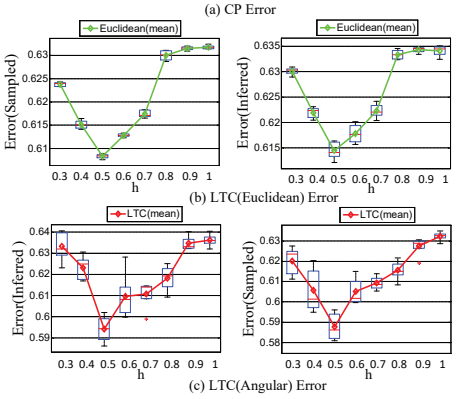


Fig. 14. Study impact of h (Abilene).

CP			
Metrics	Mean	Variance	Times
Error (sampled)	0.5479	6.03e-07	40
Error (inferred)	0.5503	5.57e-07	40

(a) CP Error

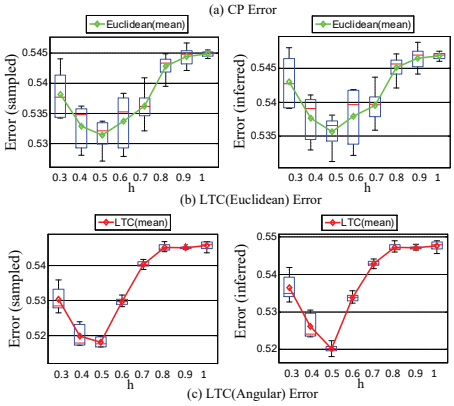


Fig. 15. Study impact of h (GÈANT).

CP			
Metrics	Mean	Variance	Times
Error (sampled)	0.7306	2.72e-05	40
Error (inferred)	0.7522	5.61e-06	40

(a) CP Error

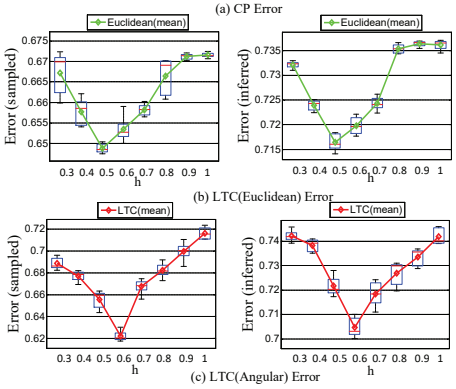


Fig. 16. Study impact of h (synthetic data).

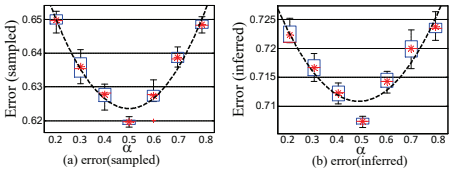


Fig. 17. Study impact of α using Abilene.

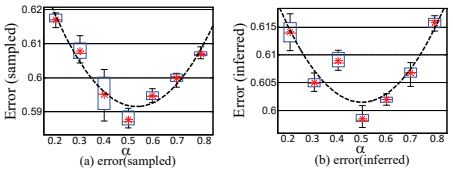


Fig. 18. Study impact of α using GÈANT.

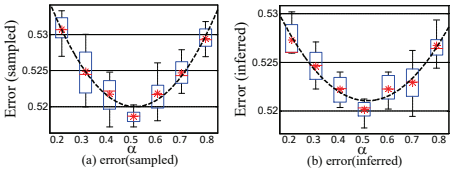


Fig. 19. Study impact of α using synthetic data.

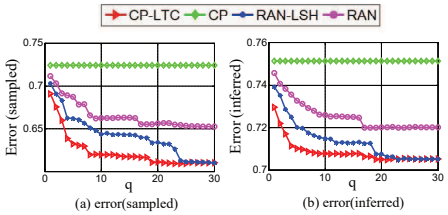


Fig. 20. Study impact of q (Abilene).

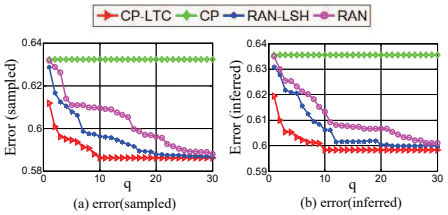


Fig. 21. Study impact of q (GÈANT).

