TNDP: Tensor-based Network Distance Prediction with Confidence Intervals

Haojun Huang, Li Li, Geyong Min, Wang Miao, Yingying Zhu, and Yangming Zhao

Abstract—The knowledge of network distances, in the form of delay or latency, for example, is beneficial to a number of distributed applications. Notice that it is difficult and expensive to implement global network measurements to obtain network distance, a feasible idea is to predict unknown distances by introducing network coordinates with limited network measurements. The existing solutions always represent the unknown network distances in a rather unique number. However, research and applications indicate that the real network distances are hard to be accurately figured out and changes subtly in an interval over time with the dynamic network environments. Accordingly, this paper proposes a tensor-based network distance prediction (TNDP) approach to represent network distance with confidence intervals, by exploiting the random distance tensor and distributed matrix factorization. With a small set of network measurements among the nodes selected randomly, a distance matrix tensor has been established and factorized into the product of two location matrixes with the adaptive SGD-based learning solution. By introducing the important training determinants, including weight matrix, regularization coefficient, and minibatch gradient descent with the exponential decay rates, the unknown distances among nodes can be accurately inferred in the forms of confidence intervals, with quick convergence and less overfitting. Extensive experimental simulations on a wide variety of available data sets demonstrate that TNDP is superior to other approaches in terms of accuracy for network distance prediction.

Index Terms—Network distance prediction, network measurements, network tensor, matrix factorization, training, confidence intervals.

✦

1 Introduction

IN current networks, there are a large number of distributed applications, such as content distribution networks (CDNs) [1], overlay networks [2], and peer-to-peer file sharing [3], that could provide the same many-to-one services for users. The users can benefit from choosing the applications that provide them services if the network distances to them are known in advance. For instance, a client would enjoy downloading the desired video from many candidate CDN servers that have the highest bandwidth to them subject to the known network distances. However, due to the high cost of network measurements and severe traffic congestion, it is infeasible to always actively probe end-to-end network distances among all the network entities [4], [5], [6]. Accordingly, it is urgent to design alternative approaches to understand the network distances for ever-incresing distributed applications.

Network distance prediction has been considered as a promising solution to fulfil this requirement [4], [7]. The main idea is to exploit the limited network distances among a small set of node-pairs, in the form of either one-way delay or more often Round-Trip Time (RTT), to predict unknown distances among other node-pairs, where direct measurements are not performed. Over the past years, a large number of approaches have been developed [8], [7], [9], [10], [11], [12], [13], [14], [15] to predict network distance by introducing network coordinates. These approaches can be fell under two categories: Euclidean embedding and Non-Euclidean embedding (see more details in Section 5). Note that Non-Euclidean embedding, for example, matrix factorization, can reconstruct the network distances more precisely than Euclidean embedding [8], [12], in this paper we mainly investigate the Non-Euclidean network distance prediction for different distributed applications. The current Non-Euclidean solutions focus on the prediction of the unknown network distances among nodes based on the known node locations. However, there are at least two limitations existing in these solutions.

Firstly, how to exploit the known locations of a set of nodes, for example, a few anchor nodes or ordinary nodes with learned locations, to infer the locations of other nodes in a distributed manner. Most current approaches such as [10], [12] are based on the centralized design concept, and thus are heavily dependent on the fixed anchor nodes to work. However, in reality, due to their random motion, node distribution, and network network congestion [16], etc, the network measurements always cannot be implemented for each ordinary node to obtain its location. It is prerequisite to design a distributed mechanism that can exploit alternative nodes with known locations as the anchor nodes for ordinary node localization.

Secondly, how to more accurately represent the unknown network distances. The network distance be-

H. Huang, L. Li and Y. Zhu are with School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China. Email: {hjhuang,lil4,yyzhu}@hust.edu.cn.
G. Min and W. Miao are with Department of Computer Science, University of Exeter, Exeter, EX4 4QF, UK. Email: {g.min, Wang.Miao}@exeter.ac.uk.
Y. Zhao is with Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY 14260 2500, USA. Email: yangming@buffalo.edu.
Manuscript received May 31, 2021; revised March. 10, 2021 (Corresponding author: Yingying Zhu).

