Low Cost Online Network Traffic Measurement With Subspace-Based Matrix Completion

Kai Jin, Kun Xie, *Member, IEEE*, Xin Wang, *Senior Member, IEEE*, Jiazheng Tian, Gaogang Xie, *Senior Member, IEEE*, Jigang Wen, and Kenli Li, *Senior Member, IEEE*

***Abstract*—Traffic Matrix (TM) is important for network operation and management. However, it is hard to measure the complete TM due to the high measurement cost. Few recent studies propose using sparse measurement with only a subset of origin and destination pairs (OD pairs) while the other OD pairs are reconstructed through matrix completion. Although effective, current sparse network monitoring schemes can hardly support online network monitoring which requires scheduling the sample taking adaptively in each new time slot one by one. To meet the online network monitoring scenario, we propose a sparse network monitoring scheme by exploiting subspace-based matrix completion. Several novel techniques are proposed in our scheme. First, to capture the dynamic rank feature, we design the scheme based on sliding window and propose an algorithm to estimate the rank of current window even though we don't know the data of the upcoming time slot. Secondly, based on the rank estimated, we propose an adaptive sampling scheduling algorithm. Finally, we propose a lightweight algorithm to speed up the reconstruction process by reusing the matrix calculation results in the previous time slot. The experimental results demonstrate that our scheme guarantees the high precision of network-wide TM monitoring while significantly reducing the measurement cost.**

***Index Terms*—Matrix completion, online measurement, sliding window model, subspace-based matrix completion.**

I. Introduction

A TRAFFIC matrix (TM) tracks the traffic volumes exchanged between OD pairs in the target network during a period. It provides important information for many network tasks such as traffic engineering, path setup, capacity planning, and congestion control [1], [2].

Manuscript received 24 January 2022; revised 27 July 2022; accepted 21 August 2022. Date of publication 26 August 2022; date of current version 6 January 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 61972144 and 62025201, in part by NSF ECCS under Grants 1731238 and 2030063, and in part by NSF CIF under Grant 2007313. Recommended for acceptance by Prof. Hong Xu. *(Corresponding author: Kun Xie.)*

Kai Jin, Kun Xie, Jiazheng Tian, Jigang Wen, and Kenli Li are with the College of Computer Science, and Electronics Engineering, Hunan University, Changsha 410002, China (e-mail: jinkai@hnu.edu.cn; xiekun@hnu.edu.cn; tian11@hnu.edu.cn; wenjigang@ict.ac.cn; lkl@hnu.edu.cn).

Xin Wang is with the Department of Electrical, and Computer Engineering, State University of New York at Stony Brook, Stony Brook, NY 11794 USA (e-mail: x.wang@stonybrook.edu).

Gaogang Xie is with the Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Computer Science, and Technology, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: xie@ict.ac.cn).

Digital Object Identifier 10.1109/TNSE.2022.3201624

Although we can apply flow-based measurement tools such as NetFlow [3] and sFlow [4] to measure the flow traffic exchanged between OD pairs, as these tools often employ too many resources (e.g., memory, CPU) [5], it is infeasible to obtain the TM by measuring the flow traffic volumes between all OD pairs at all time intervals due to high measurement overhead.

To reduce the overhead, sample-based network traffic measurement is usually performed in the current network monitoring system. For example, in a time slot, only a small portion of OD pairs are taken measurements. However, traditional sample-based measurement strategy can not guarantee that the traffic volumes of the un-measured OD pairs can be accurately recovered.

With the progress of sparse representation techniques, matrix completion (MC) attracts lots of recent interests. According to the matrix completion theory, a matrix can be accurately reconstructed with a subset of observed entries if the target matrix has a low-rank feature. Several studies have pointed out that TMs have strong spatio-temporal correlations, thus the low-rank feature [2]. It inspires the exploration of sparse network monitoring where complete TM data are obtained by selecting a subset of OD pairs to take flow measurements while inferring the un-measured data through matrix completion algorithms.

Several studies [1], [6], [7], [8] have applied matrix completion to network monitoring. Given a subset of measurement samples, these studies usually apply matrix completion to recover un-measured data, but can not guarantee that the un-measured data can be accurately recovered, as measurement samples may not carry sufficient information.

Very few studies [9], [10] try to select measurement samples following the matrix completion framework. Taking SVT (Singular Value Thresholding) [11] and MF (Matrix Factorization) [2], [12] as the basic matrix completion algorithms, these schemes model network monitoring data in each time slot using a matrix with its row denoting the origin node and column denoting the destination node. Restricted by SVT and MF, the samples are only selected within the matrix, that these schemes can only consider spatial relationship among monitoring data while ignoring their temporal features.

Different from the above network monitoring schemes, to take advantage of both spatial and temporal information, we model the network traffic data using a matrix with its row denoting the OD pairs and column denoting the time slot. We

aim to solve an online network monitoring problem in which, for each upcoming time slot, a subset of OD pairs are actively scheduled to take measurements while guaranteeing the un-measured data of the remainder OD pairs can be reconstructed accurately through matrix completion.

Among matrix completion techniques, subspace-based matrix completion [13], [14], [15], [16] reconstructs un-measured data column by column. It may fit our online network monitoring scenario, where the monitoring data of a time slot corresponds to one column. We would like to schedule the measurement process online slot by slot, thus obtaining measurement data columns one by one. However, directly applying the subspace-based matrix completion for online network-wide traffic monitoring faces a number of challenges:

- For a dynamic network, the rank of the traffic matrix changes, and it is difficult to maintain the varying subspace resulted.
- Without prior information on the data, the sampling scheduling is made hard, as it is difficult to ensure that data for each upcoming time slot can be inferred.
- The un-measured data in each new time slot should be reconstructed quickly to meet the speed requirements from the tasks that need the network monitoring data timely.

To address the above issues, in this paper, we propose a sequential and adaptive sampling scheme that exploits subspace-based matrix completion to achieve low cost and high accuracy network-wide traffic monitoring. Our scheme is designed based on the sliding window where the information from the previous windows is applied to guild the sampling scheduling in each upcoming time slot. The technique contributions made in this paper are summarized as follows:

- We propose an algorithm to estimate the rank of the current window with the information of the previous window even though the data of the upcoming time slot are unavailable.
- Based on the rank estimated, we propose an adaptive sampling scheduling algorithm to select a subset of OD pairs to take measurements in the upcoming time slot while guaranteeing the accuracy of inferring the un-measured data.
- By tracking the subspace columns in each siding window, we propose an approach to effectively maintain the dynamic subspace of the window.
- We propose a lightweight algorithm to reconstruct un-measured data by intelligently reusing the matrix calculation results in the previous time slot to speed up the reconstruction process.
- We conduct extensive experiments on two real traffic flow data sets (Abilene [17] and GÈANT [18]), a synthetic data set, and an urban road traffic speed data set [19]. We demonstrate that our scheme can obtain the complete network monitoring data at low measurement cost, with good performance in terms of reconstruction accuracy and processing time.

The rest of the paper is organized as follows. We present notations and definitions used in this paper and introduce the related work in Section II. We study the real trace data in Section III, introduce the problem and challenge in Section IV. In Section V, we present a solution overview. In Sections VI and VII, we propose our sampling scheduling algorithm. In Section VIII, we discuss how to apply our sampling scheduling scheme for practical traffic measurement. We evaluate the performance of the proposed algorithm through extensive experiments in Section IX, and conclude the work in Section X.

II. Preliminary and Related Work

We first give the notations and definitions used in this paper, and then review the related work.

A. Notations

For a vector $x = (x_1, x_2, \ldots, x_n)$ of size n, we call x_i the i th coordinate of x. Let Ω denote the set of sampling locations for any $\Omega \subset [n]$, and x_Ω denote the sub-vector of x from the coordinate set Ω. For example, for the vector $x = (5, 6, 7, 8, 9, 10)$ and $\Omega = \{1, 4\}$, x_Ω represents the sub-vector of x with $x_\Omega = (5, 8)$. $\|x\|$ denotes the vector ℓ_2 norm of x. For any matrix $\mathbf{M} \in \mathbb{R}^{N \times T}$, $\mathbf{M}_t$ denotes the t th column of the matrix $\mathbf{M}$, $\mathbf{M}_{\Omega t}$ is a sub-vector of $\mathbf{M}_t$. The column space spanned by target matrix $\mathbf{M}$ is denoted as $\mathbf{U}$, $\hat{\mathbf{U}}$ denotes our estimated column space. The projection operator of the subspace $\mathbf{U}$ is represented by $\mathcal{P}_{\mathbf{U}} = \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T$.

B. Definitions

Definition 1: (Full SVD, Skinny SVD, and Truncated SVD) [20] For an $n_1 \times n_2$ matrix $\mathbf{M}$, its Singular Value Decomposition (SVD) is defined by $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}$, where $\mathbf{U} \in \mathbb{R}^{n_1 \times n_1}$ and $\mathbf{V} \in \mathbb{R}^{n_2 \times n_2}$ are unitary matrix matrices and $\mathbf{\Sigma} \in \mathbb{R}^{n_1 \times n_2}$ is a diagonal matrix whose diagonal elements (also known as singular values) are organized in a decreasing order i.e. $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_r, 0, \cdots, 0)$, where r is the matrix rank and σ_i is the i th singular value. The SVD defined in this way is called the full SVD.

If we only keep the positive singular values, the reduced form is called the skinny SVD. For a matrix $\mathbf{M}$ of rank r, its skinny SVD is computed by $\mathbf{M} = \mathbf{U_r}\mathbf{\Sigma_r}\mathbf{V_r^T}$, where $\mathbf{\Sigma_r} = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_r)$ with $\{\sigma_i\}_{i=1}^r$ being positive singular values. More precisely, $\mathbf{U_r}$ and $\mathbf{V_r}$ are formed by taking the first r columns of $\mathbf{U}$ and $\mathbf{V}$, respectively.

Truncated SVD is another reduced form of SVD, it only keeps the first k singular values. We can get the rank k truncated SVD of $\mathbf{M}$ by $\mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k^T}$, where $\mathbf{U_k} \in \mathbb{R}^{n_1 \times k}$, $\mathbf{V_k} \in \mathbb{R}^{n_2 \times k}$, and $\mathbf{\Sigma_k} = \mathrm{diag}(\sigma_1, \sigma_2, \cdots, \sigma_k)$.

Definition 2: (Column Space) [20] For a matrix $\mathbf{M}$, its column space is the linear space spanned by its column vectors. Denoting the skinny SVD of $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathbf{T}$, then $\mathbf{U}$ contains orthogonal bases of the column space.

C. Fundamentals of Matrix Completion

Matrix completion aims to recover the entire matrix by inferring the unobserved entries using a small set of entries

observed. If we do not have constraints on the matrix, the entries inferred can be arbitrary. Matrix completion usually has a low-rank constraint on the matrix.

Given an unknown matrix $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$ with the rank $r << \min\{n_1, n_2\}$ while only a subset of its entries $\mathbf{M}_{ij}, (i,j) \in \Omega$ are observed, by finding a matrix $\mathbf{X}$ with the lowest rank to approximate $\mathbf{M}$, the matrix completion problem can be formulated as:

$$\min_{\mathbf{X}} \operatorname{rank}(\mathbf{X}), \quad \text{s.t.} \quad [\mathbf{X}]_{ij} = [\mathbf{M}]_{ij}, \forall (i,j) \in \Omega \tag{1}$$

where the sampling operator $P_\Omega : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times n_2}$ is defined by:

$$P_\Omega(\mathbf{X})_{ij} = \begin{cases} \mathbf{X}_{ij} & (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $\mathbf{X}$ is the matrix recovered.

