Matrix completion based adaptive sampling for measuring network delay with online support

Wei Meng and Laichun Li
Department of Computer Science and Technology, Beihua University, Jilin 132013, China
[mengwei18604498032@163.com, llchun18604498024@163.com]
*Corresponding author: Laichun Li

Received November 1, 2019; revised January 31, 2020; accepted March 15, 2020; published July 31, 2020

Abstract

End-to-end network delay plays an vital role in distributed services. This delay is used to measure QoS (Quality-of-Service). It would be beneficial to know all node-pair delay information, but unfortunately it is not feasible in practice because the use of active probing will cause a quadratic growth in overhead. Alternatively, using the measured network delay to estimate the unknown network delay is an economical method. In this paper, we adopt the state-of-the-art matrix completion technology to better estimate the network delay from limited measurements. Although the number of measurements required for an exact matrix completion is theoretically bounded, it is practically less helpful. Therefore, we propose an online adaptive sampling algorithm to measure network delay in which statistical leverage scores are used to select potential matrix elements. The basic principle behind is to sample the elements with larger leverage scores to keep the traits of important rows or columns in the matrix. The amount of samples is adaptively decided by a proposed stopping condition. Simulation results based on real delay matrix show that compared with the traditional sampling algorithm, our proposed sampling algorithm can provide better performance (smaller estimation error and less convergence pressure) at a lower cost (fewer samples and shorter processing time).

Keywords: Network delay, network measurement, matrix completion, adaptive sampling, leverage score

http://doi.org/10.3837/tiis.2020.07.018 **ISSN : 1976-7277**

1. Introduction

Network delay, also called network distance, plays an vital role in global distributed services and applications, such as content distribution networks (CDNs) [1], peer-to-peer file sharing [2], [3], proximity-aware distributed hash tables [4] and overlay routing [5] etc. The advantage of these systems is that they can flexibly select communication peers that respond quickly, where the end-to-end network delay is used as an indicator of QoS (Quality-of-Service). For example, a client in a CDN can choose the mirror site with the highest bandwidth to download web objects.

However, it is not feasible to obtain the network delay between all transmission pairs through network measurements, particularly in large-scale networks. Since the cost of active probing increases quadratically with the size of the network, on-demand measurements are expensive and time-consuming. Therefore, an idea of estimating the unknown pair-wise network delay from limited observed measurements, arises and motivates plentiful research on this topic.

A Network Coordinate System (NCS) [6]-[8] predicts network delays as Euclidean distances in a virtual geometric space by mapping network nodes to coordinates in this space, which is also known as Euclidean embedding. This kind of approach makes assumptions of certain Euclidean distance properties, e.g., symmetry and triangle inequality. However, such assumptions may not always hold, e.g., triangle inequality violations (TIVs) are commonly found in practice [9], [10]. To overcome these problems, matrix factorization are used [11], [12]. The basic idea is to decompose an approximated large network delay matrix into two smaller matrices, which can represent the asymmetric delays or the delays that violate triangle inequality. Matrix factorization based approaches model network delays by a matrix with a fixed rank. However, the exact rank of the delay matrix can hardly be obtained in advance.

In recent years, matrix completion by rank minimization has become an important research direction for network delay estimation[13], [14]. These approaches, compared to Euclidean embedding or matrix factorization, are not limited by triangle inequality or predetermined rank. Several studies [15]-[18] reported sufficient conditions to recover low-rank a matrix from some observations, that is the minimum number of samples required to successfully recover a high probability matrix. However, there are some limits under these conditions: 1) the matrix elements are uniformly and randomly sampled, and 2) the incoherence property of the matrix is assumed, that is the singular vectors of the matrix are not concentrated on a few coordinates, but are roughly "expanded" uniformly. In real-world applications, the subset of observed elements is far from uniformly random [17]. For example, in the application of network delay measurement, we may not have the control of accessing any paired link to fulfill the uniform-sampling requirement. Moreover, the incoherence assumption is necessary due to the uniform sampling [19]. By such sampling, most of the mass of a coherent matrix could be missed since it is distributed among just a few values. On the contrary, if the elements with more mass can be sampled with higher probability, it would be possible to recover the matrix without incoherent requirement.

In addition, although the number of samples used to successfully recovering the unknown low-rank matrix is theoretically bounded, how to find the minimum number of samples still requires practical guidance. Therefore, this paper mainly studies the sampling strategy in practical applications such as network delay measurement. Specifically, we deal with the

following two main problems: 1) which node-pair paths are selected for measurement, and 2) how many path samples are needed to successfully recover an unknown delay matrix.

An online adaptive sampling algorithm is invoked to address these problems. Specifically, we use statistical leverage score [20], [21] to select node-pair paths for measurement and a stopping criterion to determine the number of samples to complete the sampling. The basic principle behind is to sample the elements with larger leverage scores to keep the traits of important rows or columns in the matrix. The whole process of the proposed sampling algorithm is shown in **Fig. 1**, where five function blocks (red parts in brackets) are needed and will be described in detail in Section 4. After the initialization, leverage scores are used to quantify the local coherence property of the matrix, based on which the probability of each element is calculated. The sampling set is then updated by the ones with higher probability and the unknown matrix is recovered from these partial observations. The sampling process proceeds until a stopping criterion is reached.

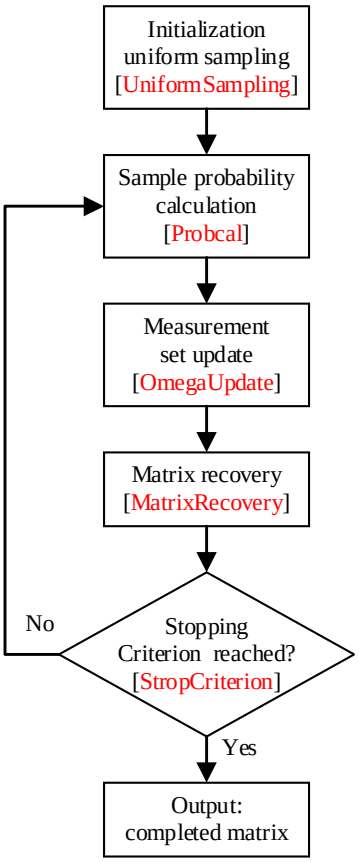


Fig. 1. Block diagram of the proposed online adaptive sampling algorithm (The red parts in brackets correspond to the specific function blocks described in Section 4.4)

The rest of the paper is organized as follows. Section 2 introduces related work on network delay estimation from limited measurements. Section 3 elaborates the problem in this paper. Section 4 describes the proposed online adaptive sampling algorithm for network delay measurement in detail. Section 5 evaluates the performance of the proposed algorithm based on real-world data of application-level RTTs (Round-Trip Times) and compares it with other traditionally used algorithms. Finally, Section 6 summarizes the whole paper and gives our future work.

2. Related Work

Euclidean embedding based approaches have been proposed to estimate network delay without performing direct measurements. The key idea is to model the target network as a geometric space (typically a n-dimensional Euclidean space) and localize each node within the network by assigning them coordinates. The Euclidean distance between any two nodes, calculated by using these coordinates, can be used to predict network delay.