TABLE 1: Comparison of solutions for addressing the existing challenges in network distance prediction

Existing solutions		**Challenge 1**	**Challenge 2**
		Centralization	Distance
Euclidean	GNP	Yes	Uniqueness
	PIC	Yes	Uniqueness
Non-Euclidean	HYP	No	Uniqueness
	IDES	Yes	Uniqueness
	DMF	No	Uniqueness
	DMFSGD	No	Uniqueness
	RMF	No	Uniqueness
	TNDP	No	Confidence interval

tween two nodes in current solutions commonly is computed as an unique number, derived from the locations of nodes in Euclidean and non-Euclidean space. However, relevant studies [8], [7], [17], [5] show that the network distance relies on multiple factors like delay, bandwidth and data flow, and furthermore, there are often multiple paths between node pairs. One uniqueness can no longer represent the real network distances. Fig. 1 illustrates a concrete example, in which network distance $d(i,j)$ between nodes i and j can be computed as the length of paths 1, 2 or 3, *i.e.*, $d(i,j)$=$d(i,a_1)$+$d(a_1,a_2)$+$d(a_2,j)$, or $d(i,j)$=$d(i,b_1)$+$d(b_1,j)$, or $d(i,j)$=$d(i,c_1)$+$d(c_1,j)$. With different path states, the traffic would be carried by one of these three paths, and hence, the users will experience various network distances. This means that the network distances among reachable nodes cannot be represented by an unique value only, but confidence intervals are more appropriate.

In order to address above issues, in this paper, we propose a tensor-based network distance prediction (TNDP) approach with confidence intervals, by introducing the random distance tensor and distributed coordinate fitting with the adaptive stochastic gradient descent (SGD) based learning. The unknown network distance between two reachable nodes in our solution is represented as a confidence interval rather than an unique number. Thus, TNDP can work in different distributed occasions to fulfill the individual requirements of users, with the help of the randomly selected reference nodes with known locations. The comparisons between TNDP and related approaches to address existing challenges are illustrated in TABLE 1. Simulation results show that TNDP provides better performance than the previous approaches in the accuracy of distance prediction.

In this paper, we make the distinct contributions as follows.

- With a small set of network measurements among randomly selected nodes, a distance matrix tensor has been established to provide enough data samples to infer unknown network distances, without additional measurement overhead.
- A novel adaptive SGD-based learning solution for matrix factorization, by introducing the important training determinants, including weight matrix, regularization coefficient, and minibatch gradient descent with the exponential decay rates, is proposed to quickly factorize each network matrix, included in distance tensor, to derive the locations of the unknown network distances. It is adaptive, distributed, and feasible with less error, less overfitting and quick convergence, enabling to obtain the unknown network distances more efficiently than traditional approaches.
- The unknown network distances have been presented in the form of confidence intervals, thereby can figure out the real network distances as far as possible. To the best of our knowledge, this solution is the first of its kind proposed in the literature and takes prospective insights to predict the network distances.
- Extensive simulations are conducted in various scenarios, referring to data sets, neighbor nodes, network dimensions, learning rate scenario, and regularization coefficient, to evaluate the performance of TNDP. Related results show that TNDP outperforms the previous approaches in the accuracy of distance prediction.