However, rank minimization problem in (1) is non-convex and NP-hard. To overcome the problem, Candès et al. [21] uses the rank's convex relaxation and transfers the problem in (1) to a nuclear norm minimization problem:

$$\min_{\mathbf{X}} \|\mathbf{X}\|_*, \quad \text{s.t.} \quad [\mathbf{X}]_{ij} = [\mathbf{M}]_{ij}, \forall (i,j) \in \Omega \tag{3}$$

where $\|\mathbf{X}\|_*$ denotes the nuclear norm of $\mathbf{X}$, which is the sum of the singular values of $\mathbf{X}$. Many algorithms have been proposed to solve the nuclear norm minimization problem (3), including ALM [22] and SVT [11].

Besides (3), matrix factorization (MF) is another way to solve the non-convex problem. To infer the missing entries in a sparse matrix $\mathbf{M} \in \mathbb{R}^{n_1 \times n_2}$, as shown in (4), matrix factorization decomposes $\mathbf{M}$ into the production of two (low-rank) matrices $\mathbf{L} \in \mathbb{R}^{n_1 \times r}, \mathbf{R} \in \mathbb{R}^{r \times n_2}$ where r is the rank of $\mathbf{M}$.

$$\min \frac{1}{2}\left(\|\mathbf{L}\|_F^2 + \|\mathbf{R}\|_F^2\right) \text{s.t.} (\mathbf{LR})_{ij} = \mathbf{M}_{ij}, \forall (i,j) \in \Omega \tag{4}$$

After using the observed entries to train $\mathbf{L}$ and $\mathbf{R}$, the sparse matrix can be recovered by $\hat{\mathbf{M}} = \mathbf{LR}$. Following the framework of MF, NMF [12] tries to find nonnegative factor matrices $\mathbf{L}$, $\mathbf{R}$ to infer the unobserved nonnegative entries, SRMF [2] proposes a spatiotemporal model to infer the unobserved data.

We focus on online network monitoring, that is, for each upcoming time slot, a new column that corresponds to the measurement data in the time slot will be added to the matrix as the last column sequentially, and we should recover the unmeasured data in the column when it arrives.

Although above matrix completion techniques can infer the unobserved entries within the matrix, they usually require training the parameters using entries observed and then infer the unobserved ones within the matrix offline. Above matrix completion techniques are not suitable for online network monitoring which requires the handling of a new column sequentially added to the matrix.

In recent years, a novel type of matrix completion algorithm, subspace-based matrix completion is proposed in [13], [14], [15], [16], which may be suitable for online monitoring. The completion rationale of these methods is that if a column vector belongs to the subspace of the matrix, it can be represented by the subspace.

Given a subspace $\mathbf{U}$, for an upcoming column $\mathbf{M}_t$ with its sampling location set being Ω, the sub-vector of the observed data is $\mathbf{M}_{\Omega t}$. If $\mathbf{M}_t \in \mathbf{U}$, that is, $\mathbf{M}_t$ can be represented by the subspace, then this column can be recovered by

$$\hat{\mathbf{M}}_t = \mathbf{U}\left(\mathbf{U}_\Omega^T \mathbf{U}_\Omega\right)^{-1} \mathbf{U}_\Omega^T \mathbf{M}_{\Omega t}, \tag{5}$$

where $\alpha_t^* = (\mathbf{U}_\Omega^T \mathbf{U}_\Omega)^{-1} \mathbf{U}_\Omega^T \mathbf{M}_{\Omega t}$ is the r-dimensional coefficient that projects $\mathbf{M}_t$ to the subspace, and it can be calculated by using the sub-vector of the observed data in the column through

$$\min_{\alpha_t \in \mathbb{R}^r} \frac{1}{2} \|\mathbf{M}_{\Omega t} - \mathbf{U}_\Omega \alpha_t\|_2^2. \tag{6}$$

The subspace-based matrix completion algorithm can perform data inference column by column, which is suitable for our online network monitoring scenario. Therefore, in this paper, we propose our online network monitoring scheme based on subspace-based matrix completion.

D. Related Work

Some matrix completion-based algorithms are designed for network monitoring. SRMF [2] captures spatio-temporal features in traffic matrices (TMs) and proposes the first spatio-temporal model to infer missing data in the network. Following SRMF, several studies [1], [6], [7], [8] are proposed for traffic matrix recovery, aiming to recover the missing data from partial measurements data. As discussed in Section I, given a subset of sample entries, these studies try to infer the un-measured data. However, they may suffer from big errors if the subset of entries does not carry sufficient information for missing data inference.

Very few studies start to select measurement samples following the matrix completion framework. By modeling the network delay measurement data in a time period as a matrix with its row being the origin node and the column being the destination node, MC-E2E [9] designs a dynamic sampling algorithm to select the OD pairs to take measurements. It iteratively performs pre-recovery using previous samples and selects locations with the largest recovery errors to take further samples. This paper also proposes a stopping condition, that is, when this stopping condition is met, there is no need for further sampling. After obtaining adequate samples, it uses SVT [11] to infer the un-measured data. Using the same stopping condition proposed in [9], LC (Local Coherence) [10] proposes an adaptive sampling algorithm for network measurement. The LC [10], [23] is a two-stage sampling algorithm. In the first stage, it takes a small number of random samples and then calculates the leverage score of each unobserved element by performing SVD decomposition using the few random samples taken. The second stage selects the sample locations with the probability determined by the magnitude of their

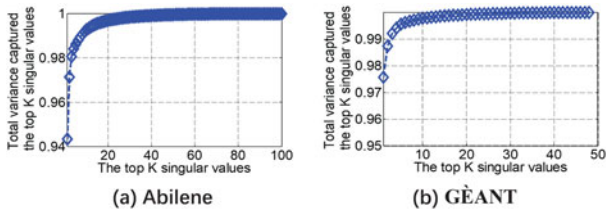


Fig. 1. Good low-rank structure of real network traffic matrix.

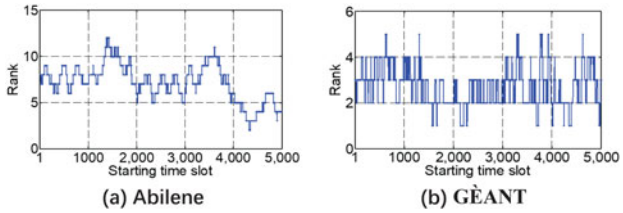


Fig. 2. Rank variation of real monitoring data in each window.

leverage scores. *Relying on multiple rounds of data pre-recovery based on insufficient sampling data, the sampling selection strategy in MC-E2E [9] and LC [10], [23] is not feasible due to its high computation cost and inaccurate sample selection. Moreover, as the monitoring matrix only records the monitoring data in one period, such sampling selection can not utilize the temporal information hidden in the monitoring data and can hardly support online network monitoring to obtain the stream of network data.*

Besides the network field, some online matrix completion methods are proposed to handle streaming or sequential data, where the columns of the matrix arrive sequentially. ORLRMR [24] proposes a matrix decomposition-based online matrix completion method in which the newly arrived column is reconstructed by $\mathbf{U}$. ReProCS [25] proposes an online completion algorithm based on recursive projective compressive sensing. These online matrix completion algorithms utilize all time slot data to recover the newly arrived data, which may lead to poor recovery performance, as too old data may have a negative impact on new data.

In this paper, we propose an online sampling scheduling algorithm based on a sliding window model, which uses the data of the previous window to determine how many entries to take measurements in the upcoming time slot and then reconstructs the un-measured ones in the time slot. By capturing the varying rank in the sliding window and dynamically adjusting the number of samples taken in each upcoming time slot according to the estimated rank of the current window, our scheme can capture both spatial and temporal features in the monitoring data while minimizing the measurement cost and guaranteeing the reconstruction performance.

III. Empirical Study With Real Trace Data

In matrix completion, low-rank is necessary for accurate reconstruction of the monitoring data, and the rank of the matrix directly impacts the number of samples required to take. Existing matrix completion solutions often assume that the data matrix has a known and fixed rank. Therefore the number of sample measurements to take is fixed and determined by the relationship between the smallest required number of samples and the rank r of the matrix. However, in network monitoring, such an assumption is unlikely to hold, and the rank of the traffic matrix varies over time.

We present the following experimental studies to demonstrate the low-rank feature of the traffic monitoring data and the dynamic rank feature of the traffic monitoring data.

A. Low-Rank Feature

According to [26], [27], [28], if a matrix has good low-rank structure, its top k singular values occupy nearly the total energy, i.e., $\sum_{i=1}^{k}\sigma_i^2 \approx \sum_{i=1}^{r}\sigma_i^2$ where σ_i is the i th singular value of the matrix. In this paper, we perform SVD on the traffic matrix, and use the ratio $g(k) = \sum_{i=1}^{k}\sigma_i^2 / \sum_{i=1}^{r}\sigma_i^2$ to verify low-rank feature of the traffic matrix.

Fig. 1 shows the fraction $g(k)$ of the total variance captured by the top k singular values for both traffic data sets Abilene [17] and GÈANT [18]. We find that in both traffic matrices, the top 10 singular values capture more than 95 percent variance in the real traces, which indicates that these two traffic data sets have good low-rank structures.

B. Dynamic Rank Feature

As shown in Section IV-A, our network monitoring model is designed based on sliding window model. We plot the rank of the consecutive sliding window by varying the starting time slot from 1 to 5000. For each sliding window, its rank is calculated as the least k that makes $g(k)$ more than 99%. We set the window size $T = 144$ in the Abilene data set and window size $T = 48$ in the GÈANT data set to make each window contain 12 hours of monitoring data. In Fig. 2, the X-axis represents the starting time slot of the matrix, and the Y-axis represents the rank of the matrix in the corresponding sliding window.

Most of the existing matrix completion-based data gathering schemes assume that the rank of the matrix is low and has a fixed value [9], [10]. However, from Fig. 2, we find that the rank varies over time in network monitoring data. Therefore, it is inefficient to use a fixed rank to guide the sampling schedule. In a dynamic network environment, accurately estimating the rank in each consecutive window is essential for designing a proper sampling scheduling algorithm. Using a large rank to guide sampling scheduling leads to over-sampling thus high cost while using a small rank may lead to poor recovery accuracy.

IV. Problem and Challenge

A. Online Traffic Monitoring Based on Sliding Window

For a network consisting of n nodes, there are $N = n \times n$ OD pairs. Let the column vector $\mathbf{M}_j$ denote the traffic volume snapshot of the whole network in the time interval $[j, j + \Delta t)$, i.e., the column of data measured at the time slot j. Each entry $M_{(i,j)}$ denotes the traffic volume data of OD pairs i at time slot j.