In general, Euclidean embedding can be divided into two categories: landmark-based and decentralized approaches. A standard landmark-based system is Global Network Positioning (GNP) [6] which firstly embeds a network into a n-dimensional geometric space. GNP system requires at least n+1 landmarks to estimate a unique set of host coordinates. The landmarks need all-pairs measurements among themselves in order to obtain their coordinates in the space. The coordinates of an ordinary host are then calculated based on the coordinates of landmarks and the direct measurements between the host and all landmarks. To remove the constraints brought by landmarks, decentralized approaches are introduced to extend the landmark to any node or eliminate the fixed infrastructure completely. One representative implementation is Vivaldi [7] where each node within the network updates its coordinates with new measurements based on spring relaxation to find the minimal energy configurations.

The performance of Euclidean embedding usually has the assumptions of distance symmetry and triangle inequality which may not hold in practice. Matrix factorization, on the contrary, overcomes these problems by assuming a fixed rank to the network delay matrix. Internet Distance Estimation Service (IDES) [11] employs Singular Value Decomposition (SVD) or Nonnegative Matrix Factorization (NMF) to recover a delay matrix in a landmark-based manner. DMFSGD algorithm proposed in [12] further settles the matrix factorization problem by using stochastic gradient descent in a decentralized way.

Instead of assigning a fixed rank to the matrix, matrix completion through rank minimization has been recently adopted to estimate network delay. In [14], the authors solve a rank minimization problem without requiring a priori knowledge about the rank of the matrix for estimating network latency of personal device. The first bounds on the number required for an $n*n$ matrix with rank r to be fully recovered by solving rank minimization is given in [15], where the bounds are proved to be $O(n^{1.2}rlogn)$ under assumptions of uniform sampling and incoherency. This result is further improved in [18] to be $O(\eta rnlog^2 n)$ where η was the coherence of a matrix (See Section IV-A). Although the number of measurements required for an exact matrix recovery is theoretically bounded, it is practically less helpful. In [13], the authors propose an information-based sampling scheme for matrix completion in network monitoring systems. However, as a nonuniformal sampling, the coherence of matrix is not considered. Moreover, based on our experimental observations, it is difficult for the algorithm to converge.

Moreover, in [30], the authors invoked an active scheme that could measure the queuing delay on the Internet routers through an end-to-end path. The proposed solution used User Datagram Protocol (UDP) based probing packets to measure in a hop-by-hop manner, which was simple and self-sufficient. In [31], a novel Matrix Completion Technique based Data Collection (MCTDC) technique. Specifically, through matrix completion technique, the authors explored the multi-dimensional correlation of data to reduce the amount of data required while ensuring the QoS. In addition, in order to select the smallest set of suitable participants, the authors redefined the contribution degree as the ratio of the effective data from a given participant and the total amount of data. In [32], the authors designed a scalable and memory-efficient algorithm that used stochastic proximal gradient descent (SPGD)

method to deal with large-scale network problems. Specifically, the inherent low rank and time characteristics of the traffic matrix were used to transform the network estimation problem into a noise immune temporal matrix completion (NiTMC) model, and the mixture of Gaussian (MoG) was used to fit complex noise. In addition, the authors devised a convergence-guaranteed optimization algorithm based on the expectation maximization (EM) and block coordinate update (BCU) methods to solve the proposed model.

Table 1. Summary of the used notations.

Item	Definition
M, X	square matrix
r	the rank of a matrix
Ω	the set of locations corresponding to the sampled elements
P_Ω	the orthogonal projector onto the space of the matrix vanishing outside of Ω
k	sampling epoch
$\| X \|_* = \sum_{i=1}^{n} \sigma_i(X)$	nuclear norm of X
σ_i	singular value
$I_r \in R^{r\times r}$	an identity matrix
$\Sigma \in R^{r\times r}$	the diagonal matrix with elements equal to the r singular values of M
p_{ij}	the probability of each eatry $(i, j), i, j \in (1, 2, \ldots n)$
$c = \dfrac{\|\Omega\|}{3crnlog^2(2n)}$	a universal constant factor
$D_T(\cdot)$	a shrink function
$\hat{X}(k \mid m)$	the estimated matrix with m samples at the k^{th} epoch
$\| X \|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} \|X_{ij}\|^2}$	the Frobenius norm of X
ε	a fixed tolerance, e.g., $\varepsilon = 10^{-3}$.
δ_k	the step size. For each iteration
γ	a probability threshold

3. Problem Formulation

Consider an unknown matrix $M \in R^{n_1\times n_2}$ with its rank $r \ll min(n1, n2)$. Here we assume M is a square matrix (i.e., $n_1 = n_2 = n$) for simplicity which does not lose the generality. The goal of matrix completion is to recover the full matrix information from a finite observable elements $M_{ij}, (i, j) \in \Omega$ where Ω is the set of locations corresponding to the sampled elements (i.e., $(i, j) \in \Omega$ if M_{ij} is sampled) with cardinality m. We can restore the matrix by minimizing the rank of the matrix

$$\min_X \quad rank(X) \quad \text{subject to } X_{ij} = M_{ij} \quad (i, j) \in \Omega \tag{1}$$

As the value of a matrix rank is equal to the number of nonzero singular values of the matrix, the rationale behind is that the rank function in (1) counts the number of nonvanishing singular values. However, Problem (1) is NP-hard.

Alternatively, the matrix can be recovered by solving the following objective function

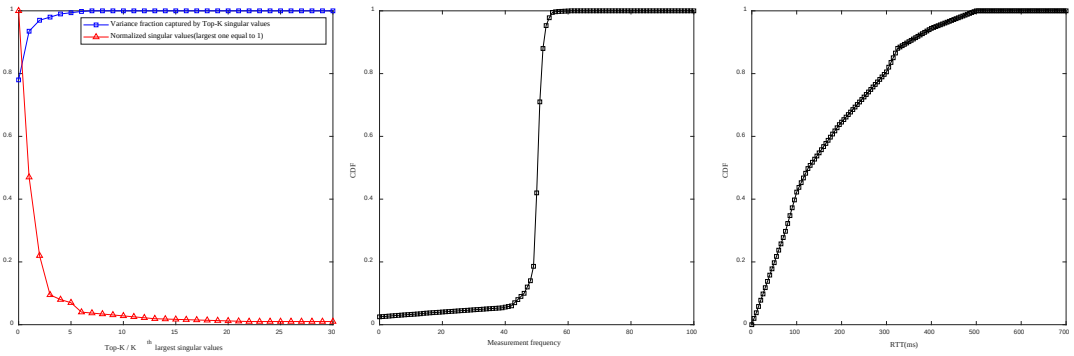
$$\min_{X} \quad \| X \|_* \quad \text{subject to } P_\Omega(X) = P_\Omega(M) \tag{2}$$

where $\| X \|_* = \sum_{i=1}^{n} \sigma_i(X)$ is the nuclear norm of X, the sum of the singular value amplitudes of the matrix. P_Ω is the orthogonal projector onto the space of the matrix vanishing outside of Ω

$$P_\Omega(X)_{ij} = \begin{cases} X_{ij} & (i, j) \in \Omega \\ 0 & otherwise \end{cases} \tag{3}$$

Here, $\| X \|_*$ in (2) is a convex function and can be solved by various methods such as semidefinite programming [15], iterative thresholding [22], or alternating minimization [23].