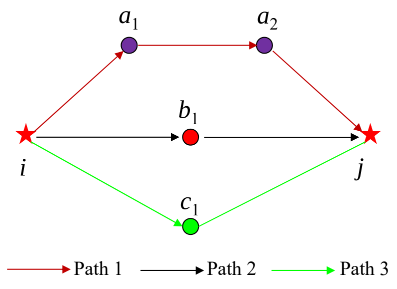
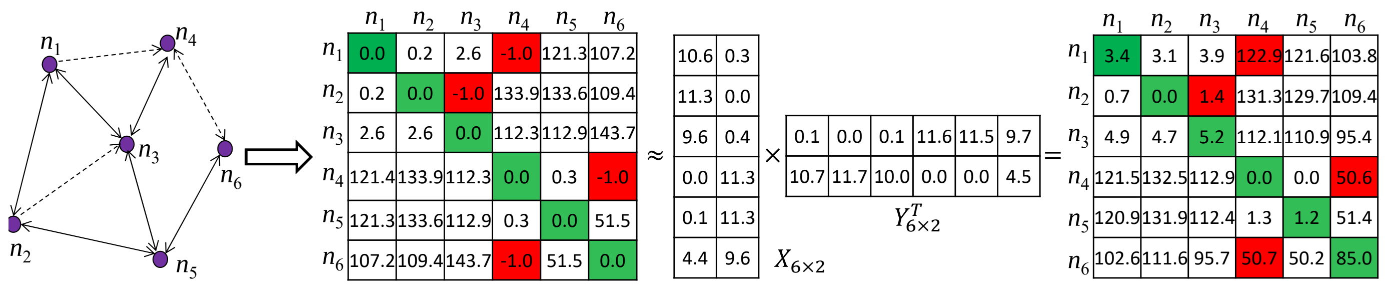


Fig. 1: The various network distances between two reachable nodes along multiple paths.

The rest of the paper is organized as follows. Section 2 presents the critical basic knowledge, including the distance matrix factorization, distance tensor, and problem formulations with confidence intervals, to help understand TNDP. Section 3 describes the details of TNDP. Section 4 presents and analyses the results of extensive simulations to validate the performance of TNDP in network distance prediction. Section 5 describes the related work. Finally, Section 6 concludes this paper.

2 Background Knowledge

This section describes the important knowledge, which is essential to understand TNDP. We first present the distributed distance matrix factorization, and then elaborate the distance tensor for network distance expression. Finally, the problem formulation on network distance prediction is given. To improve the readability, the main parameters and notations used in this paper are listed in TABLE 2.


n1 n2 n3 n4 n5 n6
n1 0.0 0.2 2.6 -1.0 121.3 107.2
n2 0.2 0.0 -1.0 133.9 133.6 109.4
n3 2.6 2.6 0.0 112.3 112.9 143.7
n4 121.4 133.9 112.3 0.0 0.3 -1.0
n5 121.3 133.6 112.9 0.3 0.0 51.5
n6 107.2 109.4 143.7 -1.0 51.5 0.0
≈
10.6 0.3
11.3 0.0
9.6 0.4
0.0 11.3
0.1 11.3
4.4 9.6
X6×2
×
0.1 0.0 0.1 11.6 11.5 9.7
10.7 11.7 10.0 0.0 0.0 4.5
Y^T 6×2
=
n1 n2 n3 n4 n5 n6
n1 3.4 3.1 3.9 122.9 121.6 103.8
n2 0.7 0.0 1.4 131.3 129.7 109.4
n3 4.9 4.7 5.2 112.1 110.9 95.4
n4 121.5 132.5 112.9 0.0 0.0 50.6
n5 120.9 131.9 112.4 1.3 1.2 51.4
n6 102.6 111.6 95.7 50.7 50.2 85.0

(a) Distance measurement (b) 6×6 distance matrix D (c) Matrix factorization (d) 6×6 factorized matrix $\widehat{D}$

Fig. 2: Matrix-factorization-based network distance prediction among 6 nodes, illustrated in (a), with missing distances in the form of delay. The missing distances d_{14}, d_{23}, d_{46} and d_{64} are represented with the length of -1.0 in (b) for facilitating matrix factorization, and given by in (d) through DMFSGD-based matrix factorization in (c) over data set PlanetLab490 [18].