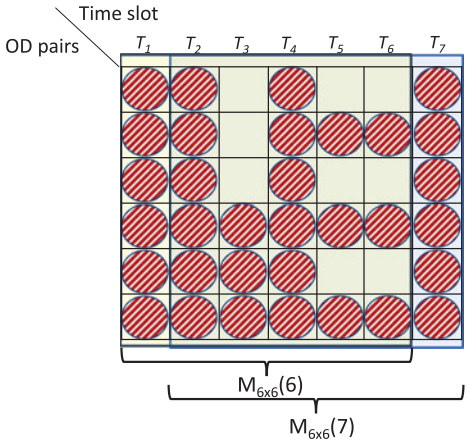


Fig. 3. An example of siding window model.

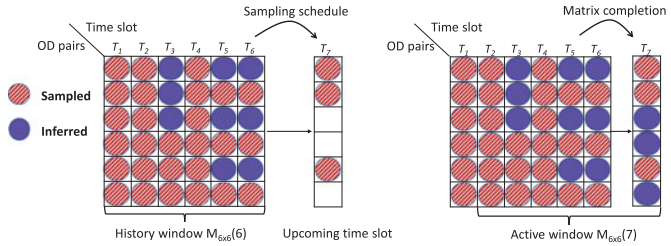


Fig. 4. Online sampling schedule and reconstruction problem.

The aim of this paper is to design an online traffic monitoring scheme based on matrix completion. That is, according to matrix completion theory, a schedule needs to be made to determine which OD pairs to take samples in each upcoming time slot to minimize measurement overhead while satisfying the reconstruction accuracy of the un-measured data.

For each upcoming time slot, as the recent data have more closed temporal relation, the reconstruction accuracy for the un-measured data may be higher by using recent data than using the data collected long ago. In this paper, we design our online traffic monitoring scheme based on the sliding window model to catch this essence.

For the sliding window model used in this paper, we define a matrix $\mathbf{M}_{N\times T}(t-1)$ to hold the monitoring data collected in recent T time slots until $t-1$. It covers the monitoring data $\{\mathbf{M}_{t-T}, ..., \mathbf{M}_{t-2}, \mathbf{M}_{t-1}\}$ where T represents the window size, N represents the number of OD pairs. The first column $\mathbf{M}_{t-T}$ in the matrix $\mathbf{M}_{N\times T}(t-1)$ represents the network monitoring data collected at the time slot $t-T$. The last column $\mathbf{M}_{t-1}$ in the matrix $\mathbf{M}_{N\times T}(t-1)$ represents the monitoring data collected in the time slot $t-1$. We refer to the window that contains the upcoming time slot t as the active window, and the adjacent window containing the data till $t-1$ as the history window. When the columns in one window are full and new data arrive, the data update operation follows the first-in-first-out (FIFO) principle, i.e., the column that enters the window the earliest will be removed first.

Fig. 3 shows an example of a sliding window model with window size $T=6$ and the number of OD pairs $N=6$. According to the definition, the matrices in two adjacent windows in Fig. 3 are denoted as $\mathbf{M}_{6\times 6}(6)$ and $\mathbf{M}_{6\times 6}(7)$, respectively.

In our scheme, the historical window is applied to guide the data sampling for the upcoming time slot. **The online traffic monitoring scheme can be described as follows**: in each upcoming time slot t, a subset of OD pairs are selected to take flow measurements based on the information from the historical window while ensuring the traffic volume data of the un-measured OD pairs to be accurately inferred. As only a subset of OD pairs are measured, the remaining pairs are inferred based on the samples observed through matrix completion. The measurement overhead can be largely reduced.

Fig. 4 shows an example to illustrate our sampling schedule problem. In the example, the size of the sliding window is $T=6$. The historical window has the measurement data $\{\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3, \mathbf{M}_4, \mathbf{M}_5, \mathbf{M}_6\}$, where the red entries are the measurement data. T_7 is the upcoming time slot. We want to determine the measurement samples in T_7 based on the historical window (the left of Fig. 4). After the measurement samples are gathered, the active window contains $\{\mathbf{M}_2, \mathbf{M}_3, \mathbf{M}_4, \mathbf{M}_5, \mathbf{M}_6, \mathbf{M}_7\}$. The scheme should guarantee that the un-measured data in T_7 be reconstructed by using the measurement data in the active window (the right of Fig. 4).

B. Solution Framework

According to the subspace-based matrix completion [13], [14], [15], [16], if we can maintain the subspace (i.e., $\mathbf{U}$) of the monitoring data, for each upcoming time slot, we can select Ω OD pairs satisfying $|\Omega| \geq c\mu(\mathbf{U})r\log(r/\delta)$ [15], where r is the rank of the monitoring data, $\mu(\mathbf{U})$ is the coherence parameter on the column space of monitoring data, c is a universal constant, δ is a probability variable reflecting the probability of failure to recover. According to (5), we can recover the un-measured data using the subspace through $\hat{\mathbf{M}}_t = \mathbf{U}(\mathbf{U}_\Omega^T\mathbf{U}_\Omega)^{-1}\mathbf{U}_\Omega^T\mathbf{M}_{\Omega t}$. Therefore, we can design an online traffic measurement scheduling scheme that takes three main steps:

- Maintaining the subspace $\mathbf{U}$ using the historical data.
- Selecting Ω OD pairs for each upcoming time slot to take traffic flow measurements, satisfying $|\Omega| \geq c\mu(\mathbf{U})r\log(r/\delta)$
- Recovering the un-measured data using the subspace through $\hat{\mathbf{M}}_t = \mathbf{U}(\mathbf{U}_\Omega^T\mathbf{U}_\Omega)^{-1}\mathbf{U}_\Omega^T\mathbf{M}_{\Omega t}$ in (5).

C. Challenging Issues

Applying subspace-based matrix completion for traffic monitoring in a dynamic network faces a number of challenges:

- **The change of rank for the matrix in each sliding window requires maintaining a dynamic subspace in the subsequent sliding window.** Our observation on real-world trace indicates that the rank of the sliding window varies over time. Accordingly, to accurately reconstruct the un-measured data, the subspace also needs to be updated to keep the low-rank feature of the

Algorithm 1: Build the Basic Subspace in the Training Phase

Require: The entire matrix (i.e., $\mathbf{M}$) of the first window.

Ensure: The set of subspace columns $\mathbf{S}$ in the first window, the estimated subspace $\hat{\mathbf{U}}$ and the estimated rank r.

1: Initialize $\mathbf{S} \leftarrow \emptyset$, $\hat{\mathbf{U}} \leftarrow \emptyset$.
2: Estimate rank r by performing full SVD on $\mathbf{M}$.
3: $\mathbf{S} \leftarrow [\mathbf{S}\ \ \mathbf{M}_1]$, mark $\mathbf{M}_1$ as subspace column, perform skinny SVD on $\mathbf{S}$ with $\mathbf{S} = \mathbf{U}_{|\mathbf{S}|}\mathbf{\Sigma}_{|\mathbf{S}|}\mathbf{V}^{\mathbf{T}}_{|\mathbf{S}|}$, set $\hat{\mathbf{U}} \leftarrow \mathbf{U}_{|\mathbf{S}|}$.
4: **for** $1 \leq i \leq r-1$ **do**
5: $\quad t = \underset{1<t\leq T, \mathbf{M}_t \notin \mathbf{S}}{\operatorname{argmax}} \|\mathbf{M}_t - \mathcal{P}_{\hat{\mathbf{U}}}\mathbf{M}_t\|_2^2$.
6: $\quad \mathbf{S} \leftarrow [\mathbf{S}\ \ \mathbf{M}_t]$, mark $\mathbf{M}_t$ as subspace column, perform skinny SVD on $\mathbf{S}$ with $\mathbf{S} = \mathbf{U}_{|\mathbf{S}|}\mathbf{\Sigma}_{|\mathbf{S}|}\mathbf{V}^{\mathbf{T}}_{|\mathbf{S}|}$, set $\hat{\mathbf{U}} \leftarrow \mathbf{U}_{|\mathbf{S}|}$.
7: **end for**
8: Return $\mathbf{S}$, $\hat{\mathbf{U}}$, r.

dynamic monitoring data. The subspace updating is essential as it will be further utilized in the reconstruction of future data.

- **It is hard to determine the number and location of samples without any information on the upcoming traffic data.** To make sure that the un-measured data can be accurately reconstructed, the number of measurement samples taken in each upcoming time slot should be large enough to capture sufficient information. If we can estimate the rank of the active window, we can identify the number of samples needed in the upcoming time slot while guaranteeing the reconstruction quality of the un-measured data. However, without the information of data in each upcoming time slot, the rank estimation is difficult. Also, as the rank can change, the sampling scheduling needs to be adaptive.
- **Fast reconstruction of the upcoming time slot**. The un-measured data should be reconstructed as soon as possible to meet the speed requirements from its subsequent tasks, such as anomaly detection and traffic management. To get the complete data in real-time, the un-measured data reconstruction should be accomplished upon the arrival of the measurement samples of each upcoming time slot.

V. Solution Overview

Our online network measurement scheme consists of two phases: training phase and window sliding phase.

The training is performed for the first T time slots, where the whole network traffic monitoring data are gathered. The main task of this phase is to build the basic subspace of the first window by selecting the subset of columns each corresponding to a time slot, and we call the columns selected as the subspace columns.

In the window sliding phase, for each upcoming time slot t ($t > T$), we first estimate the rank of the matrix of the active window, based on which we make the sampling scheduling decision and update the subspace to track the rank of the active window.

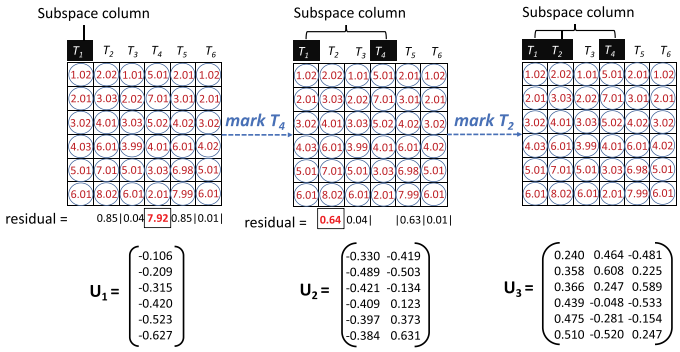


Fig. 5. Training and marked the indexes of the subspace columns.

In our sliding window model, we maintain the time slot indexes of the subspace columns for each window. In our scheme, the number of these columns also denotes the rank of the window. When a subspace column moves out of a window, the new column corresponding to the upcoming time slot may also be selected as the subspace column again.

VI. Build the Basic Subspace in the Training Phase

The whole network traffic monitoring data are gathered for the first T time slots. To identify the subspace columns in the window, we design **Algorithm 1**. As we have all the data in the first window, we can estimate the rank (i.e., r) of the first window through full SVD decomposition of the matrix in the window, where r is calculated as the least k that makes $g(k)$ larger than 99%, where $g(k)$ is defined in Section III-A.