Unlike previous research on solving Problem (2) directly, this paper focuses on how to construct sampling set Ω in (2) to better estimate the network delay matrix from the perspective of practical application. The construction of Ω includes 1) identifying indexes (i, j) belonging to Ω, and 2) the cardinality m of Ω.



(A) (B) (C)

Fig. 2. Characteristic of Harvard-226 RTT dataset: (A) low-rank property (sum of top-6 singular values $\sum_{i=1}^{6} \sigma_i$ capture 99% of the total energy of the matrix (blue square curve); singular values after σ_6 start to fall below 5% of the largest one σ_1 (red triangle curve)); (B) measurement frequency (3.9% of all the pair-node paths are not measured; most of the measured paths locates in the range of 40 to 56 times) and (C) RTT distribution among the pair-node paths.

For exact recovery of a network delay matrix, a premise is that the matrix should be low-rank. **Fig. 2A** shows the fraction of total variance captured by top-k singular values ($\sum_{i=1}^{k} \sigma_i / \sum_{i=1}^{n} \sigma_i$ blue square curve) of a typical RTT matrix from Harvard-226 dataset [12], [24]. It can be seen that the largest singular value σ_1 solely captures over 75% of the total energy of the matrix. σ_6 can be regarded as a threshold since the sum of top-6 singular values

$\sum_{i=1}^{6}\sigma_i$ already capture 99% of the total energy. **Fig. 2A** also shows the normalized singular values of the RTT matrix (red triangle curve) where σ_1 is normalized to be 1. The singular value decreases fast and those after σ_6 start to fall below 5% of σ_1.

This observed low-rank characteristic of delay matrix is consistent with previous research on various types of network delay measurements [14], [25]. The overlapped or shared bottleneck links among network paths with nearby end nodes, leads to the redundancy in the use of links across paths [12]. This therefore results in delay measurements on different paths being correlated, resulting in a matrix with a low or approximately low rank.

4. Adaptive Sampling Algorithm

In this section, the proposed online adaptive sampling algorithm is described in detail. We first present the statistical leverage score, and then the sampling stopping criterion. Next, singular value thresholding [22] as an example algorithm to recover an unknown matrix is introduced. Finally, we summarize the complete sampling algorithm.

4.1 Leverage Score Based Sampling

Consider rank-r SVD (Singular Value Decomposition) of $M \in R^{n\times n}$, $M = U\Sigma V^T$, where U, $V \in R^{n\times r}$ have orthonormal columns with $U^TU = V^TV = I_r$ ($I_r \in R^{r\times r}$ is an identity matrix) and $\Sigma \in R^{r\times r}$ is a diagonal matrix with elements equal to the r singular values of M.

The commonly used matric to quantify the importance of rows and columns of M are its leverage scores and here the normalized ones are used and defined as

$$\mu_i(M) = \frac{n}{r}\|U^Te_i\|_2^2 = \frac{n}{r}\|U(i,:)\|_2^2 \tag{4}$$

$$\nu_j(M) = \frac{n}{r}\|V^Te_j\|_2^2 = \frac{n}{r}\|V(j,:)\|_2^2 \tag{5}$$

Where e_i / e_j denotes the i^{th} / j^{th} standard basis element with appropriate dimensions, and $\mu_i(M), \nu_j(M) \in [0, \frac{n}{r}]$ for all $i, j \in (1, 2, \ldots n)$. It can be easily verified that $\sum_{i=1}^{n}\mu_i(M) = \sum_{j=1}^{n}\nu_j(M) = n$, since the summation of diagonal elements of UU^T, VV^T both equals to r.

It is proved in [26] that, if each element in M is selected according to its leverage scores $\mu_i(M) + \nu_j(M)$ with a probability $p_{ij}, i, j \in (1, 2, \ldots n)$ satisfying

$$p_{ij} \geq min\left(c(\mu_i(M) + \nu_j(M))\frac{rlog^2(2n)}{n}, 1\right) \tag{6}$$

Then there is a great chance for the nuclear norm minimization problem (2) to have a unique solution, c in (6) is a universal constant factor. We adopt (6) as the basis for selecting adaptively the entries to be measured in a network delay matrix and develop a practical

implementation for real-world applications.

Based on (6), the required amount of samples to successfully recover rank-r matrix is

$$E[p]=\sum_{i=1}^{n}\sum_{j=1}^{n}p_{ij}\geq c\frac{rlog^2(2n)}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}(\mu_i(M)+\nu_j(M))\geq crnlog^2(2n) \tag{7}$$

Therefore the matrix can be recovered from $O(rnlog^2 n)$ samples. On the contrary, for uniform sampling with an incoherence assumption mentioned previously, the number of required samples is around $O(\eta(M)rnlog^2 n)$ with $\eta(M)=max\{\max_{i\in(1,n)}\mu_i,\max_{j\in(1,n)}\nu_j\}$. Therefore, by removing the incoherence constraint, the sample complexity can be improved by up to a factor of n/r [27].

Although the required number of observed elements can be theoretically bounded by (7), it is practically less helpful because a universal constant factor c can hardly be found. According to Hoeffding's inequality that the number of observed elements concentrates around the expectation, a loose relaxation of (7) can then be assigned to the dimension of sampling set Ω

$$|\Omega|=3crnlog^2(2n) \tag{8}$$

Therefore $c=\frac{|\Omega|}{3crnlog^2(2n)}$, substituting it into (6), the probability of each eatry $(i,j),i,j\in(1,2,\ldots n)$ in the proposed sampling algorithm is obtained as

$$p_{ij}=min\left(\frac{|\Omega|}{3n^2}(\mu_i(M)+\nu_j(M)),1\right) \tag{9}$$

4.2 Sampling Stopping Criterion

Just having (9) is not enough since Ω may not contain adequate samples for successfully recovering the matrix. In order to solve this problem, we adaptively sample elements until certain criteria are reached.

It is pointed out in [13] that a low-rank matrix X can be exactly recovered if the following condition holds

$$\hat{X}(k\,|\,m)=\hat{X}(k+1\,|\,m+C) \tag{10}$$

Where $\hat{X}(k\,|\,m)$ is the estimated matrix with m samples at the k^{th} epoch and $\hat{X}(m+C)$ is recovered with additional C samples at the $(k+1)^{th}$ epoch. Here we share similar heuristic given as follows.

Sampling stopping criterion. Consider estimated network delay matrix at the k^{th} epoch, $\hat{X}(k)$, and the one estimated at the $(k+1)^{th}$ epoch, $\hat{X}(k+1)$. The adaptive sampling procedure can be considered to stop if $\hat{X}(k)$ and $\hat{X}(k+1)$ satisfy

$$\frac{\| \hat{X}(k+1) - \hat{X}(k) \|_F}{\| \hat{X}(k) \|_F} \leq \varepsilon \tag{11}$$

Where $\| X \|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} |X_{ij}|^2}$ denotes the Frobenius norm of X and ε is a fixed tolerance, e.g., $\varepsilon = 10^{-3}$.