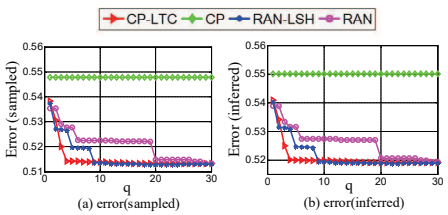


Fig. 22. Study impact of q (synthetic data).

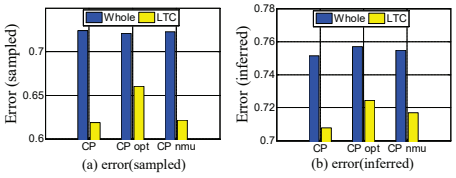


Fig. 23. Effectiveness of LTC (Abilene).

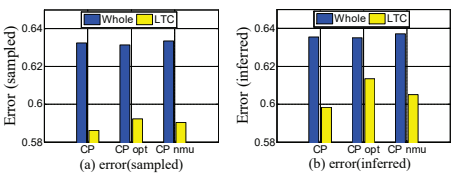


Fig. 24. Effectiveness of LTC (GÈANT).

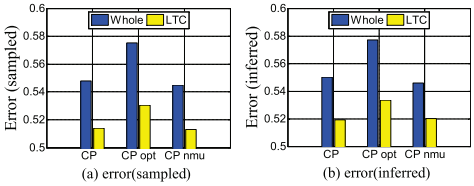


Fig. 25. Effectiveness of LTC (synthetic data).

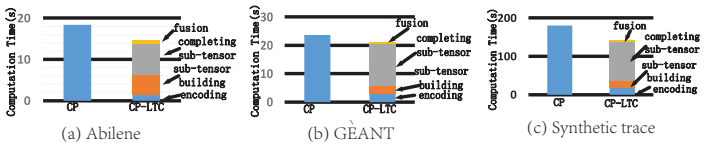


Fig. 26. Computation time comparison .

the whole large tensor without sub-tensor extraction.

Compared with the tensor completion algorithms executed using the whole data, our localized tensor factorization scheme groups entries with closer relationship into sub-tensors to more accurately recover the missing data. All the performance results (in Fig. 25, Fig.23, and Fig.24) demonstrate that our localized tensor factorization scheme is very effective in improving the recovery accuracy, and our localized tensor factorization scheme is general without depending on the underlying tensor completion algorithms.

X. Conclusion

We propose a novel LTC scheme which can exploit higher correlation among subsets of data to form and recover local sub-tensors for more accurately missing data recovery. To facilitate the finding of data similarity in the presence of incomplete measurement data, we propose a technique to encode tensor slices under the bases of factorized matrices.

Taking advantage of similarity calculation based on the encoded data, we propose following three techniques to build the local sub-

tensors and recover the large tensor. 1) In LTC, the local sub-tensors are built around some anchor points. To find anchor points that can well represent the correlations of data in a local tensor and distribute evenly inside the large tensor, we propose an algorithm to efficiently calculate the anchor points based on locality-sensitive hash. 2) To reduce the computation cost and increase the data recovery accuracy, we propose an anchor-point selection algorithm, taking into account the density of samples already taken and distances of anchors selected. 3) To more accurately recover the large tensor, we propose a similarity-sensitive local tensor completion algorithm and a data fusion algorithm. Extensive experiments using two real traffic traces demonstrate the effectiveness and efficiency of our LTC.

ACKNOWLEDGMENT

The work is supported by the National Natural Science Foundation of China under Grant Nos. 61972144, 61572184, 61725206, and 61976087, Hunan Provincial Natural Science Foundation of China under Grant No.2017JJ1010, U.S. NSF ECCS 1731238 and NSF CNS 1526843, the open project funding (CARCH201809) of State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, the Peng Cheng Laboratory Project of Guangdong Province PCL2018KP004.

REFERENCES

[1] K. Xie, L. Wang, X. Wang, G. Xie, G. Zhang, D. Xie, and J. Wen, "Sequential and adaptive sampling for matrix completion in network monitoring systems," in *IEEE INFOCOM*, 2015.

