Network Latency Estimation for Personal Devices: a Matrix Completion Approach

Rui Zhu, Bang Liu, *Student Member, IEEE,* Di Niu, *Member, IEEE,* Zongpeng Li, *Senior Member, IEEE,* and Hong Vicky Zhao, *Member, IEEE*

***Abstract*—Network latency prediction is important for server selection and quality-of-service estimation in real-time applications on the Internet. Traditional network latency prediction schemes attempt to estimate the latencies between all pairs of nodes in a network based on sampled round-trip times, through either Euclidean embedding or matrix factorization. However, these schemes become less effective in terms of estimating the latencies of personal devices, due to unstable and time-varying network conditions, triangle inequality violation and the unknown ranks of latency matrices. In this paper, we propose a matrix completion approach to network latency estimation. Specifically, we propose a new class of low-rank matrix completion algorithms which predicts the missing entries in an extracted "network feature matrix" by iteratively minimizing a weighted Schatten-p norm to approximate the rank. Simulations on true low-rank matrices show that our new algorithm achieves better and more robust performance than multiple state-of-the-art matrix completion algorithms in the presence of noise. We further enhance latency estimation based on multiple "frames" of latency matrices measured in the past, and extend the proposed matrix completion scheme to the case of 3D tensor completion. Extensive performance evaluations driven by real-world latency measurements collected from the *Seattle* platform show that our proposed approaches significantly outperform various state-of-the-art network latency estimation techniques, especially for networks that contain personal devices.**

***Index Terms*—Matrix Completion; Internet Latency Estimation; Personal Devices.**

I. Introduction

Network latency and proximity estimation has been an important topic in networking research that can benefit server selection, facility placement, and quality-of-service (QoS) estimation for latency-sensitive applications running on either desktops or mobile devices. A popular idea to estimate pairwise latencies in a large network is to partially measure end-to-end round-trip times (RTTs) between some nodes, based on which the latencies between all the nodes can be inferred.

Prior research on network latency prediction mainly falls into two categories: Euclidean embedding and matrix factorization. The Euclidean embedding approach (e.g., Vivaldi [1], GNP [2]) aims to map network nodes onto the coordinates in a Euclidean space or in a similar space with a predefined structure, such that their distances in the space predict their latencies. However, it is generally believed [3]–[5] that the triangle inequality may not hold for latencies among end users at the edge of the Internet, and thus the Euclidean assumptions do not hold. The matrix factorization approach [3] models an $n \times n$ network latency matrix M as the product of two factor matrices with lower dimensions, i.e., $M = UV^\top$, where $U \in \mathbb{R}^{n\times r}$ and $V \in \mathbb{R}^{n\times r}$, r being the rank of M. However, in reality, it is hard to know the exact rank r of the true latency matrix from noisy measurements. In fact, raw RTT measurements usually have full rank.

Network latency estimation is further complicated by the increasing popularity of personal devices, including laptops, smart phones and tablets [6]. Based on latency measurements collected from *Seattle* [7], which is an educational and research platform of open cloud computing and peer-to-peer computing consisting of laptops, phones, and desktops donated by users, we observe different characteristics of latencies as compared to those measured from desktops, e.g., from PlanetLab. *First*, not only do Seattle nodes have longer pairwise latencies with a larger variance, but there are also more observations of triangle inequality violation (TIV) and asymmetric RTTs in Seattle. *Second*, as many personal devices in Seattle mainly communicate wirelessly with less stable Internet connections, their pairwise latencies may vary substantially over time due to changing network conditions.

In this paper, we study the problem of network latency estimation for personal device networks, using a matrix completion approach to overcome the shortcomings of both Euclidean embedding and the fixed-rank assumption in latency matrices. Our contributions are manifold:

First, we propose a simple network feature extraction procedure that can decompose an incomplete $n \times n$ RTT measurement matrix M among n nodes into a complete *distance matrix* D that models the Euclidean component in latencies and an incomplete low-rank *network feature matrix* F that models correlated network connectivities. Matrix completion can then be applied to the noisy network feature matrix F to recover missing entries. Based on the analysis of measurements collected from Seattle, we show that the extracted network feature matrices have more salient low-rank properties than raw RTT matrices.

Second, to complete the extracted network feature matrix F, we solve a rank minimization problem without requiring *a priori* knowledge about the rank of F. We propose a new

Some preliminary results appeared in the 34th IEEE International Conference on Computer Communications (INFOCOM), Hong Kong, China, April 16–May 1, 2015. This work was supported by NSERC Discovery Grants. Thanks for the support of Wedge Networks Inc.

Rui Zhu, Bang Liu, Di Niu and H. Vicky Zhao are with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada. E-mail: {rzhu3, bang3, dniu}@ualberta.ca, vzhao@ece.ualberta.ca.

Zongpeng Li is with the Department of Computer Science, University of Calgary, Calgary, AB T2N 1N4, Canada. E-mail: zongpeng@ucalgary.ca.

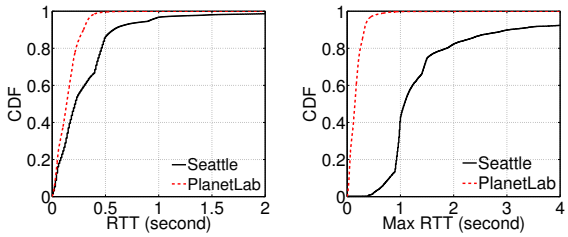
Digital Object Identifier 10.1109/TNET.2016.2612695

class of algorithms, called *Iterative weighted Schatten-p norm minimization* (IS-p), with $1 \leq p \leq 2$, to approximate rank minimization with weighted Schatten-p norm minimization, with $p = 1$ representing the nuclear norm that achieves better approximation to the rank, and $p = 2$ representing the Frobenius norm with more efficient computation. The proposed algorithm turns out to be a generalization of a number of previously proposed iterative re-weighted algorithms [8] based on either only the nuclear norm or only the Frobenius norm to a flexible class of algorithms that can trade optimization accuracy off for computational efficiency, depending on the application requirements. We prove that our algorithms can converge for any p between 1 and 2. Simulations based on synthesized low-rank matrices have shown that our algorithms are more robust than a number of state-of-the-art matrix completion algorithms, including Singular Value Thresholding (SVT) [9], Iterative Reweighted Least Squares Minimization (IRLS-p) [8] and DMFSGD Matrix Completion [3], in both noisy and noiseless scenarios.

Third, we propose to enhance the latency estimates for the current timeframe based on historical latencies via approximate tensor completion. Specifically, we model the evolving $n \times n$ latency matrices over different time periods as a 3D *tensor*, based on which the extracted 3D network feature tensor $\mathcal{F}$ has a certain "low-rank" property. Similar to rank minimization in matrix completion, to complete the missing entries in a tensor and especially those in the current timeframe, we minimize a weighted sum of the ranks of three matrices, each unfolded from the tensor along a different dimension. We then extend the proposed IS-p algorithm to solve this approximate tensor completion problem, which again leads to convex optimization that can be efficiently solved.

We perform extensive performance evaluation based on a large number of RTT measurements that we collected from both *Seattle* and *PlanetLab*. These datasets are made publicly available [10] for future research. We show that our proposed matrix completion approach with network feature extraction significantly outperforms state-of-the-art static latency prediction techniques, including matrix factorization and Vivaldi with a high dimension, on the Seattle dataset of personal devices. The proposed convex approximation to low-rank tensor completion based on 3D sampled measurements can further substantially enhance the estimation accuracy of time-varying network latencies.

The remainder of this paper is organized as follows. Sec. II reviews the related literature, followed by a comparison of latency measurements in Seattle and PlanetLab in Sec. III to motivate our studies. In Sec. IV, we propose a distance-feature decomposition procedure to extract the network feature matrices from raw RTT measurements. In Sec. V, we propose a new family of rank minimization algorithms to fully recover the network feature matrix. In Sec. VI, we extend our algorithms to the case of approximate tensor completion, which further enhances latency estimation based on historical measurements. In Sec. VII, we evaluate the performance of the proposed algorithms based on real-world datasets, in comparison with state-of-the-art algorithms. The paper is concluded in Sec. VIII.



(a) RTT distributions (b) Max RTT for each pair of nodes

Fig. 1. RTT distributions in Seattle and PlanetLab. a) CDFs of all measured RTTs. b) CDFs of the maximum RTT measured for each pair of nodes.

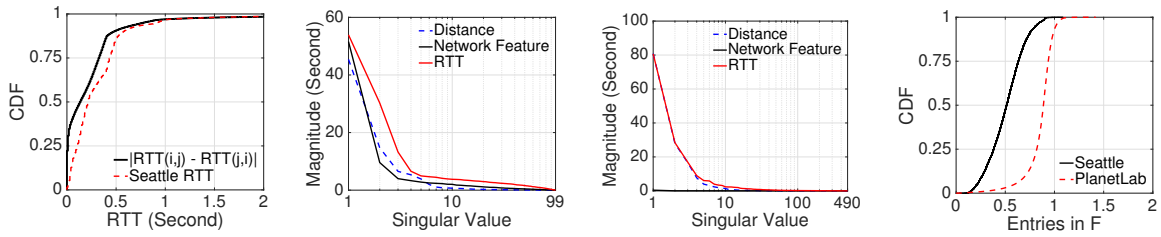
II. Relation to Prior Work

Network coordinate systems (NCSs) embed hosts into a coordinate space such as Euclidean space, and predict latencies by the coordinate distances between hosts [11]. In this way, explicit measurements are not required to predict latencies. Most of the existing NCSs, such as Vivaldi [1], GNP [2], rely on the Euclidean embedding model. However, such systems suffer from a common drawback that the predicted distances among every three hosts have to satisfy the triangle inequality, which does not always hold in practice. Many studies [12], [13] have reported the wide existence of triangle inequality violations (TIV) on the Internet.

To overcome the TIV problem, some other techniques have been proposed recently. The idea of compressive sensing is to recover a sparse vector from partial observations, and has been used to interpolate network latencies [14]. Another emerging technique is matrix completion, which aims to recover the low-rank network distance matrix from partially sampled values in the matrix. One approach to solving matrix completion problems is matrix factorization [15], which assumes the matrix to be recovered has a certain fixed rank. This approach has recently been applied to network latency estimation [3]. The estimated distances via matrix factorization do not have to satisfy the triangle inequality. However, these systems actually do not outperform Euclidean embedding models significantly, due to the reported problems such as prediction error propagation [4]. Besides, without considering the geographical distances between hosts that dictate propagation delays, they have missed a major chunk of useful information.

Another popular approach to matrix completion problems is to minimize the rank of an incomplete matrix subject to bounded deviation from known entries [16]. The advantage of this approach is that it does not assume the matrix has a known fixed rank. Some recent studies [17], [18] adopt rank minimization to recover unknown network latencies. In this paper, we also use rank minimization to recover the network latency matrix (after removing the Euclidean component). However, we propose a robust Schatten-p norm minimization algorithm which incorporates Frobenius norms on one extreme for better efficiency and nuclear norms on the other extreme for better approximation, and can thus flexibly trade complexity off for accuracy, depending on application requirements and available computational resources.

Measurement studies have been conducted for different kinds of networks, such as WiFi networks [19], Cellular networks [20], and 4G LTE networks [21], reporting latencies and



(a) $|\text{RTT}(i,j) - \text{RTT}(j,i)|$ (Seattle) (b) Singular values (Seattle) (c) Singular values (PlanetLab) (d) CDF of entries in F

Fig. 2. Properties of Seattle and PlanetLab RTT matrices, in terms of asymmetry as well as rank properties before and after feature extraction.