4.3 Matrix Recovery Algorithm

A matrix recovery algorithm is used to solve Problem (2). Note that the proposed adaptive sampling algorithm is by no means necessarily restricted to any specific recovery algorithm for obtaining $\hat{X}$. Methods such as singular value thresholding (SVT) [22], interior-point method [28], or iterative reweighted least squares (IRLS) [29] can be used as long as they fulfill the requirements by applications on accuracy, computational overhead, or processing time etc. In this paper, we take SVT as an example to solve Problem (2) because it is easy to implement its design. The complete SVT algorithm is given as follows

$$X(k) = D_T(Y(k-1)) \tag{12}$$

$$Y(k) = Y(k-1) + \delta_k P_\Omega(M - X(k)) \tag{13}$$

where $D_T(Y) = D_T(U_Y \Sigma_Y V_Y^T) = U_Y D_T(\Sigma_Y) V_Y^T$ is a shrink function which performs r-level ($r \geq 0$) soft thresholding on singular values of s matrix that

$$D_T(\Sigma_Y) = diag(max(0, \sigma_i - \tau)_{1 \leq i \leq r_Y}) \tag{14}$$

And δ_k is the step size. For each iteration, SVT conducts SVD only once and performs simple matrix operations. In practical implementations, the soft thresholding $D_r(\cdot)$ can apply to sparse vectors which saves much memory space and accelerates processing time.

It can be verified that $X(k)$ finally converges to the unique solution of the following problem [22]

$$\min_X \quad \tau \| X \|_* + \frac{1}{2} \| X \|_F^2 \text{ subject to } P_\Omega(X) = P_\Omega(M) \tag{15}$$

When $r \to \infty$, Problem (15) converges to Problem (2). Therefore, for large values of τ, (12) and (13) iteratively find an $\hat{X}$ approximating the unique optimal solution to Problem (2).

4.4 Complete Sampling Algorithm

Recall the sampling process shown in **Fig. 1**. At the beginning, the sampling set Ω is initialized by uniformly sampling m measurements at random and these measurements are used to recover the first complete matrix $\hat{X}(0)$. Once the information on every entry is obtained, from either the measurement or the recovery, the probability of each entry to be sampled for the next round is calculated based on leverage scores of the matrix. The entries are then sequenced according to their probability in a descending order and the first C entries are picked up to be measured. Based on the new sampling set Ω' with $m + C$ samples, we get

the second recovered complete matrix $\hat{X}(1)$. This sampling procedure keeps running until the stopping criterion is reached. Finally, the latest completed matrix $\hat{X}(\mathrm{k})$ is outputted. A pseudocode of the whole sampling algorithm is given in **Table 2**.

During the procedure, five function blocks (red parts in brackets in **Fig. 1**) are needed:

• *UniformSampling*. For an $n \times n$ matrix, we usually need $\beta n^2 (0 \le \beta \le 1)$ measurements at start, which can be sampled uniformly at random. These measurements are then used to reconstruct the first complete matrix $\hat{X}(0)$ for the sampling procedure to proceed. For a practical implementation, we need *UniformSampling* to provide two kinds of outputs: a sampling set Ω of indexes of matrix entries and a sparse vector corresponding to entry values. Compared to the whole n^2 elements, βn^2 refers to a small fraction. Therefore, it would save much more memory space and processing time to operate on a sparse structure than the whole matrix.

• *MatrixRecovery*. A recovery algorithm is used to solve Problem (2). Every time the sampling set Ω is updated, the recovery algorithm needs to run once. It is better to have the output of *MatrixRecovery* in the form of rank-r SVD where $\hat{X} = U(k)\Sigma(k)V^T(k)$ which is convenient for the following step to calculate sample probability.

• *ProbCal*. Except for uniform sampling as initialization, we decide which entries to sample based on their probability. We calculate the probability based on leverage scores by (9).

• *OmegaUpdate*. Once the probability is ready, we rank the corresponding entries in descending order and select the first

$$C = 2nln(2n)\frac{length(P_{ij} > \gamma)}{n^2} \tag{16}$$

entries with the largest probability. γ is a probability threshold, e.g., $\gamma = 0.05$. The output requirement for *UniformSampling* also applies to *OmegaUpdate*.

Table 2. The proposed adaptive sampling algorithm

Input: Matrix dimension n, initial pair-nodes fraction $\beta \in [0,1]$, probability threshold γ;

$k = 0$;
$[\Omega_k, SparseVector(k)] = UniformSampling(n, \beta)$;
$\hat{X}(k) = MatrixRecovery(\Omega_k, SparseVector(k))$;
while(1)
    $k = k + 1$;
    $p_{ij}(k) = ProbCal(\hat{X}(k-1), |\Omega|_{k-1})$;
    $[\Omega_k, SparseVector(k)] = OmegaUpdate(p_{ij}(k), \gamma)$;
    $\hat{X}(k) = MatrixRecovery(\Omega_k, SparseVector(k))$;
    if $\left(StopCriterion(\hat{X}(k-1), \hat{X}(k))\right)$
        break;
    end
end

Output: $\hat{X}(k)$

• *StopCriterion*. The whole sampling procedure completes after the k^{th} sampling epoch when the stopping criterion is reached. This is basically a binary function, indicating that whether (10) holds true or not.

5. Performance Evaluations

5.1 Methodology

In simulation process, we adopt Harvard-226 dataset to evaluate the performance of our proposed algorithm. The dataset includes application-level RTTs between 226 clients that were collected from a peer-to-peer file-sharing application, namely Azureus [24]. The dataset used in the evaluation contains 2492546 dynamic and passive RTT measurements with timestamps in 4 hours. Among all 50850 ($=226^2-226$, we do not consider diagonal elements of the matrix which corresponds to the delay from one node to itself) possible pair-node paths, 1991 paths were not measured, accounting for 3.9% of the total paths as shown in **Fig. 2B**. The other paths were measured unevenly, from only once to 662 times. It can be seen in **Fig. 2B** that most of the measured paths locate in the range of 40 to 56 times (accounting for 6.0% to 99%). Medians of the RTTs measured multiple times are therefore used as the true measurements for performance evaluation. The RTT distribution of the resulted true measurements are shown in **Fig. 2C**. The maximum RTT is 704ms and over 90% of the all paths experience delay less than 320ms. The low-rank characteristic of the true-measurement matrix has been analyzed and shown in **Fig. 2A**. Moreover, to better show the effectiveness of our proposed algorithm, we also use the Abilene dataset [33], which contains a time series of start-end pairs traffic collected from the Internet2 backbone network based on a 5-minute sample period. Abilene dataset includes 24 weeks of traffic records, and weekly data is represented as a 121 × 2016 (7 × 24 × 12 = 2016) traffic matrix. We consider the traffic matrices for the first two weeks.

We compare the proposed sampling algorithm with uniform sampling, the information-based sampling proposed in [13] and Stochastic Proximal Gradient Descent-based (SPGD) in [32]. In ref. (13), a continuous and information-based adaptive sampling scheme is proposed. In the proposed scheme, measurements are carried out regularly. In each period, only a subset of end-to-end paths are measured, and the measurement data based on matrix completion can obtain the complete path information of the entire network. It should be noted that the proposed adaptive sampling strategy is based only on the information of the measurements results, which is closely related to the method proposed in this paper. In [32], to deal with large-scale network problems, the authors propose a scalable and memory-efficient sampling algorithm using stochastic proximal gradient descent (SPGD) method. The codes of sampling algorithms are conducted on MATLAB R2017a, with an Intel Xeon E5-2680 v4 CPU at 2.40 GHz.