Initially, we mark the first column as the subspace column, insert it into the set of subspace columns $\mathbf{S}$ and then perform a skinny SVD on $\mathbf{S}$ with $\mathbf{S} = \mathbf{U}_{|\mathbf{S}|}\mathbf{\Sigma}_{|\mathbf{S}|}\mathbf{V}^{\mathbf{T}}_{|\mathbf{S}|}$. We set the subspace as $\hat{\mathbf{U}} = \mathbf{U}_{|\mathbf{S}|}$, where $|\mathbf{S}|$ represents the number of columns in $\mathbf{S}$. We then adopt a greedy strategy to find other $r-1$ subspace columns, as shown in step 3, among all other columns in the window, we select the column with the largest residual as the subspace column with the residual calculated by $\|\mathbf{M}_t - \mathcal{P}_{\hat{\mathbf{U}}}\mathbf{M}_t\|_2^2$ where $\mathcal{P}_{\hat{\mathbf{U}}} = \hat{\mathbf{U}}(\hat{\mathbf{U}}^T\hat{\mathbf{U}})^{-1}\hat{\mathbf{U}}^T$. As the column with large residual can hardly be represented by the subspace $\hat{\mathbf{U}}$ found so far.

Fig. 5 shows an example of **training phase**, where $N = 6$ and $T = 6$. After we get the whole data in the first window, we perform an SVD on the window matrix to calculate the rank r. In this example, we set r as the least k that makes $g(k) > 99.99\%$, hence rank $r = 3$. To find the subspace in the training window, we first mark the first column $\mathbf{M}_1$ as the subspace column, add it into $\mathbf{S}$ and get the subspace (denoted by $\mathbf{U}_1$ in Fig. 5) by performing skinny SVD on $\mathbf{S} = [\mathbf{M}_1]$. Among all the columns in the window, we find that column $\mathbf{M}_4$ has the largest residual. Thus, we mark $\mathbf{M}_4$ as the subspace column and update the subspace (denoted by $\mathbf{U}_2$) by performing skinny SVD on matrix $[\mathbf{M}_1, \mathbf{M}_4]$. Again, among the remaining columns in the window, we find $\mathbf{M}_2$ has the largest residual. Thus we further mark $\mathbf{M}_2$ as the subspace column and perform the skinny SVD on matrix $[\mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_4]$ to obtain the basic subspace (i.e., $\mathbf{U}_3$) of the first window.

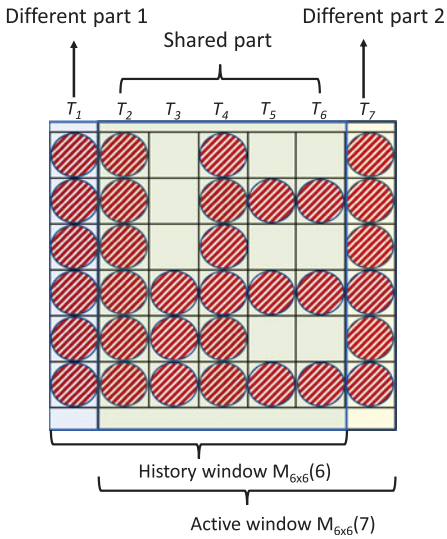


Fig. 6. Relationship of two adjacent window.

VII. Adaptive Sampling Schedule and Quick Recovery in Window Sliding Phase

Given a historical window, our goal is to design a sampling scheduling algorithm that can select a subset of OD pairs to take measurements in the upcoming time slot t ($t > T$) while ensuring the accurate reconstruction of un-measured data after data are collected. As there is no data information about the upcoming time slot, such a task is difficult.

As the upcoming time slot t ($t > T$) will be included in the active window, in this section, we first provide a mechanism to accurately estimate the rank of the active window, based on which we design the sampling scheduling algorithm.

A. Rank Estimation

Before presenting our sampling scheduling algorithm for each upcoming time slot, we first explore the rank relationship between two adjacent window matrices. Fig. 6 shows two adjacent windows. Only one column is different in any of the two adjacent windows, so the rank of these two adjacent windows should have a strong relationship. In Fig. 6, two matrices $\mathbf{M}_{6\times6}(6)$ and $\mathbf{M}_{6\times6}(7)$ in two adjacent windows share five same columns, i.e., $\{\mathbf{M}_2, \mathbf{M}_3, \mathbf{M}_4, \mathbf{M}_5, \mathbf{M}_6\}$. Only one column from each of the adjacent matrices is different, that is, $\mathbf{M}_1$ in $\mathbf{M}_{6\times6}(6)$ and $\mathbf{M}_7$ in $\mathbf{M}_{6\times6}(7)$.

In the following Theorem 7.1, we will show that the rank relationship of two adjacent windows does not change dramatically. In this section, we will use these properties in Theorem 7.1 to design our sampling scheduling algorithm.

Theorem 7.1: Given two matrices of two adjacent windows $\mathbf{M}_{N\times T}(t-1)$ and $\mathbf{M}_{N\times T}(t)$, and assume $\mathrm{rank}(\mathbf{M}_{N\times T}(t-1)) = r$, then the rank of the matrix $\mathbf{M}_{N\times T}(t)$ satisfies

$$r - 1 \leq \mathrm{rank}(\mathbf{M}_{N\times T}(t)) \leq r + 1 \tag{7}$$

Proof: Because $\mathbf{M}_{N\times T}(t-1)$ and $\mathbf{M}_{N\times T}(t)$ are the two matrices of the adjacent windows, we can see that $\mathbf{M}_{N\times T}(t-1) = (\mathbf{M}_{t-T}, \mathbf{M}_{N\times(T-1)}(t-1))$ and $\mathbf{M}_{N\times T}(t) = (\mathbf{M}_{N\times(T-1)}(t-1), \mathbf{M}_t)$ where $\mathbf{M}_{t-T} \in R^N$ and $\mathbf{M}_t \in R^N$ are non-vanishing vectors with $rank(\mathbf{M}_{t-T}) = rank(\mathbf{M}_t) = 1$, and we have:

$$\begin{aligned} &rank\big(\mathbf{M}_{N\times(T-1)}(t-1)\big) \\ \leq &rank\big(\mathbf{M}_{t-T}, \mathbf{M}_{N\times(T-1)}(t-1)\big) \\ \leq &rank(\mathbf{M}_{N\times(T-1)}(t-1)) + 1 \end{aligned} \tag{8}$$

As $rank(\mathbf{M}_{N\times T}(t-1)) = r$ and $\mathbf{M}_{N\times T}(t-1) = (\mathbf{M}_{t-T}, \mathbf{M}_{N\times(T-1)}(t-1))$, we can bound $rank(\mathbf{M}_{N\times(T-1)}(t-1))$ by:

$$r - 1 \leq rank(\mathbf{M}_{N\times(T-1)}(t-1)) \leq r \tag{9}$$

As $\mathbf{M}_{N\times T}(t) = (\mathbf{M}_{N\times(T-1)}(t-1), \mathbf{M}_t)$, we have

$$\begin{aligned} &rank\big(\mathbf{M}_{N\times(T-1)}(t-1)\big) \\ \leq &rank\big(\mathbf{M}_{N\times(T-1)}(t-1), \mathbf{M}_t\big) \\ \leq &rank\big(\mathbf{M}_{N\times(T-1)}(t-1)\big) + 1 \end{aligned} \tag{10}$$

Base on the fact that $\mathbf{M}_{N\times T}(t) = (\mathbf{M}_{N\times(T-1)}(t-1), \mathbf{M}_t)$, combining (9) and (10), we can obtain

$$r - 1 \leq rank(\mathbf{M}_{N\times T}(t)) \leq r + 1 \tag{11}$$

which completes the proof. ■

B. Straightforward Sampling and its Problem

According to Theorem 7.1, given the rank of the previous window (i.e., r), the active window that consists of the column of the upcoming time slot will be in the range of $[r-1, r+1]$.

As the rank of the active window is at most $r+1$, taking $|\Omega| \geq c\mu(\mathbf{U})(r+1)\log((r+1)/\delta)$ random samples in the upcoming time slot is sufficient to identify whether the measurement data in this time slot can be represented by the subspace of the active window or not. Therefore, a straightforward sampling in the window sliding phase can be scheduled to work as follows:

1) Among N OD pairs, randomly select $c\mu(\mathbf{U})(r+1)\log((r+1)/\delta)$ OD pairs to take measurement samples in the upcoming time slot t, with the sample location set denoted as Ω and measurement data as $\mathbf{M}_{\Omega t}$
2) For the upcoming time slot t, check if corresponding column $\mathbf{M}_{\Omega t}$ can be represented by the current subspace of the active window $\hat{\mathbf{U}}$, i.e.,

$$res = \frac{\left\|\mathbf{M}_{\Omega t} - \hat{\mathbf{U}}_\Omega(\hat{\mathbf{U}}_\Omega{}^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega{}^T\mathbf{M}_{\Omega t}\right\|}{|\Omega|} \leq \eta \tag{12}$$

where $res = \frac{\|\mathbf{M}_{\Omega t} - \hat{\mathbf{U}}_\Omega(\hat{\mathbf{U}}_\Omega{}^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega{}^T\mathbf{M}_{\Omega t}\|}{|\Omega|}$ is the residual of the upcoming column projected to the current subspace, η is a threshold.

- If the residual is not larger than η, we can identify that the upcoming time slot can be represented by the current

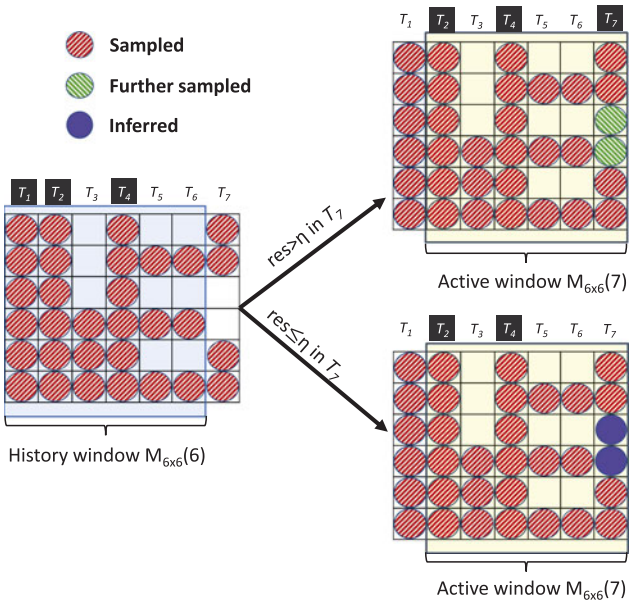


Fig. 7. Take further measurement samples or reconstruct the un-measured data according to the residual.

subspace. Thus, the un-measured data can be reconstructed through $\hat{\mathbf{M}}_t = \hat{\mathbf{U}}(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega^T\mathbf{M}_{\Omega t}$ in (5).

- Otherwise, the upcoming column can not be represented by the subspace, and the following procedures should be taken:
  a) Further measuring the remaining OD pairs in the time slot t.
  b) Marking the column $\mathbf{M}_t$ as the subspace column, and inserting $\mathbf{M}_t$ into the subspace column set $\mathbf{S}$.
  c) Performing skinny SVD on the updated $\mathbf{S}$ with $\mathbf{S} = \mathbf{U}_{|\mathbf{S}|}\mathbf{\Sigma}_{|\mathbf{S}|}\mathbf{V}_{|\mathbf{S}|}^T$, updating the subspace $\hat{\mathbf{U}}$ by setting $\hat{\mathbf{U}} = \mathbf{U}_{|\mathbf{S}|}$.