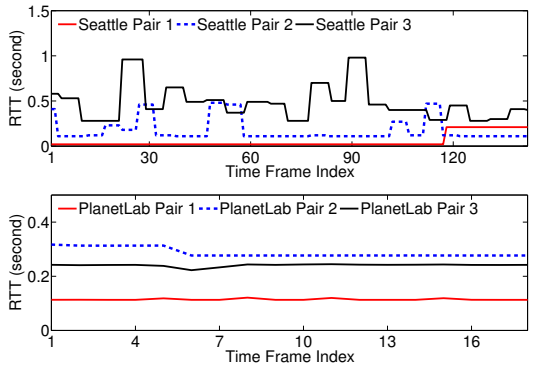


Fig. 3. The time-varying characteristics of latencies between 3 pairs of nodes in Seattle and PlanetLab.

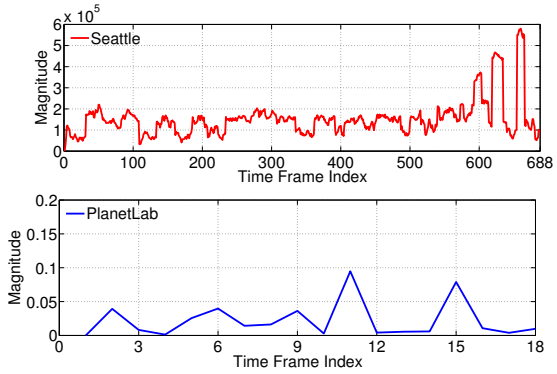


Fig. 4. The relative varying percentage (RVP) of every measured latency matrix relative to the first measured latency matrix in Seattle and PlanetLab.

other properties. The latency measurement on Seattle is cross-network in nature, as Seattle involves many different types of nodes from stable servers to personal devices including laptops and smart phones.

III. *Seattle* vs. *PlanetLab*: Measuring the Latencies

In this section, we present characteristics of latencies measured from Seattle [7] containing personal devices, in comparison to those from PlanetLab. We make all the measurements publicly available for reproducibility [10]. *Seattle* is a new open peer-to-peer computing platform that provides access to personal computers worldwide. In contrast to PlanetLab [22], which is a global research network comprised of computers mostly located in stable university networks, Seattle nodes include many personal devices, such as mobile phones, laptops, and personal computers, donated by users and institutions. Due to the diversity, mobility and instability of these personal devices, Seattle is significantly different from PlanetLab in terms of latency measurements.

We have collected the round trip times (RTTs) between 99 nodes in the Seattle network in a 3-hour period commencing at 9 pm on a day in summer 2014. The dataset has $6,743,088$ latency measurements in total, consisting of 688 latency matrices, each of which has a size of 99×99 and represents the pairwise RTTs between 99 nodes collected in a 15.7-second timeframe. In this paper, we may refer to each matrix as a "frame" since the collected data is 3D. Our data collection on Seattle was limited to 99 nodes because as a new platform that includes both personal computers and servers, Seattle is yet to receive more donations of personal devices. However, it will be clear in Sec. VII that the collected data is rich enough for the purpose of studying latency prediction algorithms.

As a benchmark dataset, we also collected the RTTs between 490 PlanetLab nodes in a 9-day period in 2013 and obtained $4,321,800$ latency measurements in total, consisting of 18 matrices (frames), each of which has a size of 490×490 and represents the pairwise RTTs collected in a 14.7-hour timeframe. We will compare the Seattle data with PlanetLab data in terms of RTT statistics, ranks of latency matrices, and time-varying characteristics.

Round Trip Times. Fig. 1(a) shows that Seattle RTTs (with a mean of 0.36 seconds) are greater than PlanetLab RTTs (with a mean of 0.15 seconds), and are spread in a wider range. While the maximum RTT observed in PlanetLab is only 7.90 seconds, the maximum RTT in Seattle is 90.50 seconds, probably because some nodes are temporarily offline, which is a common case for cellular devices not in the service region.

Asymmetry and Triangle Inequality Violation. Traditional Euclidean embedding methods for network latency prediction [1], [2] assume symmetry and triangle inequalities for pairwise latencies, which may not be true in reality, especially when an increasing number of mobile devices is present with unstable network connectivity. Fig. 2(a) shows the CDF of Seattle latencies as well as the CDF of asymmetric gap $|\text{RTT}(i,j) - \text{RTT}(j,i)|$ in Seattle. We can see that the asymmetric gaps $|\text{RTT}(i,j) - \text{RTT}(j,i)|$ have a distribution very close to that of actual latencies in Seattle, verifying that Seattle RTTs are not symmetric. This is in sharp contrast to PlanetLab, in which latencies can be assumed to be symmetric. Furthermore, to test triangle inequality violation (TIV), we randomly select $10,000,000$ triples of nodes from Seattle data and observe a TIV ratio of as high as 55.4%, while the TIV ratio in PlanetLab data is only 17.5%. Due to asymmetric RTTs and TIV, Euclidean embedding is insufficient to model pairwise latencies in Seattle.

Rank of Latency Matrices. We perform singular value decomposition (SVD) [23] on a typical latency matrix frame in Seattle and a typical frame in PlanetLab as well, and plot the singular values of both latency matrices in Fig. 2(b) and Fig. 2(c), respectively. We can observe that the singular values of both matrices decrease fast. The 15th singular value of the Seattle latency matrix is 4.9% of its largest one, while the 7th singular value of the PlanetLab latency matrix is 4.7% of its largest one. This confirms the low-rank nature of Internet latencies reported in previous measurements [24].

Time-Varying Characteristics. Fig. 3 compares the RTTs of 3 typical node pairs in the Seattle network with those in PlanetLab. Since Seattle contains many user-donated personal devices including mobile phones and laptops, its latencies may vary greatly across time, whereas the latencies in PlanetLab do not change by more than 0.1 second even across hours.

To get a better idea about the evolution of frames of data over time, we denote $M(t)$ the $n \times n$ latency matrix measured at time t, where $M_{ij}(t)$ is the RTT between node i and node j. Then, we define the Relative Varying Percentage (RVP) of $M(t)$ relative to the first matrix $M(1)$ as

$$\text{RVP}(t,1) = \frac{1}{n^2-n} \sum_{(i,j),i\neq j} [M_{ij}(t) - M_{ij}(1)]/M_{ij}(1).$$

We plot RVP$(t,1)$ for every frame t in Fig. 4 for both Seattle and PlanetLab. We can observe a huge difference between the two datasets. While the RVP of PlanetLab frames over 9 days always stays below 0.09, the RVPs of Seattle frames in merely 3 hours can be up to 5.8×10^5 with a mean 1.5×10^5. This demonstrates the time-varying nature of Seattle latencies, which makes it hard to predict the latency between two Seattle nodes. Traditional network coordinate embedding is not suitable to model the latencies in personal device networks.

IV. Static Latency Estimation via Distance-Feature Decomposition

We first present our solution to static network latency estimation involving personal devices. Combining the strengths of both Euclidean embedding and matrix completion, we model each pairwise latency in Seattle as the product of a *distance component*, representing the geographic distance that dictates propagation delay, and a *network feature component*, indicating the network connectivity between the pair of nodes. We only assume the extracted network features are correlated among nodes, while the distances satisfy Euclidean properties. In this section, we will propose a distance-feature decomposition procedure for static latency estimation.

A. Problem Definition and Our Model

Let $\mathbb{R}^n$ denote the n-dimensional Euclidean space. The set of all $m \times n$ matrices is denoted by $\mathbb{R}^{m\times n}$. Assume a network contains n nodes, and the latency matrix measured between these nodes is $M \in \mathbb{R}^{n\times n}$, where M_{ij} denotes the RTT between nodes i and j. We use Ω to denote the set of index pairs (i,j) where the measurements M_{ij} are known, and Θ to denote the set of unknown index pairs. For missing entries $(i,j) \notin \Omega$, we denote their values as $M_{ij} =$ unknown. We define the sample rate R as the percentage of known entries in M. Given an incomplete latency matrix M, the static network latency estimation problem in this paper is to recover all pairwise latencies. We denote the estimated complete latency matrix as $\hat{M} \in \mathbb{R}^{n\times n}$.

We model the RTT matrix M as the *Hadamard product* (or entry-wise product) of a *symmetric* distance matrix $D \in \mathbb{R}^{n\times n}$ and an *asymmetric* network feature matrix $F \in \mathbb{R}^{n\times n}$, i.e., $M = D \circ F$, where $M_{ij} = D_{ij}F_{ij},\ 1 \le i,\ j \le n$, D_{ij} represents the distance between nodes i and j in a Euclidean space, and F_{ij} represents the "network connectivity" from node i to node j; a smaller F_{ij} indicates a better connectivity between nodes i and j. We assume that the network feature matrix F is a *low-rank matrix contaminated by noise*. The rationale behind is as follows.

First, we assume the network feature matrix F has a low rank, because correlation exists between network connectivities on all incoming (or outgoing) links of each node, and feature vectors can clearly interpret such correlations. In particular, we call the vector $f_l^i \in \mathbb{R}^r$ as an r-dimensional *left feature vector* of node i, which represents the network feature from node i to other nodes. Similarly, we call the vector $f_r^j \in \mathbb{R}^r$ the *right feature vector* of node j, which represents the network feature from other nodes to node j. Hence, the network connectivity from node i to node j can be determined by the feature vectors, i.e., $F_{ij} = {f_l^i}^\top f_r^j$, and the whole network feature matrix F can be represented by

$$F = F_l F_r^\top, F_l \in \mathbb{R}^{n\times r}, F_r \in \mathbb{R}^{n\times r}, \tag{1}$$

where the i-th row of F_l is f_l^i and the j-th row of F_r is f_r^j.

Second, the distance matrix D defined above is *not* guaranteed to have a low rank. Note that there is another type of matrix, namely Euclidean Distance Matrix (EDM), which is defined as a matrix D' of *squared* distances $D'_{ij} := \|x_i - x_j\|^2$. The rank of D' is known to be no more than $2 + d$, where d is the dimension of the Euclidean Space [25]. However, no conclusion on rank can be made for our D, where $D_{ij} = \|x_i - x_j\|$. Therefore, we do not assume any rank properties for D.

In a nutshell, the distance matrix D models the geographical distances between nodes. D is symmetric, satisfies the triangle inequality, yet does not necessarily have a low-rank. On the other hand, the network feature matrix F models factors like network congestions and node status. F is asymmetric and may violate the triangle inequality, but is low-rank. Our model overcomes the shortness of both Euclidean embedding and low-rank matrix completion, since symmetry and triangle inequalities only need to hold for the distance matrix D but not F, and the low-rank property is only assumed for network connectivity F but not D.

B. Distance-Feature Decomposition

We propose a distance-feature decomposition procedure in Algorithm 1, where we employ a simple network feature extraction process, as described in the first 2 steps to remove the Euclidean distance component D. Specifically, we estimate the distance matrix $\hat{D}$ by performing Euclidean embedding

Algorithm 1 Distance-Feature Decomposition

1: Perform Euclidean Embedding on incomplete RTT matrix M to get a complete matrix of distance estimates $\hat{D}$
2: $F_{ij} := \begin{cases} \frac{M_{ij}}{\hat{D}_{ij}} & \forall (i,j) \in \Omega \\ \text{unknown} & \forall (i,j) \notin \Omega \end{cases}$
3: Perform matrix completion on F to get the complete matrix of network feature estimates $\hat{F}$
4: Output $\hat{M}_{ij} := \hat{D}_{ij}\hat{F}_{ij},\ 1 \le i,\ j \le n$

on raw data M, e.g., via Vivaldi [1], which can find the coordinate x_i of each node i in a Euclidean space, given partially measured pairwise RTTs. The distance between nodes i and j can then be estimated as their distance $\hat{D}_{ij} = \|x_i - x_j\|$ in the Euclidean space. We then divide M by $\hat{D}$ (element-wise), leading to an incomplete network feature matrix F. A rank minimization algorithm is then performed on F (which could be noisy) to estimate a complete network feature matrix $\hat{F}$ without having to know its rank *a priori*. Finally, the predicted latency between nodes i and j is given by $\hat{M}_{ij} := \hat{D}_{ij}\hat{F}_{ij},\ 1 \le i,\ j \le n$.