5.2 Study on Input Parameters

In **Table 2**, in addition to the matrix dimension n, there are other two input parameters: the initial pair-nodes fraction β and the probability threshold γ. The former one determines the number of initial measurements from uniform sampling while the latter one decides on the number of updated measurements to be sampled for each epoch. In this subsection, their impact on the algorithm performance is evaluated.

Specifically, we perform simulations by combining different choices of β and γ illustrated in **Fig. 3**, including the median absolute errors (MAEs) and convergence stress (CS). The CS is calculated as follows:

$$\sqrt{\frac{\Sigma_{(i,j)\notin\Omega}(X_{ij}-\hat{X}_{ij})^2}{\Sigma_{(i,j)\notin\Omega}(X_{ij})^2}} \tag{17}$$

Here, MAE is used to minimize the effect of outliers and CS is used to assess the convergence by measuring the overall fitness. Note that we only measure errors on missing entries with index $(i, j) \notin \Omega$. Among all the missing entries, we also eliminate the diagonal elements and those are measured in the dataset. β is chosen from 2.5% to 30.0% in a step size of 2.5% and γ is from 0.0 to 0.4 with a step size equal to 0.02. Since the gaps between the maximum and minimum amplitudes of the metrics are quite large, the range of β is split into two parts: 2.5% to 15.0% in the 1^{st} row and 17.5% to 30.0% in the 2^{nd} row in **Fig. 3.**

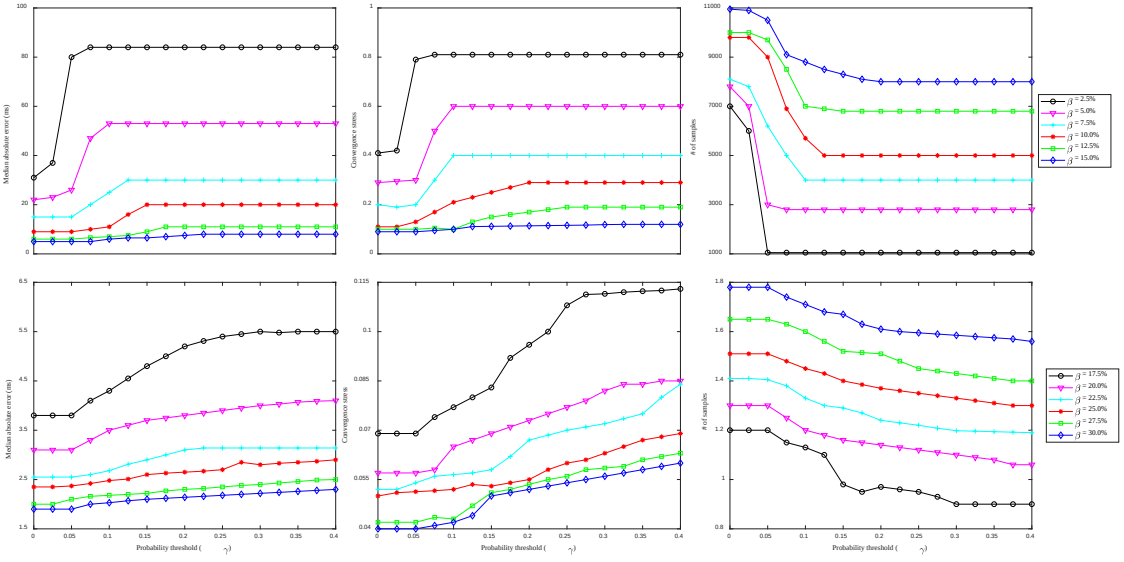


Fig. 3. Evaluation on input parameters β and γ: (A) median absolute error; (B) convergence stress and (C) final number of samples required to complete (The 1^{st} row corresponds to β of 2.5% to 15.0% while the 2^{nd} row refer to β of 17.5% to 30%, both in a step size of 2.5%).

As shown in **Fig. 3A**, under a specific γ, the median absolute error decreases with the increasing of β. A similar pattern can be seen for the convergence stress (**Fig. 3B**). This is because that a larger number of initial measurements also lead to more final measurements (**Fig. 3C**). By sampling more times, we obtain more information about the unknown matrix and therefore the matrix can be better recovered. The balance between the measurement cost and tolerable error is the core content of our works.

From **Fig. 3**, we can observe that for a fixed specific β, increasing γ causes the MAE (**Fig. 3A**) and the stress (**Fig. 3B**) to exacerbate, due to the reduced total final samples (**Fig. 3C**). Especially in the 1^{st} row in **Fig. 3** with relatively small β, when γ increases to a certain level, we can clearly see that the MAE, the stress, and the final number of samples are constants.

The reason behind this "saturation" phenomenon is that the number of entries which fulfill the requirement in updating rule (15) decreases with the increasing of γ. It can be seen in **Fig. 4** that the number of epoch for the sampling procedure to complete finally converges to 2. This can be further analyzed by **Fig. 5** where the additional number of samples to uniform sampling is 0, when the epoch number converges. This means that with an increasing threshold on sampling probability, we are shorter of suitable candidates to select, leading the sampling procedure to stop at early stage (i.e., directly after initial uniform sampling). Nevertheless, increasing β will delay the occurrence of the saturation. From both **Fig. 4** and **Fig. 5**, it is observed that the value of γ corresponding to the convergence value becomes larger with the increasing of β (e.g., γ =0.08, 0.10, 0.14, 0.20, 0.26, 0.32 for β = 2.5%, 5.0%, 7.5%, 10.0%), 12.5%, 15.0% respectively). For the 2^{nd} row in **Fig. 3C**, we can see the trend for the number of final samples to converge at a certain $\gamma > 0.4$.

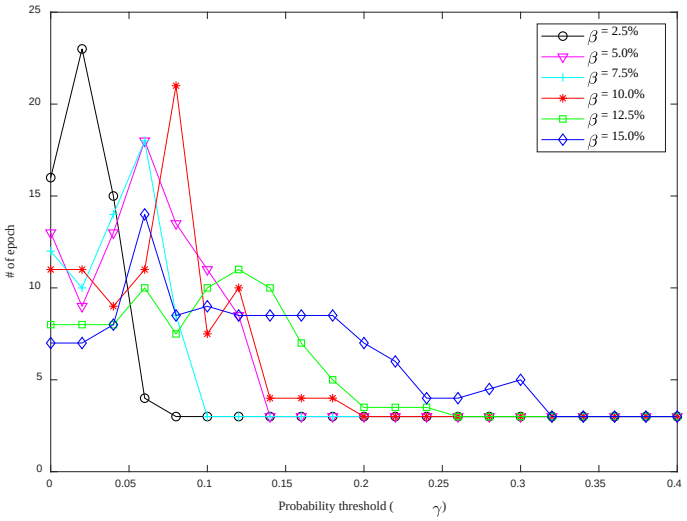


Fig. 4. Number of epoch for the proposed sampling algorithm to complete.