[2] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, "Structural analysis of network traffic flows," in *ACM SIGMETRICS*, 2003.

[3] Y. Zhang, M. Roughan, C. Lund, and D. Donoho, "Estimating point-to-point and point-to-multipoint traffic matrices: An information-theoretic approach," in *IEEE/ACM Trans. Netw*, 2005, pp. 947–960.

[4] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," *Acm Sigcomm Computer Communication Review*, vol. 34, no. 4, pp. 219–230, 2004.

[5] Y.Vardi, "Network tomography," *J. Amer. Statist. Assoc.*, vol. vol. 91, no. 433,, p. pp. 365–377, 1996.

[6] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies," *ACM IMW*, 2002.

[7] M. Roughan, Y. Zhang, W. Willinger, and L. Qiu, "Spatio-temporal compressive sensing and internet traffic matrices (extended version)," *Networking IEEE/ACM Transactions on*, vol. 20, no. 3, pp. 662 – 676, 2012.

[8] G. Gürsun and M. Crovella, "On traffic matrix completion in the internet," in *ACM IMC 2012*.

[9] Y.-C. Chen, L. Qiu, Y. Zhang, G. Xue, and Z. Hu, "Robust network compressive sensing," in *ACM MobiCom*, 2014.

[10] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications." *Siam Review*, vol. 51, no. 3, pp. 455–500, 2009.

[11] "The abilene observatory data collections. http://abilene. internet2.edu /observatory/data-collections.html."

[12] S. Uhlig, B. Quoitin, J. Lepropre, and S. Balon, "Providing public intradomain traffic matrices to the research community," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 83–86, 2006.

[13] J. Haupt, W. U. Bajwa, M. Rabbat, and R. Nowak, "Compressed sensing for networked data," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 92–101, 2008.

[14] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[15] R. H. Keshavan, A. Montanari, and S. Oh, "Matrix completion from a few entries," *IEEE Transactions on Information Theory*, vol. 56, no. 6, pp. 2980–2998, 2010.

[16] S.-J. Huang, M. Xu, M.-K. Xie, M. Sugiyama, G. Niu, and S. Chen, "Active feature acquisition with supervised matrix completion," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1571–1579.

[17] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 208–220, 2013.

[18] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, "Scalable tensor factorizations for incomplete data," *Chemometrics and Intelligent Laboratory Systems*, vol. 106, no. 1, pp. 41–56, 2011.

[19] E. Acar and T. G. Dunlavy, Daniel M.and Kolda, "A scalable optimization approach for fitting canonical tensor decompositions," *Journal of Chemometrics*, vol. 25, no. 2, p. 67–86, 2011.

[20] S. Gandy, B. Recht, and I. Yamada, "Tensor completion and low-n-rank tensor recovery via convex optimization," *Inverse Problems*, vol. 27, no. 2, p. 025010, 2011.

[21] Y. Zhang, H. Wang, D. Lian, I. W. Tsang, H. Yin, and G. Yang, "Discrete ranking-based matrix factorization with self-paced learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 2758–2767.

[22] D. Lian, R. Liu, Y. Ge, K. Zheng, X. Xie, and L. Cao, "Discrete content-aware matrix factorization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 325–334.

[23] S. C. Anyosa, J. Vinagre, and A. M. Jorge, "Incremental matrix co-factorization for recommender systems with implicit feedback," in *Companion of the The Web Conference 2018 on The Web Conference 2018*. International World Wide Web Conferences Steering Committee, 2018, pp. 1413–1418.

[24] J. Zhang, J. Chen, S. Zhi, Y. Chang, S. Y. Philip, and J. Han, "Link prediction across aligned networks with sparse and low rank matrix estimation," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 971–982.

[25] X. Ma and D. Dong, "Evolutionary nonnegative matrix factorization algorithms for community detection in dynamic networks," *IEEE transactions on knowledge and data engineering*, vol. 29, no. 5, pp. 1045–1058, 2017.

[26] J. Wang, J. Shen, P. Li, and H. Xu, "Online matrix completion for signed link prediction," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 2017, pp. 475–484.