In Fig. 2(b), Fig. 2(c), and Fig. 2(d), we show the rank properties of M, $\hat{D}$, and F for a typical Seattle frame and a typical PlanetLab frame. Here we assume that data in both frames are all known for the study of rank properties only. For Seattle, we observe that the extracted network feature matrix F has the steepest descent of singular values and is likely to have a lower rank than the original RTT matrix M. It is worth noting that although $\hat{D}$ seems to have faster decreasing singular values, it is already removed by Euclidean embedding and we do not take further actions on $\hat{D}$.

In contrast, for PlanetLab, the above decomposition phenomenon is not observed. As shown in Fig. 2(c), the singular values of $\hat{D}$ almost overlap with those of the raw latency M, while the network feature matrix F has much smaller singular values relative to M, even though most entries in F are a bit larger than those for the Seattle case, as shown in Fig. 2(d). This implies that for PlanetLab, the distance matrix D produced by Euclidean embedding can already approximate raw latencies M accurately enough. Therefore, there is no need to extract the network feature matrix F (and further perform matrix completion on F) in PlanetLab. These observations will be further re-confirmed by our trace-driven evaluation results in Sec. VII.

V. Robust Matrix Completion via Schatten-p Norm Minimization

The core of our latency estimation procedure (Step 3 in Algorithm 1) is to complete the extracted feature matrix F, which is possibly noisy. Formally, given a noisy input matrix $X \in \mathbb{R}^{m \times n}$ with missing entries, the problem of low-rank matrix completion is to find a complete matrix $\hat{X}$ by solving

$$\begin{aligned} \underset{\hat{X} \in \mathbb{R}^{m \times n}}{\text{minimize}} \quad & \text{rank}(\hat{X}) \\ \text{subject to} \quad & |\hat{X}_{ij} - X_{ij}| \le \tau,\ (i,j) \in \Omega, \end{aligned} \tag{2}$$

where τ is a parameter to control the error tolerance on known entries of the input matrix X [26] or the maximum noise that is present in the observation of each known pair $(i,j) \in \Omega$. It is well-known that problem (2) is an NP-hard problem. In contrast to matrix factorization [3], the advantage of the matrix completion formulation above is that we do need to assume the rank of the network feature matrix is known *a priori*.

One popular approach to solve (2) is to use the sum of singular values of $\hat{X}$, i.e., the nuclear norm, to approximate its rank. The nuclear norm is proved to be the convex envelope of the rank [27] and can be minimized by a number of algorithms, including the well-known singular value thresholding (SVT) [9]. Other smooth approximations include Reweighted Nuclear Norm Minimization [28], and Iterative Reweighted Least Squares algorithm IRLS-p (with $0 \le p \le 1$) [8], which attempts to minimize a weighted Frobenius norm of $\hat{X}$.

A. A Family of Iterative Weighted Algorithms

Note that all the state-of-the-art rank minimization algorithms mentioned above either minimize the nuclear norm, which is a better approximation to rank, or the Frobenius norm, which is efficient to solve. In this paper, we propose a family of robust algorithms, called Iterative weighted Schatten-p norm minimization (IS-p), with $1 \le p \le 2$, which is a generalization of a number of previous "iterative reweighted" algorithms to a tunable framework; the IS-p algorithm minimizes a reweighted nuclear norm if $p = 1$ and minimizes a reweighted Frobenius norm if $p = 2$. We will show that with IS-p is more robust to any practical parameter settings and trades complexity off for accuracy depending on the application requirements.

Algorithm 2 The IS-p Algorithm ($1 \le p \le 2$)

1: **Input**: An incomplete matrix $X \in \mathbb{R}^{m \times n}$ ($m \le n$) with X_{ij} known only for $(i,j) \in \Omega$; the error tolerance τ on known entries
2: **Output**: $\hat{X}$ as an approximate solution to (2).
3: Initially, $L^0 := I$, δ^0 is an arbitrary positive number
4: **for** $k = 1$ to maxIter **do**
5: Solve the following convex optimization problem to obtain the optimal solution $\hat{X}^k$:

$$\begin{aligned} \underset{\hat{X}}{\text{minimize}} \quad & \|L^{k-1}\hat{X}\|_p^p \\ \text{subject to} \quad & |\hat{X}_{ij} - X_{ij}| \le \tau,\ (i,j) \in \Omega \end{aligned} \tag{3}$$

6: $[U^k, \Sigma^k, V^k] := \text{SVD}(\hat{X}^k)$, where Σ^k is an $m \times n$ diagonal matrix with non-negative real numbers (singular values of $\hat{X}^k$) $\sigma_1^k, \ldots, \sigma_m^k$ on the diagonal
7: Form a weight matrix $W^k \in \mathbb{R}^{m \times m}$, where

$$W_{ij}^k := \begin{cases} \left((\sigma_i^k)^p + \delta^{k-1}\right)^{-\frac{1}{p}}, & i = j \\ 0, & i \ne j \end{cases}$$

8: Choose δ^k such that $0 < \delta^k \le \delta^{k-1}$.
9: $L^k := U^k W^k {U^k}^\top$
10: **end for**
11: $\hat{X} := \hat{X}^{\text{maxIter}}$

The IS-p algorithm is described in Algorithm 2. Note that when $p = 1$, problem (3) is a nuclear-norm minimization

problem, and when $p = 2$, problem (3) becomes Frobenius-norm minimization. In fact, for $1 \leq p \leq 2$, problem (3) is a convex problem in general. To see this, for any $X \in \mathbb{R}^{m\times n}$ ($m \leq n$), denote $\sigma_i(X)$ the ith singular value of X. Then, we have $\|X\|_p^p = \sum_{i=1}^m (\sigma_i(X))^p = \text{tr}\left((X^\top X)^{\frac{p}{2}}\right)$, which is a convex function for $p \geq 1$, since $\text{tr}(X^\top X)^{\frac{p}{2}}$ is convex and non-decreasing for $p \geq 1$ [28]. A large number of efficient solutions have been proposed to solve the nuclear-norm and Frobenius-norm versions of (3) [9], [28], while for $1 \leq p \leq 2$ problem (3) is convex in general. Therefore, we resort to existing methods to solve the convex problem (3), which will not be the focus of this paper. Furthermore, exact singular value decomposition for $\hat{X}^k$ in Step 6 can be performed within polynomial time with a complexity of $O(m^2n)$.

Let us now provide some mathematical intuition to explain why Algorithm 2 can approximate the rank minimization. Initially, we replace the objective function $\text{rank}(\hat{X})$ with $\|\hat{X}\|_p$. Subsequently, in each iteration k, we are minimizing $\|L^{k-1}\hat{X}\|_p^p$. Recall that in Step 6 of iteration k, the optimal solution $\hat{X}^k$ can be factorized as $\hat{X}^k = U^k\Sigma^k V^k$ via singular value decomposition, where $U^k \in \mathbb{R}^{m\times m}$ and $V^k \in \mathbb{R}^{n\times n}$ are unitary square matrices, i.e., ${U^k}^\top U^k = I$, ${V^k}^\top V^k = I$. Thus, we have $\|L^{k-1}\hat{X}^k\|_p^p = \|U^{k-1}W^{k-1}{U^{k-1}}^\top U^k\Sigma^k V^k\|_p^p$. If $U^{k-1} \approx U^k$ after a number of iterations, we will have

$$\begin{aligned}
\|L^{k-1}\hat{X}^k\|_p^p &\approx \|U^{k-1}W^{k-1}{U^k}^\top U^k\Sigma^k V^k\|_p^p \\
&= \|U^{k-1}(W^{k-1}\Sigma^k)V^k\|_p^p \\
&= \sum_{i=1}^m \left(\sigma_i\left(W^{k-1}\Sigma^k\right)\right)^p \\
&= \sum_{i=1}^m \left(\frac{\sigma_i^k}{((\sigma_i^{k-1})^p + \delta^{k-1})^{1/p}}\right)^p \\
&= \sum_{i=1}^m \frac{(\sigma_i^k)^p}{(\sigma_i^{k-1})^p + \delta^{k-1}},
\end{aligned} \tag{4}$$

which eventually approaches $\text{rank}(\hat{X}^k)$. To see this, note that for two sufficiently small positive constants δ^{k-1} and δ^k, upon convergence, i.e., when $\sigma_i^k = \sigma_i^{k-1}$, we have

$$\frac{(\sigma_i^k)^p}{(\sigma_i^{k-1})^p + \delta^{k-1}} \approx \frac{(\sigma_i^k)^p}{(\sigma_i^k)^p + \delta^k} \approx \begin{cases} 0 & \text{if } \sigma_i^k = 0, \\ 1 & \text{if } \sigma_i^k > 0, \end{cases}$$

Therefore, $\|L^{k-1}\hat{X}^k\|_p^p$ represents the number of nonzero singular values σ_i^k in $\hat{X}^k$, which is exactly the rank of $\hat{X}^k$.

B. Convergence Analysis

The above informal analysis only provides an intuitive explanation as to why the algorithm works, based on the hope that the algorithm will converge. The following theorem can ensure the convergence of the produced $\text{rank}(\hat{X}_k)$ and therefore guarantee the convergence of Algorithm 2.

Theorem 1. *Suppose $\hat{X}^k$ is the output of Algorithm 2 in iteration k. For any matrix $X \in \mathbb{R}^{m\times n}$ and any $p \in [1, 2]$, rank$(\hat{X}^k)$ converges. In particular, for a sufficiently large k, we have $\sigma_i(\hat{X}^k) - \sigma_i(\hat{X}^{k-1}) \to 0$, for $i = 1, \ldots, m$.*

Proof. We first present some useful lemmas.

Lemma 1. *For any $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times r}$, the following holds for all $1 \leq p \leq 2$:*

$$\sum_{i=1}^n \sigma_{n-i+1}^p(A)\sigma_i^p(B) \leq \|AB\|_p^p \leq \sum_{i=1}^n \sigma_i^p(A)\sigma_i^p(B), \tag{5}$$

where $\sigma_i(A)$ denotes the ith singular value of A.

Please refer to the appendix for a proof of this lemma.

Corollary 2. *Given an $m \times m$ diagonal matrix A with nonnegative and non-decreasing (non-increasing) diagonal entries $a_{11}, \ldots, a_{mm}$, and another $m \times n$ diagonal matrix B with nonnegative and non-increasing (non-decreasing) diagonal entries $b_{11}, \ldots, b_{mm}$, we have $\|AUB\|_p \geq \|AB\|_p$ for any $m \times m$ square unitary matrix U (i.e., $U^U = I$), where $1 \leq p \leq 2$.*

Proof of Corollary 2. Without loss of generality, we assume $a_{11} \geq a_{22} \geq \ldots \geq a_{mm} \geq 0$ and $0 \leq b_{11} \leq b_{22} \leq \ldots \leq b_{mm}$. By Lemma 1, we have

$$\begin{aligned}
\|AUB\|_p^p &\geq \sum_{i=1}^m \sigma_i^p(A)\sigma_{n-i+1}^p(UBI) = \sum_{i=1}^m \sigma_i^p(A)\sigma_{n-i+1}^p(B) \\
&= \sum_{i=1}^m a_{ii}^p b_{ii}^p = \sum_{i=1}^m \sigma_i^p(AB) = \|AB\|_p^p,
\end{aligned}$$

proving the corollary. □

We now prove Theorem 1. According to Corollary 2 and the *unitarily invariant property* of Schatten-p norms, we have

$$\begin{aligned}
\|L^{k-1}\hat{X}^k\|_p &= \|U^{k-1}W^{k-1}{U^{k-1}}^\top U^k\Sigma^k{V^k}^\top\|_p && (6)\\
&= \|W^{k-1}{U^{k-1}}^\top U^k\Sigma^k\|_p && (7)\\
&\geq \|W^{k-1}\Sigma^k\|_p && (8)\\
&= \left(\sum_{i=1}^n \frac{\left(\sigma_i^k\right)^p}{\left(\sigma_i^{k-1}\right)^p + \delta^{k-1}}\right)^{\frac{1}{p}}, && (9)
\end{aligned}$$

where (8) is due to Lemma 2, since W^{k-1} and Σ^k are diagonal matrices with nonnegative non-decreasing and non-increasing entries, respectively, and ${U^{k-1}}^\top U^k$ is still unitary.