Therefore, we can summarize the input parameters in this subsection:

• Under a specific probability threshold γ, increasing initial pair-nodes fraction β improves the performance (smaller error and less convergence pressure) but at a higher measurement cost (higher number of final samples).

• With a fixed β, increasing γ exacerbates the performance (larger error and more convergence stress) but at a lower measurement cost (lower number of final samples). The performance saturates until a γ which makes length $(P_{ij} > \gamma) = 0$ is reached.

• Increasing β delays the occurrence of the saturation phenomenon with larger value of γ.

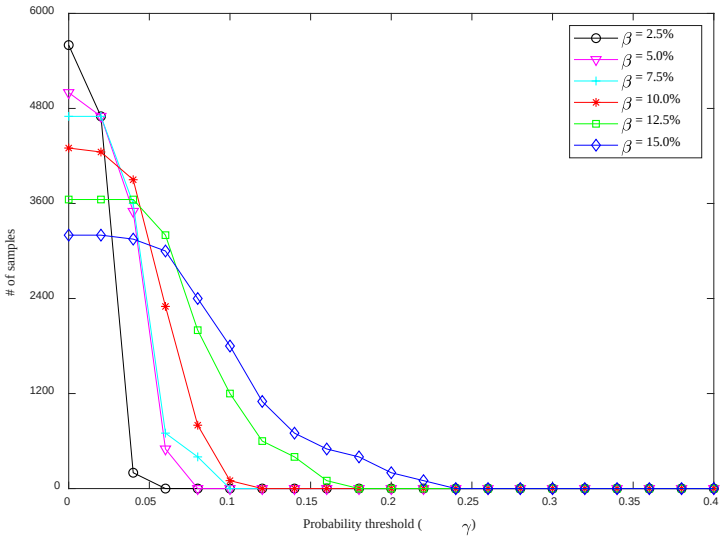


Fig. 5. Number of remained samples obtained by final samples subtracting initial uniformly sampled ones.

5.3 Comparison with Other Sampling Algorithms

In this subsection, we compare our proposed adaptive sampling algorithm with other methods: a traditional static uniform sampling, an adaptive information-based sampling devised in [13] and Stochastic Proximal Gradient Descent-based (SPGD) in [32]. Based on the analysis in Section 5.2, in order to balance between sampling performance and measurement cost, we choose β = 17.5% and γ = 0.05.

Table 3. Absolute error of different sampling algorithm (CDF=80%) on Harvard-226.

Algorithm	Absolute error (ms)
1^{st} epoch (uniform)	≤ 19.57
2^{nd} epoch (uniform)	≤ 17.01
Info-based sampling	≤ 15.58
Static uniform sampling	≤ 12.71
SPGD	≤ 13.59
Proposed adaptive sampling	≤ 12.05

Table 4. Absolute error of different sampling algorithm (CDF=80%) on Abilene.

Algorithm	Absolute error (ms)
1^{st} epoch (uniform)	≤ 16.69
2^{nd} epoch (uniform)	≤ 15.58
Info-based sampling	≤ 13.26
Static uniform sampling	≤ 14.59
SPGD	≤ 12.86
Proposed adaptive sampling	≤ 11.16

The absolute error of different sampling algorithm (CDF=80%) on Harvard-226 and Abilene is shown in **Table 3** and **Table 4** respectively. For example, on Harvard-226, the absolute

error of 1^{st} epoch is less than 19.57ms, and the value of the same metric for the proposed sampling algorithm is reduced to only 12.05ms (See **Table 3**), which is also the best among all the algorithms. The info-based sampling [13] relies on the distance between $\hat{X}(k)$ and $\hat{X}(k-1)$ to sample entries for the next epoch. Therefore, this algorithm needs two rounds of uniform sampling at the initialization stage. For a fair comparison, the results on the 1^{st} epoch of sampling are used for the info-based, SPGD and the proposed algorithms as initial data. Moreover, the info-based algorithm needs an additional initialization of the 2^{nd} epoch uniform sampling. The info-based sampling also improves its uniform initialization. The static uniform sampling is a one-epoch work, and the number of samples is the same as the proposed algorithm. Due to the lack of guidance on the number of samples, this algorithm is useless in practical applications. We use it to compare with our algorithm. As illustrated in **Table 3** and **Table 4**, the performance of the proposed algorithm is better than the information-based sampling and similar to the static uniform sampling.

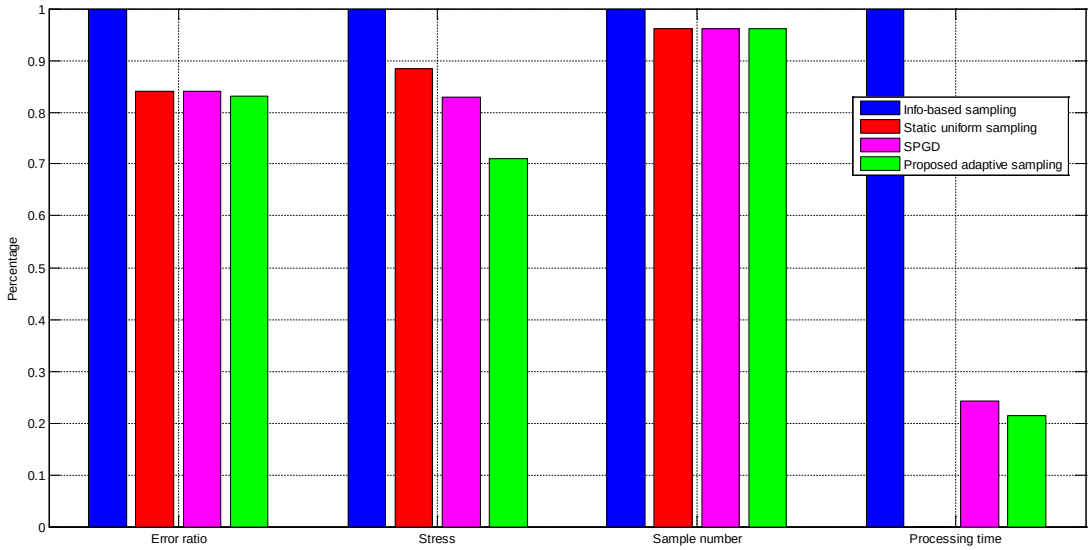


Fig. 6. Performance comparison on different sampling algorithms on Harvard-226. The proposed algorithm improves the baseline on error ratio, stress, number of sample, and processing time by 16.9%, 28.9%, 3.9%, 78.6% respectively (regardless of the processing time of static uniform sampling) on Harvard-226.