[27] Q. Wang, M. Sun, L. Zhan, P. Thompson, S. Ji, and J. Zhou, "Multi-modality disease modeling via collective deep matrix factorization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 1155–1164.

[28] S. Tulyakov, X. Alameda-Pineda, E. Ricci, L. Yin, J. F. Cohn, and N. Sebe, "Self-adaptive matrix completion for heart rate estimation from face videos under realistic conditions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2396–2404.

[29] K. Xie, C. Peng, X. Wang, G. Xie, J. Wen, J. Cao, D. Zhang, and Z. Qin, "Accurate recovery of internet traffic data under variable rate measurements," *IEEE/ACM Transactions on Networking*, vol. 26, no. 3, pp. 1137–1150, 2018.

[30] K. Xie, L. Wang, X. Wang, G. Xie, J. Wen, G. Zhang, J. Cao, D. Zhang, K. Xie, X. Wang *et al.*, "Accurate recovery of internet traffic data: A sequential tensor completion approach," *IEEE/ACM Transactions on Networking (TON)*, vol. 26, no. 2, pp. 793–806, 2018.

[31] K. Xie, L. Wang, X. Wang, G. Xie, and J. Wen, "Low cost and high accuracy data gathering in wsns with matrix completion," *IEEE Transactions on Mobile Computing*, vol. 17, no. 7, pp. 1595–1608, 2017.

[32] K. Xie, C. Peng, X. Wang, G. Xie, and J. Wen, "Accurate recovery of internet traffic data under dynamic measurements," in *IEEE INFOCOM*, 2017.

[33] K. Xie, X. Ning, X. Wang, D. Xie, J. Cao, G. Xie, and J. Wen, "Recover corrupted data in sensor networks: A matrix completion solution," *IEEE Transactions on Mobile Computing*, vol. 16, no. 5, pp. 1434–1448, 2016.

[34] K. Xie, X. Ning, X. Wang, S. He, Z. Ning, X. Liu, J. Wen, and Z. Qin, "An efficient privacy-preserving compressive data gathering scheme in wsns," *Information Sciences*, vol. 390, pp. 82–94, 2017.

[35] J. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[36] R. A. Harshman, "Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis," *UCLA Working Papers in Phonetics*, vol. 16, no. 1, p. 84, 1970.

[37] L. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[38] B. W.Bader, T. G.Kolda *et al.*, "Matlab tensor toolbox version 2.5," January 2012. [Online]. Available: http://www.sandia.gov/ tgkolda/TensorToolbox/

[39] L. De Lathauwer, B. De Moor, and J. Vandewalle, "On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors," *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, 2000.

[40] X.-Y. Liu, S. Aeron, V. Aggarwal, and X. Wang, "Low-tubal-rank tensor completion using alternating minimization," *arXiv preprint arXiv:1610.01690*, 2016.

[41] X.-Y. Liu, S. Aeron, V. Aggarwal, X. Wang, and M.-Y. Wu, "Adaptive sampling of rf fingerprints for fine-grained indoor localization," *IEEE Transactions on Mobile Computing*, vol. 15, no. 10, pp. 2411–2423, 2016.

[42] Z. Zhang and S. Aeron, "Exact tensor completion using t-svd," *arXiv preprint arXiv:1502.04689*, 2015.

[43] E. Kernfeld, M. Kilmer, and S. Aeron, "Tensor – tensor products with invertible linear transforms," *Linear Algebra and Its Applications*, vol. 485, pp. 545–570, 2015.

[44] K. Xie, L. Wang, X. Wang, G. Xie, J. Wen, and G. Zhang, "Accurate recovery of internet traffic data: A tensor completion approach," in *IEEE INFOCOM*, 2016.

[45] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer, "Local collaborative ranking," in *Proceedings of the 23rd international conference on World wide web*. ACM, 2014, pp. 85–96.

[46] N. Ruchansky, M. Crovella, and E. Terzi, "Targeted matrix completion," in *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017, pp. 255–263.

[47] V. Bharti, P. Kankar, L. Setia, G. Gürsun, A. Lakhina, and M. Crovella, "Inferring invisible traffic," in *Proceedings of the 6th International COnference*. ACM, 2010, p. 22.