Since $\hat{X}^k$ is the optimal solution to (3), we have

$$\begin{aligned}
\|L^{k-1}\hat{X}^k\|_p &\leq \|L^{k-1}\hat{X}^{k-1}\|_p && (10)\\
&= \|U^{k-1}W^{k-1}\Sigma^{k-1}{V^{k-1}}^\top\|_p && (11)\\
&= \|W^{k-1}\Sigma^{k-1}\|_p && (12)\\
&= \left(\sum_{i=1}^n \frac{\left(\sigma_i^{k-1}\right)^p}{\left(\sigma_i^{k-1}\right)^p + \delta^{k-1}}\right)^{\frac{1}{p}} && (13)
\end{aligned}$$

Since $\delta^k \leq \delta^{k-1}$, we have

$$\begin{aligned}
\sum_{i=1}^n \frac{\left(\sigma_i^k\right)^p}{\left(\sigma_i^{k-1}\right)^p + \delta^{k-1}} &\leq \sum_{i=1}^n \frac{\left(\sigma_i^{k-1}\right)^p}{\left(\sigma_i^{k-1}\right)^p + \delta^{k-1}}, \\
\sum_{i=1}^n \frac{\left(\sigma_i^k\right)^p + \delta^k}{\left(\sigma_i^{k-1}\right)^p + \delta^{k-1}} &\leq \sum_{i=1}^n \frac{\left(\sigma_i^{k-1}\right)^p + \delta^{k-1}}{\left(\sigma_i^{k-1}\right)^p + \delta^{k-1}} = n.
\end{aligned}$$

Let $x_i^k := (\sigma_i^k)^p$ and $x^k = (x_1^k, x_2^k, ..., x_n^k)$. Define a function $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}_+$, $\mathcal{L}(x) = \prod_{i=1}^n (x_i + \delta^k)$, with $\delta^k > 0$.

We will show that the sequence $\mathcal{L}(x^k)$ is monotonically non-increasing using a similar method in [29], and prove the convergence of σ_i^k for $1 \le i \le n$.

Using the inequality between the arithmetic and geometric means for nonnegative terms, we have

$$\prod_{i=1}^{n} \frac{x_i^k + \delta^k}{x_i^{k-1} + \delta^{k-1}} \le 1,$$

which implies that $\mathcal{L}(x^k) \le \mathcal{L}(x^{k-1})$. Also, since $x_i^k \ge 0$, $\mathcal{L}$ is bounded below by δ^n, the sequence $\mathcal{L}(x^k)$ converges. It implies that

$$\prod_{i=1}^{n} \frac{x_i^k + \delta^k}{x_i^{k-1} + \delta^{k-1}} = \frac{\mathcal{L}(x^k)}{\mathcal{L}(x^{k-1})} \to 1.$$

Define y^k to be $y_i^k = \frac{x_i^k + \delta^k}{x_i^{k-1} + \delta^{k-1}}$, and $y_1^k = 1 + \epsilon$. We have

$$\prod_{i=1}^{n} y_i^k = (1+\epsilon) \prod_{i=2}^{n} y_i^k \le (1+\epsilon) \left(1 - \frac{\epsilon}{n-1}\right)^{n-1} = f(\epsilon)$$

by combining $\sum_{i=1}^{n} y_i^k \le n$ and the inequality between the arithmetic and geometric means. Function $f(\epsilon)$ is continuous and satisfies $f(0) = 1$, $f'(0) = 0$, and $f''(\epsilon) < 0$, for $|\epsilon| < 1$. Hence, $f(\epsilon) < 1$ for $\epsilon \ne 0$, $|\epsilon| < 1$.

Therefore, since $\prod_{i=1}^{n} y_i^k \to 1$, we have $f(\epsilon) \to 1$, which in turn implies $\epsilon \to 0$. Hence $y_1^k \to 1$, and the same holds for all y_i^k. Thus, we have

$$y_i^k = \frac{(\sigma_i^k)^p + \delta^k}{(\sigma_i^{k-1})^p + \delta^{k-1}} \to 1.$$

By monotone convergence theorem, there exists a point $\delta^* \ge 0$ such that $\delta^k \to \delta^*$, and thus $\delta^{k-1} - \delta^k \le \delta^{k-1} - \delta^* \to 0$, implying $\delta^{k-1} - \delta^k \to 0$. Since σ_i^k is finite, we conclude that $\sigma_i^k - \sigma_i^{k-1} \to 0$, for all $i = 1, \ldots, n$, which implies $\text{rank}(\hat{X}^k) - \text{rank}(\hat{X}^{k-1}) \to 0$. □

C. Relationships to Prior Algorithms

We now point out that the proposed IS-p algorithm is a generalization of a number of previous reweighted approximate algorithms based on either nuclear norm or Frobenius norm alone to a tunable class of algorithms trading complexity off for performance.

Singular value thresholding (SVT) is an algorithm to solve the convex nuclear norm minimization:

$$\begin{aligned} \underset{\hat{X} \in \mathbb{R}^{m \times n}}{\text{minimize}} \quad & \|\hat{X}\|_* \\ \text{subject to} \quad & |\hat{X}_{ij} - X_{ij}| \le \tau, \ (i,j) \in \Omega, \end{aligned} \tag{14}$$

which approximates (2). It is shown [30] that for most matrices of rank r, (14) yields the same solution as (2), provided that the number of known entries $m \ge Cn^{6/5} r \log n$ for some positive constant C. However, when $m < Cn^{6/5} r \log n$, the nuclear-norm-minimizing solution from SVT usually cannot approximate (2) well. In fact, SVT can be viewed as only performing the first iteration of the proposed Algorithm 2 with $p = 1$. In contrast, Algorithm 2 adopts multiple iterations of reweighted minimizations to refine the results and can further approximate the rank minimization problem over iterations, even if $m < Cn^{6/5} r \log n$.

A number of iterative reweighted approximations to (2) have been proposed. They could be different in performance, mainly due to the different norms (either Frobenius norm or nuclear norm) adopted as well as the way to form the weight matrix L^k. Iterative Reweighted Least Squares (IRLS-p and sIRLS-p) [28] is also a reweighted algorithm to approximate the affine rank minimization problem (i.e., problem (2) with $\tau = 0$ in the constraint). It minimizes a weighted Frobenius norm $\|L^{k-1}X\|_F$ in each iteration k to produce an X^k, where $L^{k-1} := \sqrt{(X^{k-1\top} X^{k-1} + \delta I)^{p/2-1}}$ with $0 \le p \le 1$. By simple maths derivations, we find the weight L^{k-1} in IRLS-p is different from that in Algorithm 2, therefore yielding different approximation results. Furthermore, IRLS-p can only minimize a Frobenius norm in each iteration, whereas the nuclear norm is known to be the best convex approximation of the rank function [27]. In contrast, the proposed Algorithm 2 represents a family of algorithms including nuclear norm minimization (when $p = 1$) on one end to achieve better approximation and Frobenius norm minimization (when $p = 2$) on the other end for faster computation.

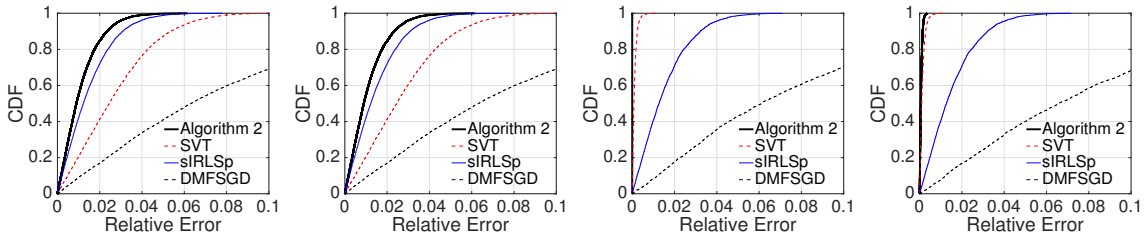
D. Performance on Synthesized Low-Rank Data

We evaluate our algorithm based on synthesized true low-rank matrices contaminated by random noises, in comparison with several state-of-the-art approaches to matrix completion:

- **Singular Value Thresholding (SVT)** [9]: an algorithm for nuclear norm minimization as an approximation to rank minimization;
- **Iterative Reweighted Least Squares (sIRLS-p)** [28]: an iterative algorithm to approximate rank minimization with a reweighted Frobenius-norm minimization in each iteration. According to [28], the performance of sIRLS-1 is proved to guarantee the recovery performance, thus we choose sIRLS-1 for comparison;
- **DMFSGD Matrix Factorization** [3]: a distributed network latency prediction algorithm that attempts to approximate a given matrix M using the product of two smaller matrices $\hat{M} = UV^\top$, where $U \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{n \times r}$, such that a loss function based on $M - \hat{M}$ is minimized, where r is the assumed rank of $\hat{M}$.

In our experiments, we randomly generate 100×100 matrices with rank $r = 20$, contaminated by noise. The generated matrix can be represented as $X = UV^\top + \epsilon N$, where U and V are randomly generated $n \times r$ matrices ($n = 100,\ r = 20$) with entries uniformly distributed between 0 and 1. N is an $n \times n$ standard Gaussian noise matrix. We run simulations under the sample rates $R = 0.3$ and $R = 0.7$ and under both the noiseless case $\epsilon = 0$ and the noisy case $\epsilon = 0.1$ to test the algorithm robustness.

Fig. 5(a) and Fig. 5(c) compare the performance of different algorithms in the noiseless case. As we can see, our algorithm is the best at low sample rate ($R = 0.3$). When the sample rate is high ($R = 0.7$), both our algorithm and SVT are the best. For the noisy case, Fig. 5(b) and Fig. 5(d) show that our algorithm outperforms all other algorithms at both the low



(a) $R = 0.3$, ϵ=0 (b) $R = 0.3$, ϵ=0.1 (c) $R = 0.7$, ϵ=0 (d) $R = 0.7$, ϵ=0.1

Fig. 5. Performance of IS-p ($p = 1$) and other algorithms on synthetic 100×100 matrices with rank $r = 20$, under sample rates $R = 0.3$ and $R = 0.7$.

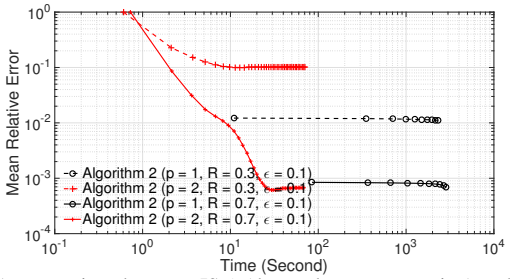


Fig. 6. A comparison between IS-1 (the nuclear-norm version) and IS-2 (the Frobenius-norm version) in terms of recovery errors and running time.

sample rate ($R = 0.3$) and the high sample rate ($R = 0.7$), thus proving that our algorithm is the most robust to noise.