Moreover, these sampling algorithms can be further estimated based on error ratio, stress, number of sample, and processing time on Harvard-226 and Abilene in **Fig. 6** and **Fig. 7** respectively. We take normalized mean absolute error (NMAE) as the error ratio, and it is calculated as follows:

$$\frac{\Sigma_{(i,j)\notin\Omega} \mid X_{ij} - \hat{X}_{ij} \mid}{\Sigma_{(i,j)\notin\Omega} \mid X_{ij} \mid} \tag{18}$$

The stress is the same one as used in Section 5.2. The number of samples refers to the final number of samples that the sampling algorithm requires to complete. The processing time is the time that the sampling algorithm takes from the initialization to reaching the stopping condition. Because uniform sampling is static one-epoch work, we do not consider its processing time. The last two metrics are used to evaluate the measurement cost and are very

important for practical applications such as network delay measurement. The information-based sampling is used as the baseline and values in parentheses (**Table 5** and **Table 6**) represent the percentage of improvement.

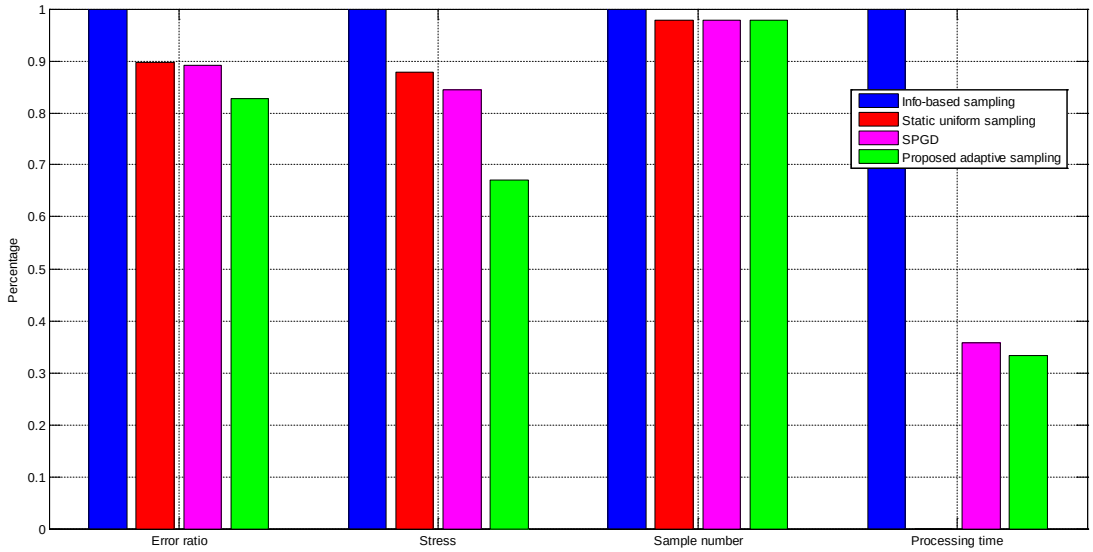


Fig. 7. Performance comparison on different sampling algorithms on Abilene. The proposed algorithm improves the baseline on error ratio, stress, number of sample, and processing time by 17.2%, 33.0%, 2.2%, 76.7% respectively (regardless of the processing time of static uniform sampling) on Abilene.

On **Table 5** and **Table 6**, we can observe that our proposed method achieves superior performance on error ratio, stress, number of sample, and processing time. For example, on Harvard-226, we can find that the proposed sampling algorithm improves the baseline on the error by 16.9%, similar to the uniform sampling (16.0%). For the stress, the proposed method improves the baseline by 28.9%. It is superior to the uniform one that "only" increases by 11.6%. This means that the proposed sampling algorithm is more suitable for real overall measurements than the other two algorithms. Compared with the information-based method and SPGD, the most important advantage of the proposed method is that it provides better performance (i.e., less error and less convergence pressure,) but with fewer samples (improves by 3.9%) in less processing time(improves by 78.6%).

The most important advantage of the proposed one over the information-based method and SPGD is that it provides better performance (i.e., smaller error and less convergence stress) but with fewer samples (improves by 3.9%) in shorter processing time (improves by 78.6%). The reason is that the distance between $\hat{X}(k-1)$ and $\hat{X}(k)$, which the info-based sampling and SPGD adopt as the criterion for selecting samples, limits the choices of suitable sampling candidates, making it difficult for the info-based sampling and SPGD to fulfill a rigorous requirement by the stopping criterion in (10) (e.g., $\varepsilon = 10^{-3}$).

Table 5. Specific values of metrics in **Fig. 6** on Harvard-226.

Algorithm	Information-based	Static uniform	SPGD	Proposed
Error ratio	0.0664	0.0558(16.0%)	0.0554(16.0%)	0.0552(16.9%)
Stress	0.0955	0.0844(11.6%)	0.0793(17.0%)	0.0679(28.9%)
Sample number	12284	11809(3.9%)	11809(3.9%)	11809(3.9%)
Processing time	1702s	-	414s(75.7%)	364s(78.6%)

Table 6. Specific values of metrics in **Fig. 7** on Abilene.

Algorithm	Information-based	Static uniform	SPGD	Proposed
Error ratio	0.0593	0.0532(10.2%)	0.529(10.8%)	0.0491 (17.2%)
Stress	0.0912	0.0801(12.2%)	0.0771(15.5%)	0.0611(33.0%)
Sample number	15762	12469(2.1%)	12349(2.2%)	12334(2.2%)
Processing time	2132s	-	551(74.2%)	496s(76.7%)

6. Conclusion and Future Work

In this paper, we propose an online adaptive sampling algorithm to measure network delay based on matrix completion. The proposed algorithm uses statistical leverage scores to sample potential elements to obtain better matrix estimation from limited measurements and uses a stopping condition to reduce the measurement cost. The most important advantage of the proposed algorithm is to fill in the gap between the theoretical bound and the practical implementation on the amount of samples needed to successfully restore a matrix. In addition, the proposed algorithm has flexibility in choosing a specific restoration algorithm for matrix completion. Simulation results based on a real-world network delay matrix indicate that the proposed algorithm can provide better performance (smaller error and less convergence pressure) at a lower cost (fewer samples and shorter processing time) than the compared algorithms.

Acknowledgements

This paper is supported by the Science and Technology Innovation Development Project in Jilin City (Grant No. 20190302008).

References

[1] Z. Zhou, H. Yu, L. Xu, Y. Zhang, S. Mumtaz and J. Rodriguez, "Dependable content distribution in D2D-based cooperative vehicular networks: A big data-integrated coalition game approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 953-964, 2018. Article (CrossRef Link)

[2] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geeis, R. Gummadi, S. Rhea, H. Weatherspoon and W. Weimer, "Oceanstore: An architecture for global-scale persistent storage," *ACM SIGPLAN Notices*, vol. 28, no. 5, pp. 190-201, 2000. Article (CrossRef Link)

[3] M.B. Shareh, H. Navidi H, H.H.S. Javadi and M. Hosseinzadeh, "Preventing Sybil attacks in P2P file sharing networks based on the evolutionary game model," *Information Sciences*, vol. 470, pp. 94-108, 2019. Article (CrossRef Link)

[4] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing*, pp. 329-350, 2001. Article (CrossRef Link)

[5] S. Vaton, O. Brun, M. Mouchet, P. Belzarena, I. Amigo, B.J. Prabhu and T. Chonavel, "Joint minimization of monitoring cost and delay in overlay networks: optimal policies with a Markovian approach," *Journal of Network and Systems Management*, vol. 27, no. 1, pp. 188-232, 2019. Article (CrossRef Link)