Fig. 7 shows an example to illustrate the above process. After we get the measurement data in T_7, we use the residual to determine whether the column can be recovered. If $res > \eta$, which means that $\mathbf{M}_7$ can not be well represented by the current subspace, we should take additional measurements from the remaining OD pairs in the time slot T_7, otherwise, the un-measured data in T_7 can be reconstructed by $\hat{\mathbf{M}}_7 = \hat{\mathbf{U}}(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega^T\mathbf{M}_{\Omega 7}$. There is a trade-off to set the threshold η. If η is too small, lots of columns will be selected as subspace columns, which would cause a high sampling cost. Otherwise, if η is too large, the subspace selected can not track the rank well, which may further compromise the reconstruction accuracy. In the experimental Section IX, we will do experiments to set a proper η.

C. Refining Rank Estimation and Adapting the Sampling Scheduling

Although the above sampling scheduling algorithm is promising, it may introduce a high sampling cost by using the up-bound $r+1$ of the estimated rank to make the sampling schedule. The use of $r+1$ may over-estimate the rank of the active window in some cases.

Algorithm 2: Window Sliding Phase

1: **for** each upcoming time slot t with its column $\mathbf{M}_t$ $(t > T)$ **do**
2: Apply Theorem 7.1 to estimate the up-bound rank of the active window as $r+1$ where r is the rank of history window.
3: **if** the oldest column $\mathbf{M}_{t-T}$ is a subspace column **then**
4: Refine the rank of the active window to r.
5: Randomly choose $|\Omega| = c\mu(\mathbf{U})r\log(r/\delta)$ OD pairs to take measurement.
6: Delete $\mathbf{M}_{t-T}$ from $\mathbf{S}$, update $\hat{\mathbf{U}}$ by performing skinny SVD on $\mathbf{S}$ with $\mathbf{S} = \mathbf{U}_{|\mathbf{S}|}\mathbf{\Sigma}_{|\mathbf{S}|}\mathbf{V}_{|\mathbf{S}|}^\mathbf{T}$, set $\hat{\mathbf{U}} \leftarrow \mathbf{U}_{|\mathbf{S}|}$.
7: **else**
8: Randomly choose $|\Omega| = c\mu(\mathbf{U})(r+1)\log((r+1)/\delta)$ OD pairs to take measurement.
9: **end if**
10:
11: **if** $\frac{\|\mathbf{M}_{\Omega t} - \hat{\mathbf{U}}_\Omega(\hat{\mathbf{U}}_\Omega{}^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega{}^T\mathbf{M}_{\Omega t}\|}{|\Omega|} \le \eta$ **then**
12: $\hat{\mathbf{M}}_t \leftarrow \hat{\mathbf{U}}(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega^T\mathbf{M}_{\Omega t}$.
13: **else**
14: Take full measurement in $\mathbf{M}_t$.
15: $\mathbf{S} \leftarrow [\mathbf{S}\ \mathbf{M}_t]$, mark $\mathbf{M}_t$ as subspace column, perform skinny SVD on $\mathbf{S}$ with $\mathbf{S} = \mathbf{U}_{|\mathbf{S}|}\mathbf{\Sigma}_{|\mathbf{S}|}\mathbf{V}_{|\mathbf{S}|}^\mathbf{T}$, set $\hat{\mathbf{U}} \leftarrow \mathbf{U}_{|\mathbf{S}|}$.
16: Set the rank $r \leftarrow |\mathbf{S}|$.
17: **end if**
18: **end for**

Let us look at the example in Fig. 8. In this example, the historical window $\mathbf{M}_{6\times6}(9)$ has two columns marked as the subspace columns, i.e., $\mathbf{S} = \{\mathbf{M}_4, \mathbf{M}_7\}$. So the rank of the history window $\mathbf{M}_{6\times6}(9)$ is considered to be 2. Directly applying the Theorem 7.1, the estimated rank of the active window $\mathbf{M}_{6\times6}(10)$ will fall into the range of $[1, 3]$, with the up-bound 3. However, as the window moves, the oldest subspace column $\mathbf{M}_4$ moves out of the window. So only one subspace column $\mathbf{S} = \{\mathbf{M}_7\}$ is left in the active window. In this case, even if we consider the upcoming column $\mathbf{M}_{10}$ as a subspace column, the active window $\mathbf{M}_{6\times6}(10)$ will only have at most two subspace columns. We call this problem the **overestimation problem**.

Instead of directly applying Theorem 7.1 to estimate the rank of an active window, we make the following strategy to refine the rank estimated.

- Given the rank of the historical window r, the up-bound of the rank estimated for the active window through Theorem 7.1 is $r+1$.
- If the oldest column in the history window is a subspace one, the rank of the active window is set to r.

With the refined rank, we design the algorithm for the sampling scheduling and un-measured data reconstruction in **Algorithm 2**. After we identify the rank of the active window, we will select random OD pairs to take measurements in Step 5 and Step 8.

D. Strategy for Fast Un-Measured Data Reconstruction

Fast reconstruction of the un-measured data is essential for network applications. Different from traditional matrix completion algorithms which usually require a process with

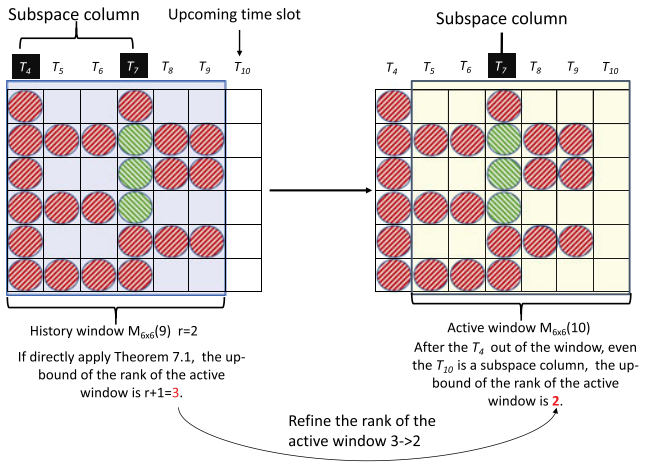


Fig. 8. An example of refine the rank of the active window.

multiple iterations to train the parameters, our subspace-based matrix completion is fast and can reconstruct the un-measured data without iterations through $\hat{\mathbf{M}}_t \leftarrow \hat{\mathbf{U}}(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega^T\mathbf{M}_{\Omega t}$ as shown in Step 12 in **Algorithm 2**.

However, $\hat{\mathbf{U}}(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega^T\mathbf{M}_{\Omega t}$ involves a matrix inverse operation $(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}$, which is of high computation cost. We use the following example to show the possibility of further speeding up the reconstruction process.

In Fig. 9, if we take samples $\Omega = \{2,3,5\}$ in T_8 and $\mathbf{M}_8$ can be represented by the subspace of the historical window $\mathbf{M}_{6\times6}(8)$, the un-measured data in T_8 can be reconstructed as $\hat{\mathbf{M}}_8 = \hat{\mathbf{U}}(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega^T\mathbf{M}_{\Omega 8}$ where $\mathbf{M}_{\Omega 8}$ is the measurement data in T_8 and $\hat{\mathbf{U}}$ is the subspace in the history window $\mathbf{M}_{6\times6}(8)$. Let $A = \hat{\mathbf{U}}(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega^T$. If we take the same sampling locations in T_9, i.e., $\Omega = \{2,3,5\}$, to reconstruct the un-measured data in T_9, we can perform $\hat{\mathbf{M}}_9 = \hat{\mathbf{U}}'(\hat{\mathbf{U}}_\Omega'^T\hat{\mathbf{U}}'_\Omega)^{-1}\hat{\mathbf{U}}_\Omega'^T\mathbf{M}_{\Omega 9}$ where $\mathbf{M}_{\Omega 9}$ is the measurement data in T_9 and $\hat{\mathbf{U}}'$ is the subspace of the active window $\mathbf{M}_{6\times6}(9)$. As shown in Fig 9, as the subspace columns are the same in these adjacent windows i.e., $\mathbf{M}_{6\times6}(8)$ and $\mathbf{M}_{6\times6}(9)$, easily we have $\hat{\mathbf{U}} = \hat{\mathbf{U}}'$. As $A = \hat{\mathbf{U}}(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega^T$ is calculated when reconstructing $\mathbf{M}_8$, $\mathbf{M}_9$ can be quickly reconstructed through $\hat{\mathbf{M}}_9 = A\mathbf{M}_{\Omega 9}$.

Based on the observation of the above example, we propose the following strategy to increase the processing speed.

- Denoting $\hat{\mathbf{U}}$ as the subspace of the history window. Let $\mathcal{P}_{\hat{\mathbf{U}}_\Omega} = \hat{\mathbf{U}}_\Omega(\hat{\mathbf{U}}_\Omega{}^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega{}^T$ and $A = \hat{\mathbf{U}}(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega^T$, where Ω is the location set sampled in the last column of the history window.
- When window slides, if the two adjacent windows share the same subspace columns, **using the same locations in Ω to take measurement samples in the upcoming time slot** can reuse the $\mathcal{P}_{\hat{\mathbf{U}}_\Omega}$ and A that are calculated in the historical window to speed up the process in the active window. That is, the Step 11-12 in **Algorithm 2** can be rewritten as: If $\frac{\|\mathbf{M}_{\Omega t}-\mathcal{P}_{\hat{\mathbf{U}}_\Omega}\mathbf{M}_{\Omega t}\|}{|\Omega|} \le \eta$, then $\hat{\mathbf{M}}_t \leftarrow A\mathbf{M}_{\Omega t}$ where $\mathcal{P}_{\hat{\mathbf{U}}_\Omega}$ and A that are calculated in the historical window.

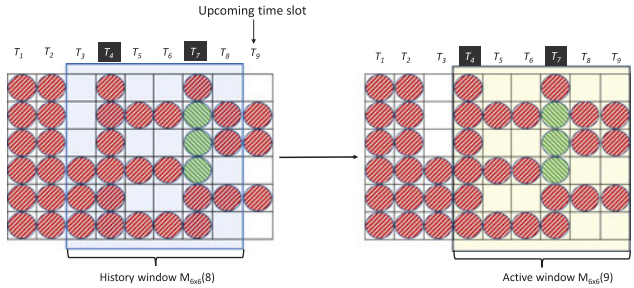


Fig. 9. Speed up case in window sliding phase.

E. Complexity Analysis

As the window slides, two procedures are taken, removing the data of the oldest time slot and adding the data of the upcoming time slot.

1) Removing the Data of the Oldest Time Slot: If the oldest column $\mathbf{M}_{t-T}$ is a subspace column, we need to update the current subspace $\hat{\mathbf{U}}$ by performing a skinny SVD on $\mathbf{S} \in \mathbb{R}^{N\times r-1}$, which incurs the $\mathcal{O}(Nr^2)$ time complexity.

2) Adding the Data of the Upcoming Time Slot: For the upcoming time slot t, we first need to identify whether the corresponding column $\mathbf{M}_t$ can be represented by the current column space with $\frac{\|\mathbf{M}_{\Omega t}-\hat{\mathbf{U}}_\Omega(\hat{\mathbf{U}}_\Omega{}^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega{}^T\mathbf{M}_{\Omega t}\|}{|\Omega|}$. As the complexity of $(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}$ is $\mathcal{O}(|\Omega|r^2)$, and the complexity of $\hat{\mathbf{U}}_\Omega(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega^T\mathbf{M}_{t\Omega}$ is $\mathcal{O}(|\Omega|^2r)$, we can easily find that the time complexity of the identification process is $\mathcal{O}(|\Omega|r^2 + |\Omega|^2r)$.