Under the same setting, we now investigate the tradeoff between setting $p = 1$ (the Nuclear norm version) and $p = 2$ (the Frobenius norm version) in IS-p in Fig. 6. In general, the nuclear norm version (IS-1) usually converges in a few iterations (usually one iteration) and more iterations will give little improvement. On the other hand, the Frobenius norm version (IS-2) requires more iterations to converge, and the relative recovery error decreases significantly as more iterations are adopted.

Specifically, under $R = 0.3$, IS-1 already achieves a low error within about 10 seconds. In this case, although IS-2 leads to a higher error, yet it enables a tunable tradeoff between accuracy and running time. When $R = 0.7$, IS-2 is better considering both the running time and accuracy. Therefore, we make the following conclusions:

First, IS-1 (the nuclear norm version) achieves better accuracy in general, yet at the cost of a higher complexity. IS-1 could be slower when more training data is available. The reason is that when problem (3) for $p = 1$ in Algorithm 2 is solved by a semidefinite program (SDP) (with performance guarantees [27]), which could be slow when data size increases. Note that SVT or other first-order algorithms cannot be applied to (3) due to the weight matrix L in the objective. Therefore, IS-1 should only be used upon abundant computational power or high requirement on accuracy.

Second, IS-2 has a low per-iteration cost, i.e., the error decreases gradually when more iterations are used. Therefore, it allows the system operator to flexibly tune the achieved accuracy by controlling the running time invested. Furthermore, although IS-2 does not always lead to the best performance, the achieved relative error is usually sufficient for the purpose completing the network feature matrix F. Due to this flexible nature of IS-2, we set $p = 2$ for our experiments on network latency estimation in Sec. VII, so that we can control the rank of the recovered network feature matrix $\hat{F}$ that we want to achieve, under a given budget of running time.

In our experiments, we actually set $\delta^k = \delta^{k-1}/\eta$, where $\eta > 1$ is a constant. We find that good performance is usually achieved by a large initial value of δ and an appropriate η. Specifically, we set the initial δ to 100,000 and $\eta = 2$.

VI. Dynamic Latency Estimation via Tensor Approximation

Most existing network latency prediction techniques [1], [3], [15] attempt to predict static median/mean network latencies between stable nodes such as PlanetLab nodes. However, for personal devices with mobility and time-varying network conditions, as has been illustrated in Fig. 3 and Fig. 4, static network latency estimation based on only the current frame is not effective enough to capture the changing latencies.

The above fact motivates us to study the *dynamic latency prediction* problem, that is to predict the missing network latencies in the current frame based on both the current and previous frames, i.e., based on a sliding window of latency frames sampled from random node pairs at different times up to the present. Note that although latencies in Seattle change frequently, they may linger in a state for a while before hopping to a new state, as shown in Fig. 3. Therefore, we can improve the prediction accuracy for the current frame, if we utilize the autocorrelation between frames at different times in addition to the inter-node correlation in the network feature matrix.

A. Feature Extraction from A Tensor

We use a *tensor* $\mathcal{M} = (M_{ijt}) \in \mathbb{R}^{n \times n \times T}$ to represent a 3-dimensional array that consists of T RTT matrices with missing values, each called a "frame" of size $n \times n$, measured at T different time periods.

Let $\mathbf{\Omega}$ denote the set of indices (i, j, t) where the measurements M_{ijt} are known and $\mathbf{\Theta}$ denote the set of unknown indices. The problem is to recover missing values in $\mathcal{M}$, especially the missing entries in the current timeframe with $t = 1$. Similar to the static case, we model $\mathcal{M}$ as a Hadamard product (entry-wise product) of a distance tensor $\mathcal{D} \in \mathbb{R}^{n \times n \times T}$ and a network feature tensor $\mathcal{F} \in \mathbb{R}^{n \times n \times T}$, i.e., $\mathcal{M} = \mathcal{D} \circ \mathcal{F}$, where $M_{ijt} = D_{ijt}F_{ijt}$, $1 \leq i,\ j \leq n, t = 1, \ldots, T$, with D_{ijt} representing the distance between nodes i and j in a Euclidean space at time t, and F_{ijt} representing the "network connectivity" from node i to node j at time t. Similarly, we assume the network feature tensor $\mathcal{F}$ is a *low-rank* tensor contaminated by noise.

To extract the network feature tensor $\mathcal{F}$ from 3D sampled data $\mathcal{M}$, we can apply Euclidean embedding to each frame of $\mathcal{M}$, using Vivaldi, to obtain $\hat{D}_{ijt}$ as described in Algorithm 3. Note that, in practice, it is sufficient to perform Euclidean embedding offline beforehand for the mean of several frames, assuming the distance component $\hat{D}_{ijt} \equiv \hat{D}_{ij}$ does not vary across t, such that the time-varying component is captured by the network feature tensor $\mathcal{F}$. The remaining task is to complete $\mathcal{F}$ to obtain $\hat{\mathcal{F}}$. Then, the missing latencies can be estimated as $\hat{M}_{ijt} := \hat{D}_{ijt}\hat{F}_{ijt}$.

Algorithm 3 Tensor Completion with Feature Extraction

1: Perform Euclidean Embedding on each frame of $\mathcal{M}$ to get a complete tensor of distance estimates $\hat{\mathcal{D}}$

2: $F_{ijt} := \begin{cases} \frac{M_{ijt}}{\hat{D}_{ijt}} & \forall (i,j,t) \in \mathbf{\Omega} \\ \text{unknown} & \forall (i,j,t) \notin \mathbf{\Omega} \end{cases}$

3: Perform approximate tensor completion on $\mathcal{F}$ to get the complete matrix of network feature estimates $\hat{\mathcal{F}}$

4: Output $\hat{M}_{ijt} := \hat{D}_{ijt}\hat{F}_{ijt},\ 1 \le i,\ j \le n,\ 1 \le t \le T$

B. Approximate Tensor Completion

In order to complete all missing values in $\mathcal{F}$, we generalize the matrix completion problem to tensor completion and extend our IS-p algorithm to the tensor case. In this paper, we only focus on tensors in $\mathbb{R}^{n\times n\times T}$, a size relevant to our specific latency estimation problem, although our idea can be applied to general tensors. Given a tensor $\mathcal{X} \in \mathbb{R}^{n\times n\times T}$ with missing entries, tensor completion aims to find a complete low-rank tensor $\hat{\mathcal{X}}$ by solving

$$\begin{aligned} \underset{\hat{\mathcal{X}}\in\mathbb{R}^{n\times n\times T}}{\text{minimize}} \quad & \text{rank}(\hat{\mathcal{X}}) \\ \text{subject to} \quad & |\hat{\mathcal{X}}_{ijt} - \mathcal{X}_{ijt}| \le \tau,\ (i,\ j,\ t) \in \mathbf{\Omega}, \end{aligned} \tag{15}$$

where τ is a parameter to control the error tolerance on known entries. However, unlike the case of matrices, the problem of finding a low rank approximation to a tensor is ill-posed. More specifically, it has been shown that the space of rank-r tensors is non-compact [31] and that the nonexistence of low-rank approximations occurs for many different ranks and orders. In fact, even computing the rank of a general tensor (with a dimension$\geq$ 3) is an NP hard problem [32] and there is no known explicit expression for the convex envelope of the tensor rank.

A natural alternative is to minimize a weighted sum of the ranks of some 2D matrices "unfolded" from the 3D tensor, hence reducing tensor completion to matrix completion. The unfold operation is illustrated in Fig. 7 for a 3D tensor along each of the three dimensions. Here I_1, I_2 and I_3 are index sets for each dimension. These unfolded matrices can be computed as follows:

- The column vectors of $\mathcal{X}$ are column vectors of $X_{(1)} \in \mathbb{R}^{n\times nT}$.
- The row vectors of $\mathcal{X}$ are column vectors of $X_{(2)} \in \mathbb{R}^{n\times nT}$.
- The (depth) vectors on the third dimension of $\mathcal{X}$ are column vectors of $X_{(3)} \in \mathbb{R}^{T\times n^2}$.

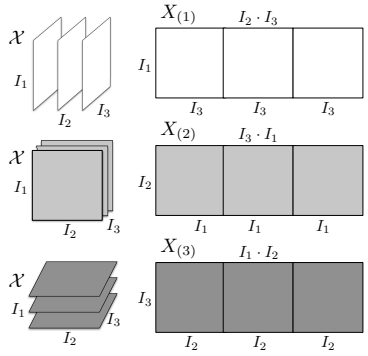


Fig. 7. Illustration of tensor unfolding for the 3D case.

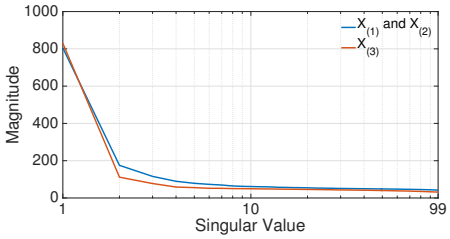


Fig. 8. The singular values of three unfolded matrices from Seattle data. The sizes of them are 99×68112, 99×68112 and 688×9801, respectively. A thresholding of 0.9 (the 95-percentile of latencies) is applied to exclude the impact of outliers.

Fig. 8 shows the singular values of all three unfolded matrices generated from the tensor of 688 frames in Seattle data. In particular, for matrix $X_{(3)}$, each row is a vector consisting of all the $99 \times 99 = 9801$ pairwise latencies. Even though $X_{(3)}$ has a large size of 688×9801, its singular values drop extremely fast: the 5th singular value of $X_{(3)}$ is only 6% of its largest singular value. This implies that latencies measured at consecutive time frames for a same pair of nodes are highly autocorrelated along time and that $X_{(3)}$ can be deemed as a low-rank matrix contaminated by noise.

With unfolding operations defined above, the problem of "low-rank" tensor approximation can be formulated as minimizing the weighted sum of ranks for all three unfolded matrices [33]:

$$\begin{aligned} \underset{\hat{\mathcal{X}}\in\mathbb{R}^{n\times n\times T}}{\text{minimize}} \quad & \sum_{l=1}^{3} \alpha_l \cdot \text{rank}(\hat{X}_{(l)}) \\ \text{subject to} \quad & |\hat{\mathcal{X}}_{ijt} - \mathcal{X}_{ijt}| \le \tau,\ (i,\ j,\ t) \in \mathbf{\Omega}, \end{aligned} \tag{16}$$

where α_l is a convex combination coefficient, with $\alpha_l \geq 0$ and $\sum_{l=1}^{3} \alpha_l = 1$.

Apparently, the above non-convex problem of minimizing the weighted sum of ranks is still hard to solve. We propose a generalization of the proposed IS-p algorithm to the tensor case. Our "low-rank" tensor approximation algorithm is described in Algorithm 4. The algorithm first solves a convex optimization problem by minimizing the sum of weighted Schatten-p norms of all unfolded matrices within the given noise tolerance. Here the weight matrices $L_{(l)}$ are assigned for each unfolded matrix of tensor $\mathcal{X}$. Then the algorithm will update weight matrices $L_{(l)}$ one by one. This procedure is similar to what we did in 2D matrix completion.