[6] M. Hua, Y. Wang, Z. Zhang, C. Li. Y. Huang and L. Yang, "Power-efficient communication in UAV-aided wireless sensor networks," *IEEE Communications Letters*, vol. 22, no. 6, pp. 1264-1267, 2018. Article (CrossRef Link)

[7] J. Simonjan and B. Rinner, "Decentralized and resource-efficient self-calibration of visual sensor networks," *Ad Hoc Networks*, vol. 88, pp. 112-128, 2019. Article (CrossRef Link)

[8] Q. He, X. Wang, Z. Lei, M. Huang, Y. Cai and L. Ma, "TIFIM: A two-stage iterative framework for influence maximization in social networks," *Applied Mathematics and Computation*, vol. 354, pp. 338-352, 2019. Article (CrossRef Link)
[9] C. Fan, B. Raichel and G.V. Buskirk, "Metric violation distance: hardness and approximation," in *Proc. of Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 196-209, 2018. Article (CrossRef Link)
[10] S. Lee, Z. Zhang, S. Sahu and D. Saha, "On suitability of Euclidean embedding of internet hosts," in *Proc. of Joint International Conference on Measurement & Modeling of Computer Systems*, pp. 157-168, 2006. Article (CrossRef Link)
[11] Y. Chi, Y. Lu and Y. Chen, "Nonconvex optimization meets low-rank matrix factorization: An overview," *IEEE Transactions on Signal Processing*, vol. 67, no. 20, pp. 5239-5269, 2019. Article (CrossRef Link)
[12] Y. Liao, W. Du, P. Geurts and G. Leduc, "DMFSGD: a decentralized matrix factorization algorithm for network distance prediction," *IEEE/ACM Transactions on Networking*, vol. 21, no. 5, pp. 1511-1524, 2013. Article (CrossRef Link)
[13] K. Xie, L. Wang, G. Wang, G. Xie, G. Zhang, D. Xie and J. Wen, "Sequential and adaptive sampling for matrix completion in network monitoring systems," in *Proc. of IEEE INFOCOM*, pp. 2443-2451, 2015. Article (CrossRef Link)
[14] R. Zhu, B. Liu, D. Niu, Z. Li and H.V. Zhao, "Network latency estimation for personal devices: a matrix completion approach," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, pp. 1-14, 2016. Article (CrossRef Link)
[15] E.J. Candès and B. Recht, "Exact matrix completion via convex optimization," *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717-719, 2009. Article (CrossRef Link)
[16] E.J. Candes and T. Tao, "The power of convex relaxation: near-optimal matrix completion," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2053-2080, 2010. Article (CrossRef Link)
[17] J. Beauquier, J. Burman, S. Kutten, T. Nowak and C. Xu, "Data collection in population protocols with non-uniformly random scheduler," in *Proc. of International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics*, pp. 13-25, 2017. Article (CrossRef Link)
[18] B. Recht, "A simpler approach to matrix completion," *Journal of Machine Learning Research*, vol. 12, no. 4, pp. 3413-3430, 2009.
[19] Y. Chen, S. Bhojanapalli, S. Sanghavi and R. Ward, "Coherent matrix completion," in *Proc. of International Conference on International Conference on Machine Learning*, pp. 674-682, 2019. Article (CrossRef Link)
[20] E. Pauwels, F. Bach and J.P. Vert, "Relating leverage scores and density using regularized christoffel functions," in *Proc. of Advances in Neural Information Processing Systems*, pp. 1663-1672, 2018. Article (CrossRef Link)
[21] P. Drineas, M. Magdon-Ismail, M.W. Mahoney and D. Woodruff, "Fast approximation of matrix coherence and statistical leverage," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 3475-3506, 2011.
[22] J. Cai, E.J. Candès and Z. Shen, "A singular value thresholding algorithm for matrix completion," *Siam Journal on Optimization*, vol. 20, no. 4, pp. 1956-1982, 2008. Article (CrossRef Link)
[23] J. Prateek, N. Praneeth and S. Sujay, "Low-rank matrix completion using alternating minimization," in *Proc. of ACM Symposium on Theory of Computing*, pp. 1-7, 2012. Article (CrossRef Link)
[24] J. Ledlie, P. Gardner and M.I. Seltzer, "Network coordinates in the wild," in *Proc. of USENIX Conference on Networked Systems Design & Implementation*, pp. 299-312, 2007. Article (CrossRef Link)
[25] L. Tang and M. Crovella, "Virtual landmarks for the internet," in *Proc. of ACM SIGCOMM conference on Internet measurement*, pp. 143-152, 2003. Article (CrossRef Link)

[26] Y. Chen, S. Bhojanapalli, S. Sanghavi and R. Ward, "Completing any low-rank matrix, provably," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 2999-3034, 2013. Article (CrossRef Link)
[27] A. Eftekhari, M.B. Wakin and R.A. Ward, "MC^2: a two-phase algorithm for leveraged matrix completion," *Information and Inference: A Journal of the IMA*, vol. 7, no. 3, pp. 581-604, 2018. Article (CrossRef Link)
[28] Z. Liu and L. Vandenberghe, "Interior-point method for nuclear norm approximation with application to system identification," *Siam Journal on Matrix Analysis & Applications*, vol. 31, no. 3, pp.1235-1256, 2009. Article (CrossRef Link)
[29] K. Mohan and M. Fazel, "Iterative reweighted least squares for matrix rank minimization," in *Proc. of Annual Allerton Conference on Communication, Control, and Computing*, pp. 1-6, 2010. Article (CrossRef Link)
[30] K. Salehin, R. Rojas-Cessa and K.W. Kwon, "COMPRESS: a self-sufficient scheme for measuring queueing delay on the Internet routers," in *Proc. of International Conference on Computing, Networking and Communications*, pp. 624-629, 2019. Article (CrossRef Link)
[31] Y. Ren, Y. Liu, N. Zhang, A. Liu, N. Xiong and Z. Cai, "Minimum-cost mobile crowdsourcing with QoS guarantee using matrix completion technique," *Pervasive and Mobile Computing*, vol. 49, pp. 23-44, 2018. Article (CrossRef Link)
[32] F. Xiao, L. Chen, H. Zhu and R. Hong, "Anomaly-tolerant network traffic estimation via noise-immune temporal matrix completion model," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1192-1204, 2019. Article (CrossRef Link)
[33] Abilene/Internet2. [Accessed: January. 23, 2020]. [Online]. Available: http://www.cs.utexas.edu/~yzhang/research/AbileneTM/.

Wei Meng received the B.S. in Computer and Practical Application from Changchun University of Science and Technology, P.R. China in 2002. She received her M.S. degree in Computer Technology from Northeast Electric Power University, P.R. China in 2010. Now, she works in School of Computer Science and Technology, Beihua University. Her research interests include Database technology, Computer network and Artificial intelligence.

Laichun Li received the B.S. in Computer Science and Technology from Kunming University of Science and Technology, P.R. China in 2002. He received the Master degree in Computer Technology from Changchun University of Technology, P.R. China in 2009. Now, he works in School of Computer Science and Technology, Beihua University. His research interests include Embedded system, Computer network and Artificial intelligence.