- If $\mathbf{M}_t$ can be represented by the current column space $\hat{\mathbf{U}}$, we just need to use $\hat{\mathbf{U}}(\hat{\mathbf{U}}_\Omega^T\hat{\mathbf{U}}_\Omega)^{-1}\hat{\mathbf{U}}_\Omega^T\mathbf{M}_{t\Omega}$ in (5) to reconstruct it. The complexity of the reconstruction is $\mathcal{O}(N|\Omega|r)$.
- Otherwise, $\mathbf{M}_t$ is a subspace column and we need to update the current subspace $\hat{\mathbf{U}}$ by performing the skinny SVD on a $N \times r$ matrix, which requires $\mathcal{O}(Nr^2)$ complexity.

Therefore, the time complexity for each time slot is $\mathcal{O}(Nr^2 + (|\Omega|r^2 + |\Omega|^2r) + max(N|\Omega|r, Nr^2))$. As $N >> |\Omega| > r$, it can be further rewritten as $\mathcal{O}(N|\Omega|r)$, which grows linearly with the size of N.

VIII. Traffic Measurements With the Sampling Scheduling

In this section, we describe how to apply our measurement scheduling scheme for practical use.In the training phase, all OD pairs perform the traffic measurements and send the measured data to the controller. After collecting data, the controller can calculate the basic subspace. In the window sliding phase, as shown in Fig. 10, the workflow of our sampling scheduling scheme for each time slot can be divided into three steps:

- Scheduling of probing: the controller selects a set of OD pairs to take probing measurements based on the maintained subspace.
- Path probing: the selected network nodes perform the measurements following the probing schedule.

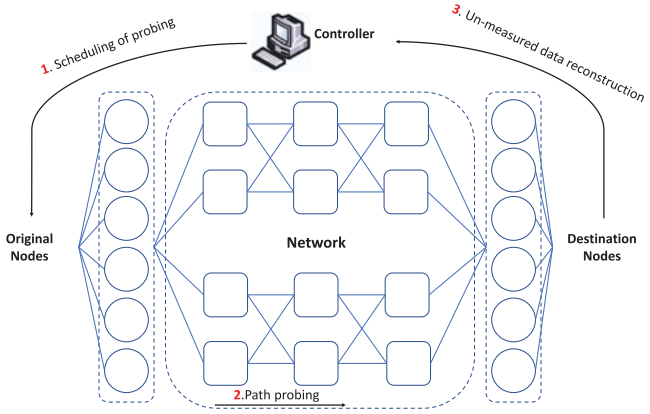


Fig. 10. Workflow of our sampling scheduling scheme for each time slot.

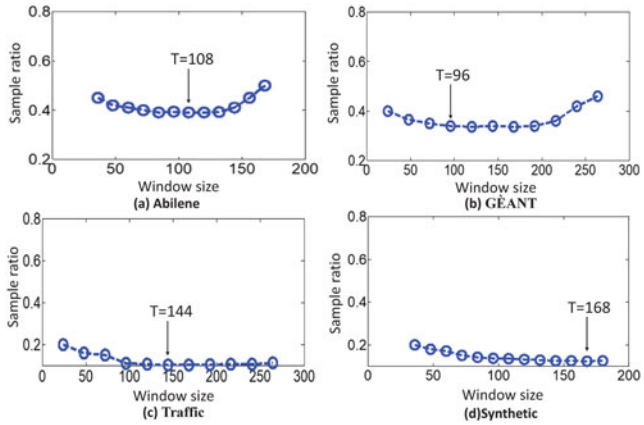


Fig. 11. Window size impact.

- Un-measured data reconstruction: after collecting the probing data, the controller reconstructs un-measured data of each OD pair through our method.

IX. Performance Evaluations

A. Experiment Setting

To evaluate the performance of our algorithm, we conduct a series of experiments on two network traffic data sets. Moreover, to validate that our algorithm is general and can be applied for online sampling scheduling in other scenarios, we also use a real road traffic speed data set and a synthetic data set to evaluate the performance.

- Abilene [17]: Abilene network consists of 12 routers, thus 144 OD pairs, and the data set contains the traffic data collected every 5 minutes in 24 weeks.
- GÈANT [18]: GÈANT network consists of 23 routers, thus 529 OD pairs, and the trace contains the traffic data collected every 15 minutes for several months.
- Traffic [19]: This data set contains the road traffic speed data collected every 10 minutes for two months in Guangzhou, China, from anonymous road segments.
- Synthetic: This data set is generated through the following steps: 1) generating a low-rank matrix $\mathbf{M}$ with its size being 1000×10000; 2) randomly selecting 100 columns and changing their values to simulate the varying rank scenario; 3) generating a Gaussian noise matrix $\mathbf{Z}$ and injecting it into $\mathbf{M}$. Here we limit $\|\mathbf{Z}\|_F / \|\mathbf{M}\|_F = 0.2$, where $\|\cdot\|_F$ denotes the Frobenius norm.

We use the reconstruction error ratio of the un-measured data to evaluate the accuracy of the implemented algorithms. The reconstruction error ratio is calculated as $\frac{\sqrt{\sum_{(i,j)\in\bar{\Omega}}(\mathbf{M}_{i,j}-\hat{\mathbf{M}}_{i,j})^2}}{\sqrt{\sum_{(i,j)\in\bar{\Omega}}(\mathbf{M}_{i,j})^2}}$, where $\mathbf{M}_{i,j}$ and $\hat{\mathbf{M}}_{i,j}$ denote the raw data and the reconstructed data at the element (i, j) of the target matrix $\mathbf{M}$ respectively. $\bar{\Omega}$ denotes the set of un-measured element indices. This metric is the relative error for the element locations with the values inferred from the matrix completion algorithms. To evaluate the speed of the different sampling algorithms and matrix completion algorithms, we use the metric **processing time** to measure the average number of seconds to perform the task.

Data normalization [29] is often used in data preprocessing to scale the data into the range [0,1]. We preprocess the above data sets in the following manner, normalizing the data set by:

$$M_{i,j} = \frac{M_{i,j} - \min\{M_{i,j}\}}{\max\{M_{i,j}\} - \min\{M_{i,j}\}} \tag{13}$$

where the $\min\{M_{i,j}\}$ is the minimum value and $\max\{M_{i,j}\}$ is the maximum value of the data sets. After using the normalized data to get the reconstruction results, we perform the inverse of normalization on the reconstruction results to calculate the inferred error ratio.

All algorithms are implemented using Matlab, and all the experiments are run on a laptop with CPU Intel(R) Core(TM) i7-8750H CPU@2.20GHz and 24GB memory.

B. Impact of Window Size

We use 1000 time slots data to investigate how the parameter window size T impacts reconstruction performance. First, we set an error ratio threshold for each data set, i.e., 0.2 in Abilene, GÈANT, and Synthetic, 0.1 in Traffic. Then, we vary the window sizes to explore how many samples are required to take with different window sizes to make the error ratio less than or close to the threshold. The results are shown in Fig. 11.

From Fig. 11, we can see that 1) with the increase of T, the required sampling ratio decreases; 2) after T reaches a value, with the increase of T, the required sampling ratio gradually increases. It is because when T is small, the required sampling ratio is high, as the window contains too many subspace columns. When T is large, the window contains too many data, resulting in a decrease in the correlation between the reconstructed columns and the subspace columns, so more samples are needed to maintain the accuracy of the reconstruction.

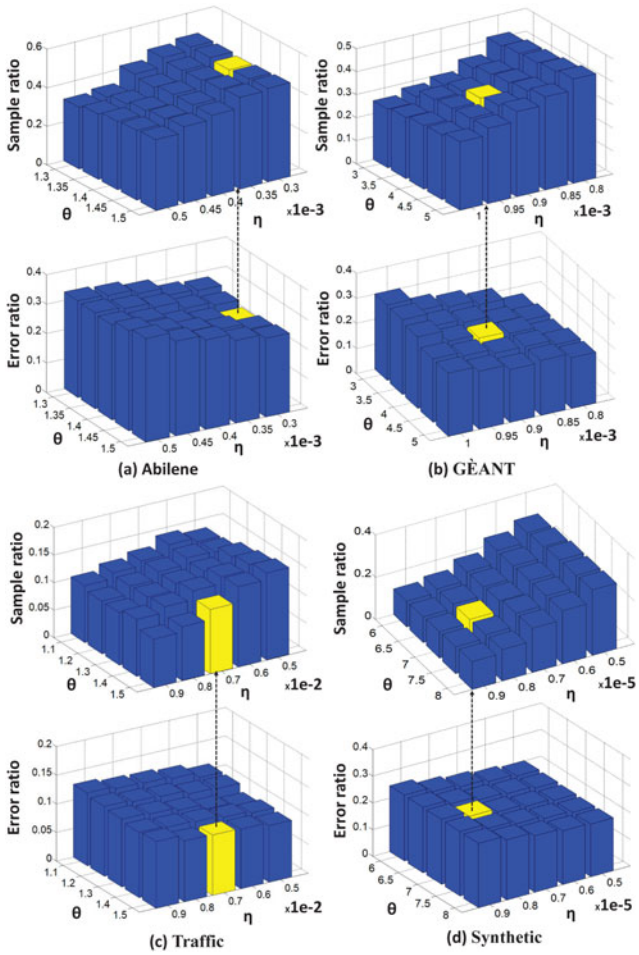


Fig. 12. Parameter impact.

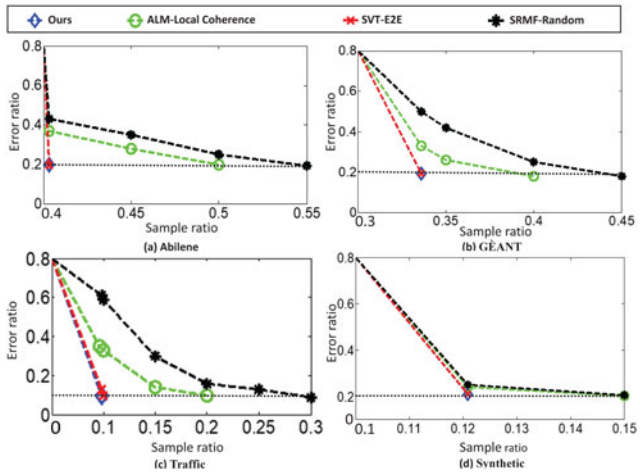


Fig. 13. Reconstruction error in a window.

According to the results in Fig. 11, the window setting is listed as follows. We set $N = 144$ and $T = 108$ in Abilene, $N = 529$ and $T = 96$ in GÈANT, $N = 214$ and $T = 144$ in Traffic, and $N = 1000$ and $T = 168$ in Synthetic, respectively.

C. Impact of Parameters

According to **Algorithm 2**, the parameter η determines whether an upcoming column $\mathbf{M}_t$ is a subspace column. If yes, the algorithm take full measurements for $\mathbf{M}_t$; otherwise, $|\Omega|$ entries in $\mathbf{M}_t$ are selected to take measurements. Therefore, the parameters that affect the sampling ratio are $|\Omega|$ and η.