It is not hard to check that problem (17) is a convex problem for all $1 \le p \le 2$, since for a fixed weight matrix L, $||LX||_p^p$ is

Algorithm 4 IS-p Algorithm for Tensor Completion

Initialize $L_{(l)}^0 := I,\ p,\ \delta_{(l)}^0,\ \tau_{(l)},\ \eta_{(l)},\ l = 1, 2, 3$
for $k = 1$ to maxIter **do**
  Solve the following convex optimization problem to obtain the optimal solution $\hat{\mathcal{X}}^k$:

$$\begin{aligned} \underset{\hat{\mathcal{X}}}{\text{minimize}} \quad & \sum_{l=1}^{3} \alpha_l \|L_{(l)}^{k-1} \hat{X}_{(l)}\|_p^p \\ \text{subject to} \quad & |\hat{\mathcal{X}}_{ijt} - \mathcal{X}_{ijt}| \leq \tau,\ (i,j,t) \in \mathbf{\Omega} \end{aligned} \tag{17}$$

  for $l = 1$ to 3 **do**
    $[U_{(l)}^k, \Sigma_{(l)}^k, V_{(l)}^k] := SVD\left(\hat{X}_{(l)}^k\right)$, where $\Sigma_{(l)}^k$ is a diagonal matrix with diagonal elements of $\{\sigma_{(l),i}^k\}$.
    $W_{(l),ij}^k := \begin{cases} \left(\left(\sigma_{(l),i}^k\right)^p + \delta_{(l)}^{k-1}\right)^{-\frac{1}{p}}, & i = j \\ 0, & i \neq j \end{cases}$
    $L_{(l)}^k := U_{(l)}^k W_{(l)}^k {U_{(l)}^k}^\top$
    Choose $\delta_{(l)}^k$ such that $0 < \delta_{(l)}^k \leq \delta_{(l)}^{k-1}$.
  end for
end for
$\hat{X} := \hat{X}^{\text{maxIter}}$

a convex function of X. In (17), we can see that the objective function is a convex combination of three convex functions. Note that the convergence of Algorithm 4 cannot be extended directly from the matrix case, but we observe in simulation that our algorithm has robust convergence performance.

VII. Performance Evaluation

We evaluate our proposed network latency estimation approaches on both single frames of $2D$ RTT matrices and 3D multi-frame RTT measurements, in comparison with a number of state-of-the-art latency estimation algorithms. For network latency prediction based on 2D data, we evaluate our algorithm on the Seattle dataset and PlanetLab dataset; for dynamic network latency prediction based on 3D data, our algorithm is evaluated based on the Seattle dataset. We have made both datasets publicly available [10] for reproducibility.

A. Single-Frame Matrix Completion

We define the relative estimation error (RE) on missing entries as $|\hat{M}_{ij} - M_{ij}|/M_{ij}$, for $(i,j) \in \Omega$, which will be used to evaluate prediction accuracy. We compare our algorithm with the following approaches:

- **Vivaldi** with dimension $d = 3$, $d = 7$, and $d = 3$ plus a height parameter;
- **DMFSGD Matrix Factorization** as mentioned in Sec. VII, is a matrix factorization approach for RTT prediction under an assumed rank, and
- **PD with feature extraction** as our earlier work [18], which uses Penalty Decomposition for matrix completion with feature extraction as shown in Alg. 1.

For our method, the Euclidean embedding part in feature extraction is done using Vivaldi with a low dimension of $d = 3$ without the height.

We randomly choose 50 frames from the 688 frames in the Seattle data. For PlanetLab data, as differences among the 18 frames are small, we randomly choose one frame to test the methods. Recall that the sample rate R is defined as the percentage of known entries. Each chosen frame is independently sampled at a low rate $R = 0.3$ (70% latencies are missing) and at a high rate $R = 0.7$, respectively.

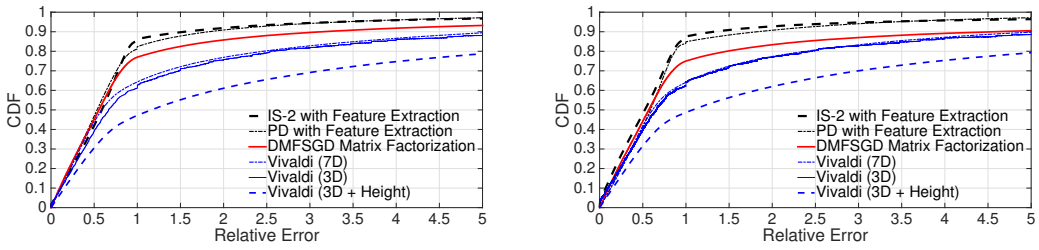
For DMFSGD, we set the rank of the estimation matrix $\hat{M}$ to $r = 20$ for Seattle data and $r = 10$ for PlanetLab data, respectively, since the 20^{th} (or 10^{th}) singular value of M is less than 5% of the largest singular value in Seattle (or PlanetLab). In fact, $r = 10$ is adopted by the original DMFSGD work [3] based on PlanetLab data. We have tried other ranks between 10-30 and observed similar performance. We plot the relative estimation errors on missing latencies in Fig. 9 for the Seattle data and Fig. 10 for the PlanetLab data. They are both under 5 methods.

For the Seattle results in Fig. 9(a) and Fig. 9(b), we can see that the IS-2 algorithm with feature extraction outperform all other methods by a substantial margin. We first check the Vivaldi algorithms. Even if Vivaldi Euclidean embedding is performed in a 7D space, it only improves over 3D space slightly, due to the fundamental limitation of Euclidean assumption. Furthermore, the 3D Vivaldi with a height parameter, which models the "last-mile latency" to the Internet core [1], is even worse than the 3D Vivaldi without heights in Seattle. This implies that latencies between personal devices are better modeled by their pairwise core distances multiplied by the network conditions, rather than by pairwise core distances plus a "last-mile latency".

The DMFSGD algorithm is also inferior to our algorithm both because it solely relies on the low-rank assumption, which may not be enough to model the Seattle latency matrices accurately, and because the proposed IS-p algorithm has better performance than DMFSGD in terms matrix completion.

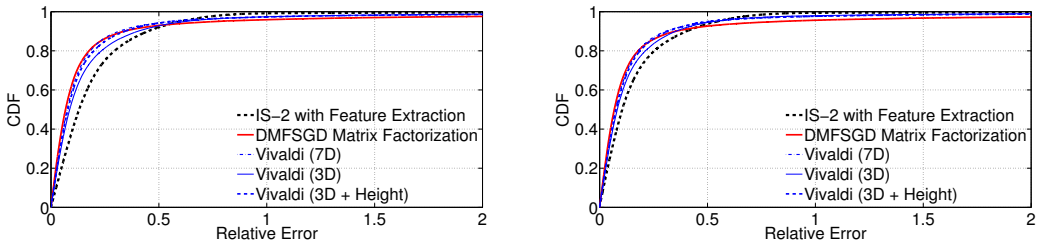
Fig. 9 also shows that the proposed IS-2 with feature extraction is even better than our work [18] that adopts the Penalty Decomposition (PD) heuristic for matrix completion after feature extraction, the latter showing the second best performance among all methods on Seattle data. This justifies our adoption of IS-2 as a high-performance algorithm for the matrix completion part, especially for highly unpredictable Seattle latencies.

In contrast, for the PlanetLab results shown in Fig. 10(a) and Fig. 10(b), our algorithm does not have a clear benefit over other state-of-the-art algorithms. As shown in our measurement in Sec. III, the latencies in PlanetLab are symmetric and only a small portion of them violate the triangle inequality. Thus, network coordinate systems such as Vivaldi already have excellent performance. Furthermore, in Fig. 2(c), we can also see that the RTT matrix M and the distance matrix $\hat{D}$ have similar singular values. Hence, there is no need to extract the network feature matrix F for PlanetLab. In this case, performing a distance-feature decomposition could introduce additional errors and is not necessary. These observations again show the unique advantage of our approach to personal device networks, although it could be an overkill for stable PlanetLab nodes.



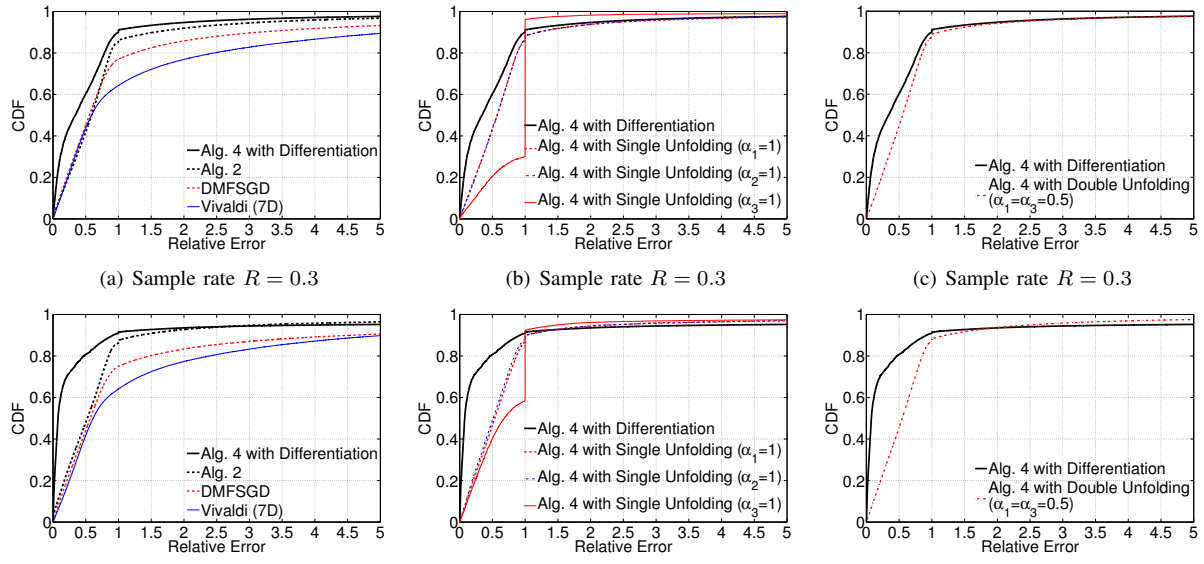
(a) Seattle (Sample rate $R = 0.3$) (b) Seattle (Sample rate $R = 0.7$)

Fig. 9. The CDFs of relative estimation errors on missing values for the Seattle dataset, under sample rates $R = 0.3$ and $R = 0.7$, respectively.



(a) PlanetLab (Sample rate $R = 0.3$) (b) PlanetLab (Sample rate $R = 0.7$)

Fig. 10. The CDFs of relative estimation errors on missing values for the PlanetLab dataset, under sample rates $R = 0.3$ and $R = 0.7$, respectively.



(a) Sample rate $R = 0.3$ (b) Sample rate $R = 0.3$ (c) Sample rate $R = 0.3$

(d) Sample rate $R = 0.7$ (e) Sample rate $R = 0.7$ (f) Sample rate $R = 0.7$

Fig. 11. The CDFs of relative estimation errors on the missing values in the *current* frame with sample rates $R = 0.3$ and $R = 0.7$ for the Seattle dataset. Feature extraction has been applied in all experiments.

B. Multi-Frame Tensor Approximation

We test our multi-frame latency tensor completion approach on 50 groups of consecutive frames in Seattle. Each group contains $T = 3$ consecutive frames of incomplete RTT measurements, forming an incomplete tensor, and such triple-frame groups are randomly selected from the Seattle dataset. The objective is to recover all the missing values in each selected tensor.