TABLE 2: The important parameters and notations

Symbol	Meaning
$d_{ij}/\widehat{d_{ij}}$	The measured/estimated distance from nodes i to j
D	$n \times n$ distance matrix formed by network distances among n nodes
$\mathbb{R}^{D_1..D_k}$	Distance tensor including k matrix D_1,...,D_k
X	$n \times d$ matrix storing d basis vectors of D
Y	$n \times d$ matrix storing d linear coefficients of D
x_i	The d-dimensional incoming vector of node i in X
y_i	The d-dimensional outgoing vector of node i in Y
(x_i, y_i)	The location of node i derived from distance matrix factorization
$\bar{d_{ij}}/\sigma_{ij}^2$	Mean value and variance of k predicted distances
β_1/β_2	Two exponential decay rates w.r.t. the gradient and the squared gradient

2.1 Distributed Distance Matrix Factorization

It is considered that the measurement network distances d_{ij} ($1 \leq i, j \leq n$) among n nodes selected randomly can be constructed an $n \times n$ distance matrix D, where a small number of elements are unknown. In order to recover the missing elements, we can factorize D into the product of two $n \times m$ ($m \ll n$) matrixes X and Y [19], [20], *i.e.*,

$$\begin{aligned} D &= \begin{bmatrix} d_{11} & \dots & d_{1n} \\ \dots & \dots & \dots \\ d_{n1} & \dots & d_{nn} \end{bmatrix} \\ &\approx \hat{D} = XY^T \\ &= \begin{bmatrix} x_{11} & \dots & x_{1m} \\ \dots & \dots & \dots \\ x_{n1} & \dots & x_{nm} \end{bmatrix} \begin{bmatrix} y_{11} & \dots & y_{1m} \\ \dots & \dots & \dots \\ y_{n1} & \dots & y_{nm} \end{bmatrix}^T. \end{aligned} \tag{1}$$

Fig. 2 illustrates such a common matrix-factorization-based network distance prediction among 6 nodes with missing distances. Denote $x_i = [x_{i1}, ..., x_{im}]$ and $y_i = [y_{i1}, ..., y_{im}]$. Let x_i and y_i stand for the incoming vector and outgoing vector of node i, respectively. Built on Eq. (1), the unknown or known distance d_{ij} between two arbitrary reachable nodes i and j can be estimated as

$$d_{ij} \approx \hat{d}_{ij} = x_i y_j^T = \sum_{k=1}^{m} x_{ik} y_{jk}. \tag{2}$$

Generally speaking, this can be achieved by minimizing

$$F(D, X, Y, W) = \sum_{i,j=1}^{n} w_{ij} f(d_{ij}, x_i y_j^T), \tag{3}$$

where W represents a weight matrix with w_{ij}= 1 when d_{ij} is measured and 0 otherwise, and $f(d_{ij}, \hat{d}_{ij})$, the square loss function, is given as follows:

$$f(d_{ij}, \hat{d}_{ij}) = (d_{ij} - \hat{d}_{ij})^2. \tag{4}$$

There are a number of existing optimization methods like Newton algorithms or Gradient Descent, which can be used to achieve this goal along the negative gradient direction of $f(d_{ij}, \hat{d}_{ij})$ [21], [22] while will result in large errors on the unseen data used in learning due to overfitting [23]. The absolute errors of distance prediction in Fig. 2 have up to [0.1, 48.3]ms. An efficient solution that addresses this issue is to regularize the previous norms of X and Y, by introducing the following regularization coefficient:

$$\begin{aligned} F(D, X, Y, W, \lambda) = &\sum_{i,j=1}^{n} w_{ij} f(d_{ij}, x_i y_j^T) \\ &+ \lambda(\sum_{i=1}^{n} x_i x_i^T + \sum_{i=1}^{n} y_i y_i^T). \end{aligned} \tag{5}$$