[48] L. De Lathauwer, "Blind separation of exponential polynomials and the decomposition of a tensor in rank-(l_r,l_r,1) terms," *SIAM Journal on Matrix Analysis and Applications*, vol. 32, no. 4, pp. 1451–1474, 2011.

[49] B. W. Bader and T. G. Kolda, "Algorithm 862: Matlab tensor classes for fast algorithm prototyping," *ACM Transactions on Mathematical Software (TOMS)*, vol. 32, no. 4, pp. 635–653, 2006.

[50] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *ACM SCG*, 2004.

[51] V. M. Zolotarev, *One-dimensional stable distributions*. American Mathematical Soc., 1986, vol. 65.

[52] A. Andoni and P. Indyk, "E 2 lsh 0.1 user manual," 2005.

[53] X. Li, K. Xie, X. Wang, G. Xie, J. Wen, G. Zhang, and Z. Qin, "Online internet anomaly detection with highaccuracy: A fast tensor factorization solution," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*.

[54] H. Spath, *The cluster dissection and analysis theory FORTRAN programs examples*. Prentice-Hall, Inc., 1985.

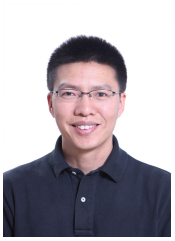
[55] G. McLachlan and D. Peel, *Finite mixture models*. John Wiley & Sons, 2004.

Kun Xie received PhD degree in computer application from Hunan University, Changsha, China, in 2007. She is currently a professor with Hunan University, Changsha, China. Her research interests include network measurement, network security, big data, and AI. She has published over 60 papers in major journals and conference proceedings (including journals IEEE/ACM TON, IEEE TMC, IEEE TC, IEEE TWC, IEEE TSC and conferences SIGMOD, INFOCOM, ICDCS, SECON, IWQoS)

Xiangge Wang is now a master in the College of Computer Science and Electronics Engineering, Hunan University. Her research interests include tensor completion and traffic data analysis.

Xin Wang (M'1 / ACM'4) received the Ph.D. degree in electrical and computer engineering from Columbia University, New York, NY. She is currently an Associate Professor in the Department of Electrical and Computer Engineering of the State University of New York at Stony Brook, Stony Brook, NY. Her research interests include algorithm and protocol design in wireless networks and communications, mobile and distributed computing, as well as networked sensing and detection. Dr. Wang achieved the NSF career award in 2005, and ONR challenge award in 2010.

Yuxiang Chen is now a PhD candidate in the College of Computer Science and Electronics Engineering, Hunan University. His research interests include parallel computing and matrix completion.



Gaogang Xie received his B.S. degree in Physics, M.S. degree and Ph.D. degree in computer science all from Hunan University respectively in 1996, 1999 and 2002. He is currently a Professor and Director of Network Technology Research Center with the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China. His research interests include Internet architecture, packet processing and forwarding, and Internet measurement.

Yudian Ouyang is now a PhD candidate in the College of Computer Science and Electronics Engineering, Hunan University. Her research interests include matrix completion, tensor completion, and AI.

Jigang Wen received PhD degrees in computer application from Hunan University, China, in 2011. He worked as a research assistant in the department of computing in Hong Kong Polytechnic University from 2008 to 2010. His research interests include wireless network and mobile computing, high speed network measurement and management.

Jiannong Cao (M'93-SM'05-FM'14) received the Ph.D degree in computer science from Washington State University, Pullman, WA, USA, in 1990. Dr. Cao is currently a Chair Professor of Department of Computing at The Hong Kong Polytechnic University, Hong Kong. He is also the director of the Internet and Mobile Computing Lab in the department and the director of University's Research Facility in Big Data Analytics. His research interests include parallel and distributed computing, wireless sensing and networks, pervasive and mobile computing, and big data and cloud computing.

Dafang Zhang received Ph.D. degree in Application Mathematics in Hunan University, Changsha, China, in 1997. He is currently a professor with Hunan University, Changsha, China. His research interests include packet processing, Internet measurement, wireless network and mobile computing, and big data.