Recall that tensor completion is applied on the network feature tensor $\mathcal{F}$, whose unfolding matrices are $F_{(l)}$ for $l = 1, 2, 3$. Since our tensor has a size of $\mathbb{R}^{n\times n\times T}$, the first two unfolded matrices $F_{(1)}$ and $F_{(2)}$ have the same size $n \times nT$. Since $T = 3$ in our experiment, the size of the other unfolded matrix $F_{(3)}$ is $3 \times n^2$. As the convex combination coefficient $\alpha_1, \alpha_2, \alpha_3$ assigned to the three unfolded matrices may affect the performance of data recovery, in our evaluation, we consider the following versions of Algorithm 4:

- **Algorithm 4 with single unfolding:** only one unfolded matrix is assigned a positive weight 1 while the other two ones are assigned weight 0.
- **Algorithm 4 with double unfolding:** two of the unfolded matrices are assigned with equal weight 0.5;
- **Algorithm 4 with differentiation:** Divide the index set of all missing entries $\mathbf{\Theta}$ into two subsets:

$$\mathbf{\Theta}_A = \{(i,j)|\mathcal{M}_{ijt} \text{ is known for at least one } t \in \{1,\dots,T-1\}\},$$
$$\mathbf{\Theta}_B = \{(i,j)|\mathcal{M}_{ijt} \text{ is missing for all } t \in \{1,\dots,T-1\}\}.$$

To recover the missing entries in $\mathbf{\Theta}_A$, apply Algorithm 4 with weights $\alpha_1 = \alpha_2 = 0, \alpha_3 = 1$. To recover the

missing entries in $\mathbf{\Theta}_B$, apply Algorithm 4 with weights $\alpha_1 = 1, \alpha_2 = \alpha_3 = 0$.

We compare the above versions of Algorithm 4 with static prediction methods based on single frames, including Algorithm 2, DMFSGD and Vivaldi (7D). All versions of Algorithm 4 and Algorithm 2 are applied with feature extraction.

First, in Fig. 11(a) and Fig. 11(d), we compare Algorithm 4 with all the static prediction algorithms. For both low and high sample rates $R = 0.3$ and $R = 0.7$, Algorithm 4 leveraging tensor properties significantly outperforms the static latency prediction methods. It verifies the significant benefit of utilizing multi-frames, and reveals the strong correlation between different latency frames over time. By exploiting the low-rank structure of all three unfolded matrices, Algorithm 4 takes full advantage of the implicit information in the tensor data.

Second, we compare the performance of all different versions of Algorithm 4 in Fig. 11(b), Fig. 11(e), Fig. 11(c) and Fig. 11(f), under different weight assignment schemes for the unfolded matrices $F_{(l)}$ for $l = 1, 2, 3$.

Fig. 11(b) and Fig. 11(e) compare various single unfolding schemes to Algorithm 4 with differentiation. Among all single unfolding schemes, Algorithm 4 performs similarly for $l = 1$ and 2, which outperforms $l = 3$. The reason is that if an entry is missing in all 3 frames, we cannot hope to recover it only based on $F_{(3)}$. The discrepancy between using the single unfolding $F_{(1)}$ (or $F_{(2)}$) and using $F_{(3)}$ is shrinking when the sample rate is high $(R = 0.7)$, because the chance that a node pair is missing in all 3 frames is small. This motivates us that we can benefit more from historical values of M_{ij} when they are available rather than using network condition correlations between different nodes for estimation, and weight differentiation in Algorithm 4 would improve the recovery performance of our algorithm.

We further evaluate the performance of Algorithm 4 with double unfolding, and show the results in Fig. 11(c) and Fig. 11(f). The weight assignments used for double unfolding are $\alpha_1 = 0.5, \alpha_2 = 0, \alpha_3 = 0.5$. As we can see, the algorithm with differentiation still outperforms the algorithm that minimizes the sum of the ranks of two unfolded matrices, at both high $(R = 0.7)$ and low $(R = 0.3)$ sample rates.

Through all the above comparisons, we show the benefits of incorporating multiple latency frames to perform multi-frame recovery, and the advantage of differentiated treatments to missing node pairs $(i,\ j) \in \mathbf{\Theta}_A$ and $(i,\ j) \in \mathbf{\Theta}_B$. Specifically, the third unfolded matrix $F_{(3)}$ is suitable for dealing with node pairs $(i,\ j) \in \mathbf{\Theta}_A$, while any of the first two unfolded matrices $F_{(1)}$ and $F_{(2)}$ are better to handle missing entries $(i,\ j) \in \mathbf{\Theta}_B$. It is shown that Algorithm 4 with such differentiation is optimal.

VIII. Concluding Remarks

In this paper, we measure the latency characteristics of the Seattle network which consists of personal devices, and revisit the problem of network latency prediction with the matrix completion approach. By decomposing the network latency matrix into a distance matrix and a network feature matrix, our approach extracts the noisy low-rank network features from a given incomplete RTT matrix and recovers all missing values through rank minimization. We propose a robust class of matrix completion algorithms, called IS-p, to approximate the rank minimization problem with reweighted Schatten-p norm minimization, and prove that the algorithm can converge for any p between 1 and 2. We further enhance the latency prediction with the help of partially collected historical observations forming a tensor, and extend our IS-p algorithm to the case of approximate tensor completion. Extensive evaluations based on the Seattle data show that our proposed algorithms outperform state-of-the-art techniques, including network embedding (e.g., high-dimensional Vivaldi with/without heights) and matrix factorization (e.g., DMFSGD) by a substantial margin, although they do not show much improvement on traditional PlanetLab data. This reveals the fact that our algorithms can better estimate latencies in personal device networks, for which traditional schemes are insufficient due to triangle inequality violation, asymmetric latencies and time-varying characteristics. The prediction accuracy is further significantly improved by exploiting the inherent autocorrelation property in the data sampled over multiple periods, through the proposed approximate tensor completion scheme.

References

[1] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A Decentralized Network Coordinate System," in *Proc. ACM SIGCOMM*, vol. 34, no. 4, 2004.

[2] T. E. Ng and H. Zhang, "Predicting Internet Network Distance with Coordinates-Based Approaches," in *Proc. IEEE INFOCOM*, 2002.

[3] Y. Liao, W. Du, P. Geurts, and G. Leduc, "DMFSGD: A Decentralized Matrix Factorization Algorithm for Network Distance Prediction," *IEEE/ACM Trans. Netw. (TON)*, vol. 21, no. 5, pp. 1511–1524, 2013.

[4] Y. Chen, X. Wang, C. Shi, E. K. Lua, X. Fu, B. Deng, and X. Li, "Phoenix: A Weight-Based Network Coordinate System Using Matrix Factorization," *IEEE Trans. Netw. Service Manag.*, vol. 8, no. 4, pp. 334–347, 2011.

[5] G. Wang, B. Zhang, and T. Ng, "Towards Network Triangle Inequality Violation Aware Distributed Systems," in *Proc. ACM SIGCOMM IMC*, 2007.

[6] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang, "Characterizing and Modeling Internet Traffic Dynamics of Cellular Devices," in *ACM SIGMETRICS Performance Evaluation Review*, 2011, pp. 305–316.

[7] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, "Seattle: A Platform for Educational Cloud Computing," in *Proc. ACM SIGCSE*, 2009.

[8] K. Mohan and M. Fazel, "Iterative Reweighted Algorithms for Matrix Rank Minimization," *The Journal of Machine Learning Research (JMLR)*, vol. 13, no. 1, pp. 3441–3473, 2012.

[9] J.-F. Cai, E. J. Candès, and Z. Shen, "A Singular Value Thresholding Algorithm for Matrix Completion," *SIAM J. Optim.*, vol. 20, no. 4, pp. 1956–1982, 2010.

[10] [Online]. Available: https://github.com/uofa-rzhu3/NetLatency-Data

[11] B. Donnet, B. Gueye, and M. A. Kaafar, "A Survey on Network Coordinates Systems, Design, and Security," *IEEE Commun. Surveys & Tutorials*, vol. 12, no. 4, 2010.

[12] J. Ledlie, P. Gardner, and M. I. Seltzer, "Network Coordinates in the Wild," in *Proc. USENIX NSDI*, 2007.

[13] S. Lee, Z.-L. Zhang, S. Sahu, and D. Saha, "On Suitability of Euclidean Embedding of Internet Hosts," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, no. 1, 2006, pp. 157–168.

[14] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu, "Spatio-Temporal Compressive Sensing and Internet Traffic Matrices," in *Proc. ACM SIGCOMM*, vol. 39, no. 4. ACM, 2009, pp. 267–278.

[15] Y. Mao, L. K. Saul, and J. M. Smith, "IDES: An Internet Distance Estimation Service for Large Networks," *IEEE J. Sel. Areas Commun. (JSAC)*, vol. 24, no. 12, pp. 2273–2284, 2006.

[16] E. J. Candès and B. Recht, "Exact Matrix Completion via Convex Optimization," *Found. Comut. Math*, vol. 9, no. 6, pp. 717–772, 2009.

[17] 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 *Proc. IEEE INFOCOM*, 2015, pp. 2443–2451.

[18] B. Liu, D. Niu, Z. Li, and H. V. Zhao, "Network Latency Prediction for Personal Devices: Distance-Feature Decomposition from 3D Sampling," in *Proc. IEEE INFOCOM*, 2015.

[19] K. LaCurts and H. Balakrishnan, "Measurement and Analysis of Real-World 802.11 Mesh Networks," in *Proc. ACM SIGCOMM IMC*, 2010.

[20] J. Sommers and P. Barford, "Cell vs. WiFi: On the Performance of Metro Area Mobile Connections," in *Proc. ACM SIGCOMM IMC*, 2012.

[21] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A Close Examination of Performance and Power Characteristics of 4G LTE Networks," in *Proc. ACM MobiSys*, 2012.

[22] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: An Overlay Testbed for Broad-Coverage Services," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 3, pp. 3–12, 2003.

[23] G. H. Golub and C. Reinsch, "Singular Value Decomposition and Least Squares Solutions," *Numerische Mathematik*, vol. 14, no. 5, pp. 403–420, 1970.

[24] L. Tang and M. Crovella, "Virtual Landmarks for the Internet," in *Proc. ACM SIGCOMM IMC*, 2003.

[25] J. Gower, "Properties of Euclidean and Non-Euclidean Distance Matrices," *Linear Algebra Appl.*, vol. 67, no. 0, pp. 81–97, 1985.

[26] Y. Zhang and Z. Lu, "Penalty Decomposition Methods for Rank Minimization," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2011.

[27] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed Minimum-Rank Solutions of Linear Matrix Equations via Nuclear Norm Minimization," *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.

[28] I. Daubechies, R. DeVore, M. Fornasier, and C. S. Güntürk, "Iteratively Reweighted Least Squares Minimization for Sparse Recovery," *Comm. Pure Appl. Math*, vol. 63, no. 1, pp. 1–38, 2010.

[29] M. S. Lobo, M. Fazel, and S. Boyd, "Portfolio Optimization with Linear and Fixed Transaction Costs," *Ann. Oper. Res.*, vol. 152, no. 1, pp. 341–365, 2007.

[30] E. J. Candès and T. Tao, "The Power of Convex Relaxation: Near-optimal Matrix Completion," *IEEE Trans. Info. Theory (TIT)*, vol. 56, no. 5, pp. 2053–2080, 2010.

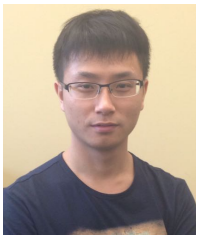
[31] V. De Silva and L.-H. Lim, "Tensor Rank and the Ill-Posedness of the Best Low-Rank Approximation Problem," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1084–1127, 2008.

[32] C. J. Hillar and L.-H. Lim, "Most Tensor Problems Are NP-Hard," *Journal of the ACM (JACM)*, vol. 60, no. 6, p. 45, 2013.

[33] S. Gandy, B. Recht, and I. Yamada, "Tensor Completion and Low-n-Rank Tensor Recovery via Convex Optimization," *Inverse Problems*, vol. 27, no. 2, 2011.

[34] I. Olkin, A. W. Marshall, and B. C. Arnold, *Inequalities: Theory of Majorization and Its Applications*, 2nd ed., ser. Springer Series in Statistics. New York: Springer, 2011.