As $|\Omega| = c\mu(\mathbf{U})r\log(r/\delta)$, we have $|\Omega| = c\mu(\mathbf{U})r(\log(r) - \log(\delta))$. As the parameter δ reflects the probability of failure reconstruction for a column $\mathbf{M}_t$, whether $\mathbf{M}_t$ can be reconstructed only depends on if $(\mathbf{U}_\Omega^T\mathbf{U}_\Omega)$ is invertible according to **Algorithm 2**. In real traffic data sets, $(\mathbf{U}_\Omega^T\mathbf{U}_\Omega)$ is usually invertible due to the presence of noise. Therefore, we just consider $|\Omega| = c\mu(\mathbf{U})r\log(r)$ without parameter δ.

Enlarging $\mu(\mathbf{U})$ by a constant factor c, if we set θ equal to $c\mu(\mathbf{U})$, the change of θ is equivalent to changing the value of c and $\mu(\mathbf{U})$. For visualization purposes, we use θ as a joint parameter for c and $\mu(\mathbf{U})$. To investigate how the parameters θ and η impact the reconstruction performance, we implement our algorithm in the T time slots data (i.e., a window's data) following the first training window. Fig. 12 draws the error ratio and sample ratio by varying θ and η.

As expected, with the decrease of η, the sample ratio increases because more time slots take full measurements, so the error ratio decreases accordingly at a larger measurement cost. In Fig. 12(a)(b), we choose the parameter corresponding to the lowest sampling ratio when the error ratio is less than 0.2 for data sets Abilene and GÈANT. In Fig. 12(c), we choose the parameter that corresponds to the lowest sampling ratio when the error ratio is less than 0.1 for data set Traffic. In Fig. 12(d), we choose the parameter corresponding to the lowest sample ratio when the error ratio is close to 0.2 for the data set Synthetic. We use yellow to mark the above results. Consequently, in Abilene, we set $\theta = 1.4$ and $\eta = 0.0003$, in GÈANT, we set $\theta = 4$ and $\eta = 0.0009$, in Traffic, we set $\theta = 1.5$ and $\eta = 0.007$, in Synthetic, we set $\theta = 7$ and $\eta = 0.000008$ for the following experiments.

D. Performance Comparison

1) Comparison With Dynamic Sampling Schemes: For comparison, we implement three sampling schedules and un-measured data reconstruction schemes.

The first scheme (called ALM-Local Coherence) performs local coherence sampling [23] along with ALM (Augmented Lagrangian Method) [22] based reconstruction algorithm. The second scheme (called SVT-E2E) performs E2E [9] sampling along with SVT [11] based reconstruction algorithm. The third scheme (called SRMF-Random) performs random sampling along with SRMF [2] as the reconstruction algorithm. Our scheme can handle the sampling scheduling and un-measured data inferring column by column for online execution, while the peer algorithms can only support data inferring in a whole matrix, and can

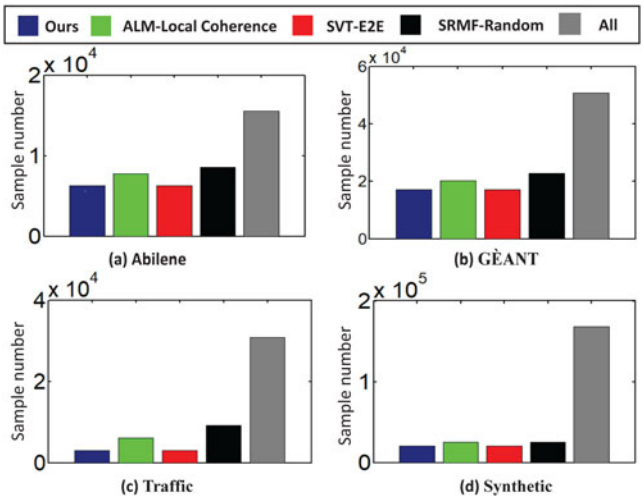


Fig. 14. Number of samples to achieve the same reconstruction accuracy.

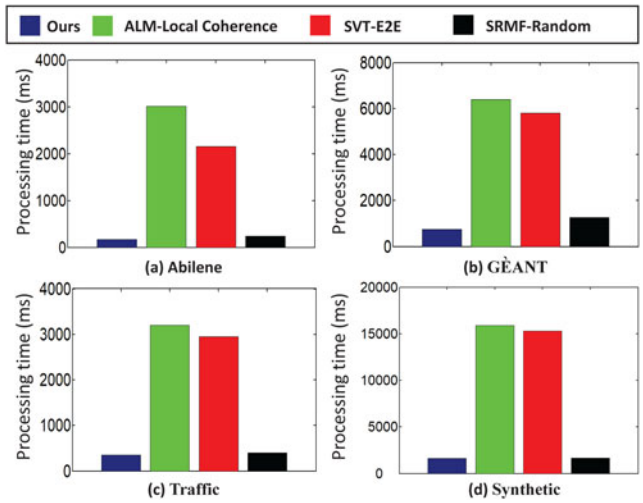


Fig. 15. Processing time to achieve the same reconstruction accuracy.

not support such operations executed column by column. For fair comparison, we count the performance of a whole window, i.e., a matrix.

Fig. 13 shows the reconstruction accuracy under different data sets. The X-axis is the sample ratio. The Y-axis denotes the error ratio. To study the reconstruction accuracy of the other compared schemes, we increase the sampling ratio until the error ratio drops to that under our algorithms.

As shown in Fig. 13, among all algorithms implemented, to achieve the same reconstruction accuracy with error ratio (0.2 in Abilene, GÈANT, and Synthetic, 0.1 in Traffic), the sampling ratio under our algorithm and SVT-E2E are the lowest. Moreover, from Fig. 15, we find that the processing time under SVT-E2E is nearly 8 times that under our algorithm. This is because the sampling schedule and the un-measured data reconstruction are quick in our scheme. In contrast, the E2E sampling algorithm and SVT reconstruction all involve iterative operations, which is time-consuming.

We also draw Fig. 14 to show the number of samples needed to achieve the same reconstruction error ratio (i.e., 0.2 in Abilene, GÈANT, and Synthetic, 0.1 in Traffic). As a reference, excluding the three compared schemes, we show the number of full measurements in a window (denote as 'ALL' in Fig. 14). As shown in Fig. 14, our algorithm and SVT-E2E have the lowest sample number compared with other schemes.

We draw Fig. 15 to show the processing time to achieve the same reconstruction error ratio in a window. The processing time includes the time of sampling scheduling and un-measured data reconstruction. As shown in Fig. 15, our algorithm is the fastest scheme. It only takes 170 ms in Abilene, 744 ms in GÈANT, 351 ms in Traffic, and 1612 ms in Synthetic to obtain the data in a window through the sampling and reconstruction process.

2) Comparison With Fixed Sampling Schemes: Besides dynamic sampling schemes in Section IX-D1, we also compare our scheme with four fixed sampling schemes under uniform sampling. Different from our scheme, these schemes fix the sampling ratio for each column no matter how the current

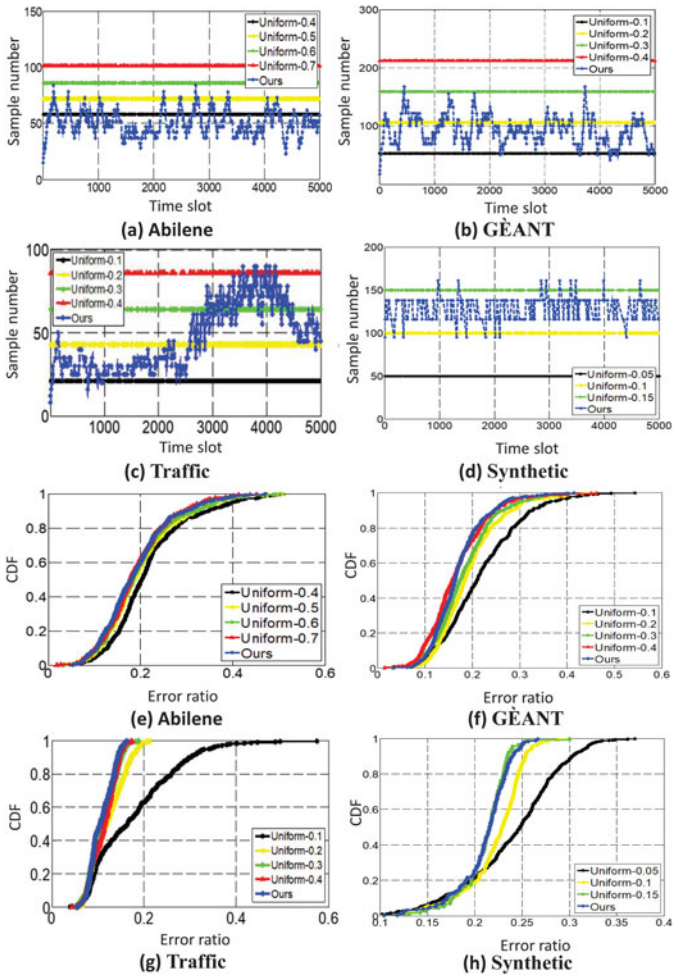


Fig. 16. Compared with fixed sampling schemes.

subspace changes, while for our sampling algorithm, the number of samples changes according to data variation. In this experiment, we use $\frac{\|\mathbf{M}_{\Omega}-\hat{\mathbf{M}}_{\Omega}\|}{\|\mathbf{M}_{\Omega}\|}$ to metric the inferred error ratio

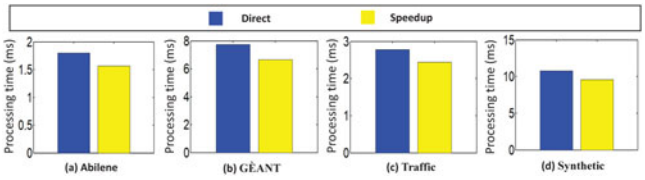


Fig. 17. Validation of the speedup strategy.

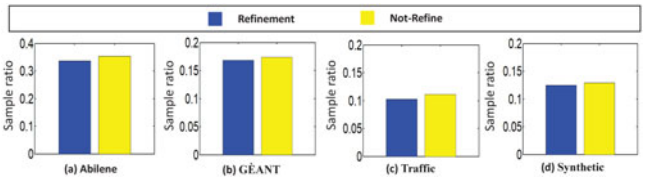


Fig. 19. Validation of the rank refinement strategy.

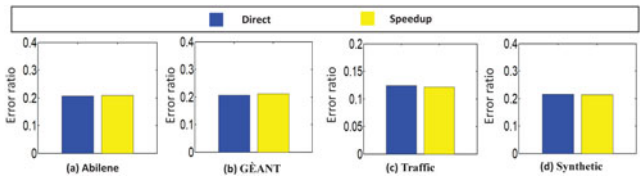


Fig. 18. Mean value of the error ratio of the reconstructed column (speedup strategy).