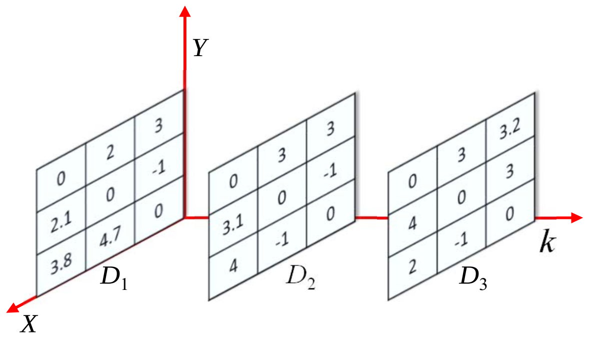
The additional term represents the regularization which controls the norm of X and Y to prevent overfitting, and λ is the regularization coefficient. We decompose Eq. (5) into two subproblems,

$$\begin{cases} f_{i,in} = \sum_{j=1}^{n} w_{ij} f(d_{ij}, x_i y_j^T) + \lambda x_i x_i^T, \\ f_{i,out} = \sum_{j=1}^{n} w_{ij} f(d_{ji}, x_j y_i^T) + \lambda y_i y_i^T. \end{cases} \tag{6}$$

By minimizing both $f_{i,in}$ and $f_{i,out}$, the locations, referring to the unknown distance d_{ij}, can be obtained at a sever, which is responsible for network distance prediction.

2.2 Distance Tensor for Network Distance Prediction

Matrix factorization can be used to predict the network distance, represented in an uniqueness [12], [8]. However, in reality, the network distance between two nodes always changes with time, depending on network flow, measurement strategies and node mobility, etc., and should be a series of variations. Based on this observation, we introduce distance tensor for all network distance estimations. In order to facilitate understanding, we first define the tensor as follows.



$$A = \left[\begin{array}{ccc|ccc|ccc} 0 & 2 & 3 & 0 & 3 & 3 & 0 & 3 & 3.2 \\ 2 & 0 & -1 & 3.1 & 0 & -1 & 4 & 0 & 3 \\ 3 & 4.7 & 0 & 4 & -1 & 0 & 2 & -1 & 0 \end{array}\right]$$

Fig. 3: A distance tensor $A \in \mathbb{R}^{3\times3\times3}$, where there exist 3 3×3 distance matrices and k=3.

***Definition** 1:* A tensor, also known as Nth-order/N-way tensor, multidimensional array, N-way or N-mode array, is defined as higher-order generalization of a vector and a matrix, which are referred to as the first-order and second-order tensors, respectively, and denoted by $A \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_N}$. Here, N, called way or mode, denotes the order of A. The element of A is denoted by $a_{i_1,i_2,\ldots,i_N}, i_n \in 1,2,\ldots,i_n, 1 \le n \le N$.

Fig. 3 illustrates an example of a distance tensor $A \in \mathbb{R}^{3\times3\times3}$. Built upon Eq. (1), d_{ij}, the missing entry in $n\times n$ distance matrices, can be derived through k distance matrix factorization. Let D_l ($1\le l\le k$) denote the i-th $n\times n$ distance matrix. Then, we have

$$d_{ij} \in \mathbb{R}^{D_1\times D_2\times\ldots\times D_k}. \tag{7}$$

Built upon Eq. (5), each D_l can be factorized into the form of $X_l Y_l^T$. Let $x_{l,i}$ and $y_{l,i}$ denote the l-th incoming vector and outgoing vector of node i, respectively, obtained from the matrix factorization of D_l. Then, there are k incoming vectors and outgoing vectors, *i.e.*, $x_{1,i}$, $x_{2,i}$, $\cdots$, $x_{k,i}$, and $y_{1,i}$, $y_{2,i}$, $\cdots$, $y_{k,i}$ for node i. Similarly, node j will obtain k incoming vectors and outgoing vectors, *i.e.*, $x_{1,j}$, $x_{2,j}$, $\cdots$, $x_{k,j}$, and $y_{1,j}$, $y_{2,j}$, $\cdots$, $y_{k,j}$. Let $[d_{ij}]_l$ denote the l-th predicted distance between nodes i and j, and $\bar{d}_{ij}$ stand for average distance of k different d_{ij}. Thus,