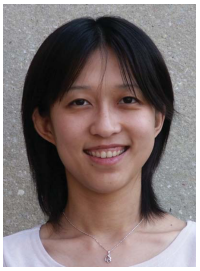
Rui Zhu received the B.E. degree in Electrical and Information Engineering from Xidian University, Xi'an, China, in 2011 and the M.Sc degree in cryptography from Xidian University, Xi'an, China, in 2014. Since September 2014, he has been pursuing the Ph.D. degree at the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada. His research interests include cloud computing, statistical machine learning for networking, and information theory.



Bang Liu received the B.E. degree in Electronic Information Science from University of Science and Technology of China, Hefei, China, in 2013 and the M.Sc. degree in Computer Engineering from University of Alberta, Edmonton, Canada, in 2015. Since January 2016, he has been pursuing the Ph.D. degree at the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada. His research interests include statistical machine learning for networking, spatial data analysis, and natural language processing.

Di Niu received the B.Engr. degree from the Department of Electronics and Communications Engineering, Sun Yat-sen University, China, in 2005 and the M.A.Sc. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada, in 2009 and 2013. Since 2012, he has been with the Department of Electrical and Computer Engineering at the University of Alberta, where he is currently an Assistant Professor. His research interests span the areas of cloud computing and storage, data mining and statistical machine learning for social and economic computing, and distributed and parallel systems. He is a member of IEEE and ACM.

Zongpeng Li received his B.E. degree in Computer Science and Technology from Tsinghua University (Beijing) in 1999, his M.S. degree in Computer Science from University of Toronto in 2001, and his Ph.D. degree in Electrical and Computer Engineering from University of Toronto in 2005. Since 2005, he has been with the University of Calgary, where he is now Professor of Computer Science. In 2011-2012, Zongpeng was a visitor at the Institute of Network Coding, Chinese University of Hong Kong. His research interests are in computer networks, network coding, cloud computing, and energy networks.



H. Vicky Zhao received the B.S. and M.S. degree from Tsinghua University, China, in 1997 and 1999, respectively, and the Ph. D degree from University of Maryland, College Park, in 2004, all in electrical engineering. She was a Research Associate with the Department of Electrical and Computer Engineering and the Institute for Systems Research, University of Maryland, College Park from Jan. 2005 to July 2006. Since 2006, she has been with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada, where she is now an Associate Professor. Her research interests include information security and forensics, multimedia social networks, digital communications, and signal processing.

APPENDIX
PROOF FOR LEMMA 1

In this appendix, we present our proof of Lemma 1, the key lemma in the proof of convergence of Algorithm 2. Before our formal proof, we firstly introduce some notations and preliminaries for matrix inequalities, which play an important role in our proof for Lemma 1.

Let A be an $n \times n$ matrix. The vector of eigenvalues of A is denoted by $\lambda(A) = (\lambda_1(A), \lambda_2(A), \ldots, \lambda_n(A))$, and they are ordered as $\lambda_1(A) \geq \lambda_2(A) \geq \ldots \geq \lambda_n(A) \geq 0$. The vector of diagonal elements of A is denoted by $d(A) = (d_1(A), d_2(A), \ldots, d_n(A))$. For any A, there exists the singular value decomposition. The singular values are arranged in decreasing order and denoted by $\sigma_1(A) \geq \sigma_2(A) \geq \ldots \geq \sigma_n(A) \geq 0$.

It is clear that the singular values of A are the nonnegative square roots of the eigenvalues of the positive semidefinite matrix $A^\top A$, or equivalently, they are the eigenvalues of the positive semidefinite square root $(A^\top A)^{1/2}$, so that $\sigma_i(A) = [\lambda_i(A^\top A)]^{1/2} = \lambda_i[(A^\top A)^{1/2}], \quad (i = 1, \ldots, n)$.

We then introduce the theory of majorization, one of the most powerful techniques for deriving inequalities. Given a real vector $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$, we rearrange its components as $x_{[1]} \geq x_{[2]} \geq \ldots \geq x_{[n]}$. The definition of majorization is as follows. For $x, y \in \mathbb{R}^n$, if

$$\sum_{i=1}^{k} x_{[i]} \leq \sum_{i=1}^{k} y_{[i]} \text{ for } k = 1, \ldots, n-1$$

and

$$\sum_{i=1}^{n} x_{[i]} = \sum_{i=1}^{n} y_{[i]},$$

then we say that x is *majorized* by y and denote $x \prec y$. If

$$\sum_{i=1}^{n} x_{[i]} \leq \sum_{i=1}^{n} y_{[i]},$$

we say that x is *weakly majorized* by y and denote $x \prec_w y$. We introduce some properties for majorization and weak majorization, which can be found in quite a wide range of literature, e.g. [34]. Interested readers can find detailed proofs in these references.

Lemma 2 (cf. [34], Ch. 1)**.** *Let $g(t)$ be an increasing and convex function. Let $g(x) := (g(x_1), g(x_2), \ldots, g(x_n))$ and $g(y) := (g(y_1), g(y_2), \ldots, g(y_n))$. Then, $x \prec_w y$ implies $g(x) \prec g(y)$.*

Theorem 3 (cf. [34], Ch. 9)**.** *If A is a Hermitian matrix (real symmetric for a real matrix A), then we have $d(A) \prec \lambda(A)$.*

Note that the singular values of A are the eigenvalues of the positive semidefinite matrix $A^\top A$. We then have:

Corollary 4. *If A is a real symmetric matrix, and we denote $|A|$ as the positive semidefinite square root of $A^\top A$, we have $d(|A|) \prec \lambda(|A|) = \sigma(A)$.*

Lemma 3 (cf. [34], Ch. 9)**.** *For any matrices A and B, we have $\sigma(AB) \prec_w \sigma(A) \circ \sigma(B)$, where $\circ$ denotes the Hadamard product (or entry-wise product).*

Lemma 4 (Abel's Lemma)**.** *For two sequences of real numbers $a_1, \ldots, a_n$ and $b_1, \ldots, b_n$, we have*

$$\sum_{i=1}^{n} a_i b_i = \sum_{i=1}^{n-1} (a_i - a_{i+1}) \left(\sum_{j=1}^{i} b_i \right) + a_n \sum_{i=1}^{n} b_i.$$

Lemma 5. *If $x \prec y$ and $w = (w_1, w_2, \ldots, w_n)$, where $0 \leq w_1 \leq w_2 \leq \ldots \leq w_n$, we have*

$$\sum_{i=1}^{n} w_i x_i \geq \sum_{i=1}^{n} w_i y_i.$$

Proof. For any $1 \leq k < n$, we have

$$\sum_{i=1}^{k} x_i \leq \sum_{i=1}^{k} y_i.$$

Then, since $w_k \leq w_{k+1}$, we have

$$(w_k - w_{k+1}) \sum_{i=1}^{k} x_i \geq (w_k - w_{k+1}) \sum_{i=1}^{k} y_i$$

In addition, for $k = n$, we have

$$w_n \sum_{i=1}^{n} x_i = w_n \sum_{i=1}^{n} y_i,$$

since $x \prec y$ implies that the summation of x_i and y_i are identical. Summing up all n inequalities above, we have

$$\begin{aligned} & \sum_{k=1}^{n-1} (w_k - w_{k+1}) \left(\sum_{i=1}^{k} x_i \right) + w_n \sum_{i=1}^{n} x_i \\ \geq & \sum_{k=1}^{n-1} (w_k - w_{k+1}) \left(\sum_{i=1}^{k} y_i \right) + w_n \sum_{i=1}^{n} y_i. \end{aligned} \quad (18)$$

By applying the Abel's Lemma for both sides, we have

$$\sum_{k=1}^{n} w_k x_k \geq \sum_{k=1}^{n} w_k y_k,$$

which proves the lemma. □

Theorem 5 (cf. [34], Ch. 9)**.** *If A and B are two positive semidefinite matrices, then*

$$\begin{aligned} \mathrm{tr}(AB)^\alpha &\leq \mathrm{tr}(A^\alpha B^\alpha), \quad \alpha > 1, & (19) \\ \mathrm{tr}(AB)^\alpha &\geq \mathrm{tr}(A^\alpha B^\alpha), \quad 0 < \alpha \leq 1. & (20) \end{aligned}$$

Now we are ready to prove our Lemma 1 as follows:

Proof of Lemma 1. The right inequality is a consequence of Lemma 3. To see this, let $g(t) = t^p$ for all $1 \leq p \leq 2$. For all $t \geq 0$, $g(t)$ is an increasing and convex funciton. Thus, we have $\sigma(AB) \prec_w \sigma(A) \circ \sigma(B)$ from Lemma 3, and from Lemma 2 we have $g(\sigma(AB)) \prec_w g(\sigma(A) \circ \sigma(B))$ and it implies that

$$\sum_{i=1}^{n} \sigma_i^p(AB) \leq \sum_{i=1}^{n} \sigma_i^p(A) \sigma_i^p(B).$$

So now we can focus on the left inequality. Here we denote $|A|$ as the positive semidefinite square root of $A^\top A$. Suppose

the singular value decomposition of A is $U_A\Sigma_A V_A^\top$, and that of B is $U_B\Sigma_B V_B^\top$. By the unitary invariance of the singular values and the Schatten-p norm, we have

$$||AB||_p^p = ||U_A\Sigma_A V_A^\top U_B\Sigma_B V_B^\top||_p^p \tag{21}$$
$$= ||(\Sigma_A V_A^\top U_B)\Sigma_B||_p^p \tag{22}$$
$$= ||A_1 B_1||_p^p. \tag{23}$$

Here we let $A_1 := \Sigma_A V_A^\top U_B$ and $B_1 := \Sigma_B$. Thus, without loss of generality, we can assume that B is diagonal. Then, from the definition of Schatten-p norm, we have

$$\begin{aligned} ||AB||_p^p &= \mathrm{tr}(|AB|^p) = \mathrm{tr}\left(\sqrt{B^\top A^\top AB}\right)^p \\ &\geq \mathrm{tr}((B^\top)^{\frac{p}{2}}(A^\top A)^{\frac{p}{2}}B^{\frac{p}{2}}) \qquad (24) \\ &= \mathrm{tr}((BB^\top)^{\frac{p}{2}}(A^\top A)^{\frac{p}{2}}) \\ &= \mathrm{tr}(|B|^p(A^\top A)^{\frac{p}{2}}) \\ &= \mathrm{tr}(|B|^p|A|^p) \qquad (25) \end{aligned}$$

Here (24) is from (20) in Theorem 5, since $|B|$ is diagonal with all nonnegative entries, and $A^\top A$ is a real symmetric matrix. Since B is diagonal, $d(|B|)$ is just a permutation of its singular value vector $\sigma(B)$. Thus, we can simply rearrange the order of sum in (25) as

$$\mathrm{tr}(|B|^p|A|^p) = \sum_{i=1}^n d_i(|B|)d_i(|A|) \tag{26}$$
$$= \sum_{i=1}^n d_{[n-i+1]}(|B|)d_{\pi(i)}(|A|), \tag{27}$$

where $\pi(\cdot)$ is a permutation indicating the order of the new summation, and $d_{[i]}(|B|) = \sigma_i(B)$. From Lemma 4, we can see that $d(|A|) \prec \sigma(A)$, and by Lemma 5, we finally have

$$||AB||_p^p = \sum_{i=1}^n d_{[n-i+1]}^p(|B|)d_{\pi(i)}^p(|A|) \tag{28}$$
$$= \sum_{i=1}^n \sigma_{n-i+1}^p(B)d_{\pi(i)}^p(|A|) \tag{29}$$
$$\geq \sum_{i=1}^n \sigma_{n-i+1}^p(B)\sigma_i^p(A). \tag{30}$$

□