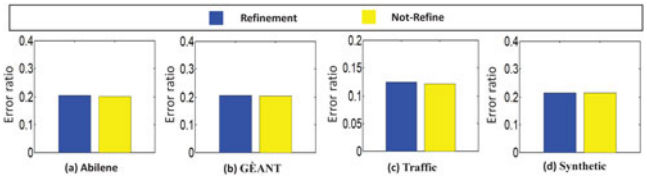


Fig. 20. Mean value of the error ratio of the reconstructed column (rank refinement strategy).

for each reconstructed column $\mathbf{M}_t$, where $\bar{\Omega}$ denotes the set of un-measured data indices in $\mathbf{M}_t$.

We draw Figs. 16(a)(b)(c)(d) to show the number of samples needed for each time slot when the subspace changes in 5000 time slots after the training window. We also draw the cumulative distribution function (CDF) of the error ratio in these 5000 time slots in Fig. 16(e)(f)(g)(h), where the error ratio represents the average of the reconstruction error of subsequent reconstructed columns when the subspace changes.

Combining Figs. 16(a) with Fig. 16(e), we find: 1) the sampling ratio under our scheme dynamically changes according to the data variation; 2) although the sampling ratio is much less than 0.7, our scheme can achieve the similar accuracy with Uniform-0.7. The similar results can be found by combining Figs. 16(b)(c)(d) and 16(f)(g)(h). These experimental results demonstrate that our scheme can capture the data variation feature to adaptively change the sampling ratio while accurately recovering the un-measured data.

3) Validation of Reconstruction Speedup Strategy: In Section VII-D, we propose a strategy to speed up the reconstruction of un-measured data. To validate the effectiveness of the strategy, we perform two kinds of implementations. In the first kind of implementation (termed **Direct**), we directly adopt Algorithm 2 to perform sampling schedule and un-measured data reconstruction. In the second kind of implementation (termed **Speedup**), we adopt the strategy for fast un-measured data reconstruction proposed in Section VII-D along with Algorithm 2 to perform sampling schedule and un-measured data reconstruction.

Figs. 17 and 18 show the processing time and the reconstruction error under these two kinds of implementations. In this experiment, we test 5000 time slots. We draw Fig. 17 to show the average process time to handle each time slot, and Fig. 18 to show the average error ratio of the columns corresponding to these 5000 time slots. From Fig. 18, we find that the error ratio under the two strategies is similar.

In Fig. 17, in the Abilene data set, the average processing time in each time slot is 1.8 ms under the direct scheme, while it is 1.57 ms under the speedup scheme. The runtime performance improvement is approximately 12.7%. The GÈANT data set takes 7.7 ms and 6.6 ms under the direct and speedup schemes, respectively. The improvement is roughly 14.3%. In the Traffic data set, the average processing time in each time slot is 2.78 ms and 2.44 ms in the direct and speedup schemes. The improvement is approximately 12.2%. The Synthetic data set takes 10.8 ms under the direct scheme and 9.6 ms under speedup scheme. The improvement is roughly 11.1%. These results demonstrate that the strategy proposed in Section VII-D is effective to speed up the reconstruction process of the un-measured data.

4) Validation of the Rank Refinement Strategy: In the Section VII-C, we propose the rank refinement strategy to address the overestimation problem. We provide two kinds of implementations to verify the effectiveness of this strategy. In the first one (termed **Refinement**), we directly adopt Algorithm 2. In the second one (termed **Not-Refine**), we remove the refinement process from Algorithm 2.

As shown in Figs. 19 and 20, we draw the sampling ratio and the reconstruction error of the two kinds of implementations over 5000 time slots. Compared with the implementation without rank refinement, our rank refinement strategy reduces the sampling cost while achieving a similar reconstruction accuracy, which indicates that the rank refinement strategy effectively addresses the rank overestimation problem.

X. Conclusion

In this work, we focus on the online data measurement scheduling and reconstruction problem for real-world network traffic monitoring. By studying two real network data sets, Abilene and GÈANT, we find that although these data sets have a good low-rank structure, the rank varies with time. We design our scheme based on the sliding window model and propose a rank estimation method to estimate the rank of the current window. According to the rank estimated, we propose an adaptive sampling scheduling algorithm to select a subset

of OD pairs to take measurement samples in the upcoming time slot and reconstruct it by subspace-based matrix completion. For faster online measurement and reconstruction, we propose a speedup technique that reuses the results in the previous time slot to speed up the reconstruction of un-measured data in the current time slot. Extensive experiments on three real data sets show that our scheme guarantees the high-precision recovery of network measurement data with low measurement cost.

REFERENCES

[1] M. Mardani and G. B. Giannakis, "Robust network traffic estimation via sparsity and low rank," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 4529–4533.

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

[3] B. Claise, G. Sadasivan, and V. Valluri, "Cisco systems NetFlow services export version 9," Internet Eng. Task Force, Tech. Rep. RFC 3954, 2004.

[4] M. Wang, B. Li, and Z. Li, "sFlow: Towards resource-efficient and agile service federation in service overlay networks," in *Proc. IEEE 24th Int. Conf. Distrib. Comput. Syst.*, 2004, pp. 628–635.

[5] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with OpenSketch," in *Proc. 10th USENIX Symp. Netw. Syst. Des. Implementation*, 2013, pp. 29–42.

[6] R. Du, C. Chen, B. Yang, and X. Guan, "VANET based traffic estimation: A matrix completion approach," in *Proc. IEEE Glob. Commun. Conf.*, 2013, pp. 30–35.

[7] G. Gürsun and M. Crovella, "On traffic matrix completion in the internet," in *Proc. Internet Meas. Conf.*, 2012, pp. 399–412.

[8] Y.-C. Chen, L. Qiu, Y. Zhang, G. Xue, and Z. Hu, "Robust network compressive sensing," in *Proc. 20th Annu. Int. Conf. Mobile Comput. Netw.*, 2014, pp. 545–556.

[9] K. Xie et al., "Sequential and adaptive sampling for matrix completion in network monitoring systems," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2443–2451.

[10] S. Liu and Q. Wang, "Adaptive coherent sampling for network delay measurement," in *Proc. IEEE Int. Conf. Commun.*, 2020, pp. 1–6.

[11] J.-F. Cai, E. J. Candés, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. Optim.*, vol. 20, no. 4, pp. 1956–1982, 2010.

[12] C. Févotte and J. Idier, "Algorithms for nonnegative matrix factorization with the β-divergence," *Neural Comput.*, vol. 23, no. 9, pp. 2421–2456, 2011.

[13] A. Krishnamurthy and A. Singh, "Low-rank matrix and tensor completion via adaptive sampling," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 836–844.

[14] A. Krishnamurthy and A. Singh, "On the power of adaptivity in matrix completion and approximation," 2014, *arXiv:1407.3619*.

[15] M.-F. F. Balcan and H. Zhang, "Noise-tolerant life-long matrix completion via adaptive sampling," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2955–2963.

[16] Y. Wan, J. Yi, and L. Zhang, "Matrix completion from non-uniformly sampled entries," 2018, *arXiv:1806.10308*.

[17] Abilene Dataset. [Online]. Available: http://abilene.internet2.edu/observatory/data-collections.html

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

[19] X. Chen, Y. Chen, and Z. He, "Urban traffic speed dataset of Guangzhou, China," Mar. 2018. [Online]. Available: https://doi.org/10.5281/zenodo.1205229

[20] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 171–184, Jan. 2013.

[21] E. J. Candés and B. Recht, "Exact matrix completion via convex optimization," *Found. Comput. Math.*, vol. 9, no. 6, 2009, Art. no. 717.

[22] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," 2010, *arXiv:1009.5055*.

[23] Y. Chen, S. Bhojanapalli, S. Sanghavi, and R. Ward, "Coherent matrix completion," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 674–682.

[24] X. Guo, "Online robust low rank matrix recovery," in *Proc. 24th Int. Joint Conf. Artif. Intell.*, 2015, pp. 3540–3546.

[25] B. Lois and N. Vaswani, "Online matrix completion and online robust PCA," in *Proc. IEEE Int. Symp. Inf. Theory*, 2015, pp. 1826–1830.

[26] K. Xie et al., "Accurate and fast recovery of network monitoring data with GPU-accelerated tensor completion," *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1601–1614, Aug. 2020.

[27] K. Xie, X. Li, X. Wang, G. Xie, J. Wen, and D. Zhang, "Active sparse mobile crowd sensing based on matrix completion," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 195–210.

[28] I. Markovsky, *Low-Rank Approximation: Algorithms, Implementation, Applications*. Berlin, Germany: Springer, 2018.

[29] S. Aksoy and R. M. Haralick, "Feature normalization and likelihood-based similarity measures for image retrieval," *Pattern Recognit. Lett.*, vol. 22, no. 5, pp. 563–582, 2001.

Kai Jin is currently working toward the Ph.D. degree with the College of Computer Science and Electronics Engineering, Hunan University, Changsha, China. His research interests include matrix completion and traffic data analysis.

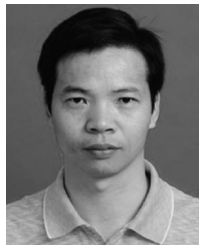
Kun Xie (Member, IEEE) received the Ph.D. degree in computer application from Hunan University, Changsha, China, in 2007. She is currently a Professor with Hunan University. She has authored or coauthored more than 60 articles in major journals and conference proceedings, including IEEE/ACM TON, IEEE TMC, IEEE TC, IEEE TWC, IEEE TSC, SIGMOD, INFOCOM, ICDCS, SECON, DSN, and IWQoS. Her research interests include network measurement, network security, Big Data, and AI.

Xin Wang (Senior Member, IEEE) received the Ph.D. degree in electrical and computer engineering from Columbia University, New York, NY, USA. She is currently an Associate Professor with the Department of Electrical and Computer Engineering, The State University of New York, Stony Brook, Stony Brook, NY, USA. Her research interests include algorithm and protocol design in wireless networks and communications, mobile and distributed computing, and networked sensing and detection. She was a Member of ACM in 2004. She was the recipientof the NSF Career Award in 2005 and the ONR Challenge Award in 2010.

Jiazheng Tian received the Ph.D. degree from the College of Computer Science and Electronics Engineering, Hunan University, Changsha, China, in 2022. He is currently a Postdoctoral Fellow with the College of Computer Science and Electronics Engineering, Hunan University. He has authored or coauthored some papers in major journal and conference proceedings (including journal IEEE/ACM ToN, and conferences INFOCOM). His research interests include tensor completion and traffic data analysis.

Gaogang Xie (Senior Member, IEEE) received the Ph.D. degree in computer science from Hunan University, Changsha, China, 2002. He is currently a Professor with Computer Network Information Center, Chinese Academy of Sciences, Beijing, China, and the University of Chinese Academy of Sciences, Beijing, China. His research interests include Internet architecture, packet processing and forwarding, and Internet measurement.

Jigang Wen received the Ph.D. degree in computer application from Hunan University, Changsha, China, in 2011. He was a Research Assistant with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, from 2008 to 2010. His research interests include network measurement and management.



Kenli Li (Senior Member, IEEE) received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 2003. He is currently a Cheung Kong Professor of computer science and technology with Hunan University, Changsha, China, the Dean of the College of Information Science and Engineering of Hunan University, and the Director with the National Supercomputing Center in Changsha. His major research interests include parallel and distributed processing, high-performance computing, and Big Data management.