$$\bar{d}_{ij} = \frac{\sum_{l=1}^{k}[d_{ij}]_l}{k}. \tag{8}$$

Similar to [20], the distances d_{ij} and $\bar{d}_{ij}$ follow Gaussian distribution, and can be given as follows:

$$\frac{d_{ij} - \bar{d}_{ij}}{\sigma/\sqrt{k}} \sim N(0,1). \tag{9}$$

Let μ_{ij} and ν_{ij} denote the upper and lower bounds of the predicted distance between nodes i and j. Given a $(1-\eta)$ confidence level, we have

$$P(\mu_{ij} < d_{ij} < \nu_{ij}) = 1 - \eta. \tag{10}$$

Following Gaussian distribution, we have

$$P(|\frac{\bar{d}_{ij} - d_{ij}}{\sigma_{ij}/\sqrt{k}}| < u_{\eta/2}) = 1 - \eta, \tag{11}$$

where $u_{\eta/2}$ denotes the number that the probability is greater than or smaller than $\eta/2$. Then, μ_{ij} and ν_{ij} can be computed as follows:

$$\begin{cases} \mu_{ij} = \bar{d}_{ij} - u_{\eta/2}\sqrt{\frac{\sigma_{ij}^2}{k}}, \\ \nu_{ij} = \bar{d}_{ij} + u_{\eta/2}\sqrt{\frac{\sigma_{ij}^2}{k}}. \end{cases} \tag{12}$$

This means that the confidence interval of d_{ij} can be represented as $[\mu_{ij}, \nu_{ij}]$, obtain from Eq. (12).

2.3 Problem Formulation

Given the fore-mentioned network models and notations, we can formulate the network distance prediction for missing entry d_{ij} in D as the problem to find out a confidence interval through distributed and adaptive distance matrix factorization.

Specifically, node i, which intends to know the distance to node j, first launches k network measurements among more than $(n-1)$ nodes (including j) selected randomly. This means that an $n \times n \times k$ distance tensor, with k network matrices $D_1, \cdots, D_k$, has been established. Then, TNDP performs multiple matrix factorization with adaptive SGD-based approaches, by minimizing Eq.(5). After that, the incoming vector $x_{l,i}$ and outgoing vector $y_{l,j}$ of nodes i and j, respectively, in the l-th matrix factorization can be given in the form of

$$\begin{cases} x_{l,i} = \arg\min \sum_{j=1}^{n} w_j^i (D_{ij} - x_{l,i} y_{l,j}^T)^2, \\ y_{l,j} = \arg\min \sum_{i=1}^{n} w_i^j (D_{ji} - x_{l,j} y_{l,i}^T)^2. \end{cases} \tag{13}$$

Built on Eq. (2), it is easy to obtain the l-th predicted network distance d_{ij}. Thus, the average distance $\bar{d}_{ij}$ of

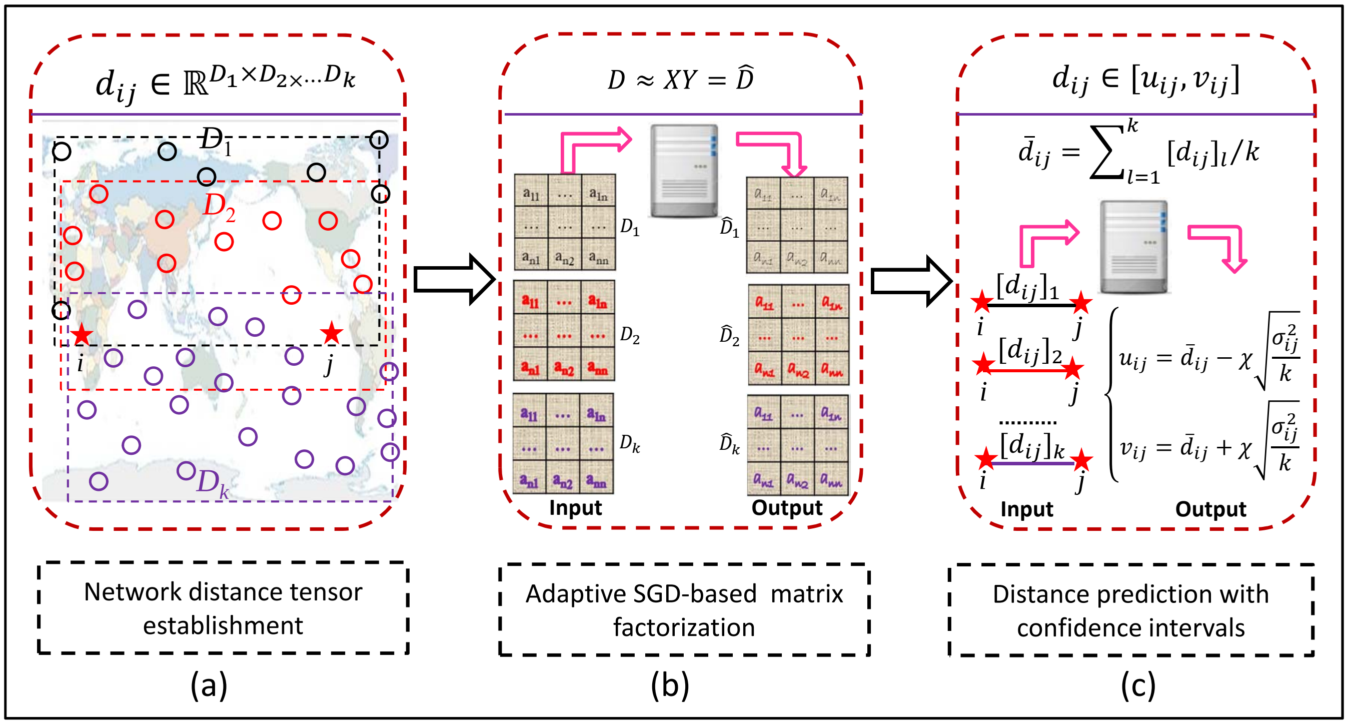

$d_{ij} \in \mathbb{R}^{D_1 \times D_2 \times \ldots D_k}$
D_1
D_2
D_k
i
j
$D \approx XY = \widehat{D}$
D_1
D_2
D_k
$\widehat{D}_1$
$\widehat{D}_2$
$\widehat{D}_k$
Input
Output
$d_{ij} \in [u_{ij}, v_{ij}]$
$\bar{d}_{ij} = \sum_{l=1}^{k} [d_{ij}]_l / k$
$[d_{ij}]_1$
$[d_{ij}]_2$
$[d_{ij}]_k$
$u_{ij} = \bar{d}_{ij} - \chi \sqrt{\frac{\sigma_{ij}^2}{k}}$
$v_{ij} = \bar{d}_{ij} + \chi \sqrt{\frac{\sigma_{ij}^2}{k}}$
Input
Output
Network distance tensor establishment
Adaptive SGD-based matrix factorization
Distance prediction with confidence intervals
(a)
(b)
(c)

Fig. 4: The architecture of TNDP, which consists of three steps: Distance tensor establishment, the adaptive SGD-based matrix factorization, and distance prediction with confidence intervals. The network measurement among more than n nodes have been implemented to establish an $n \times n \times k$ distance tensor $\mathbb{R}^{D_1 \times D_2 \times \ldots \times D_k}$ in (a), where network distance d_{ij} is missing. Each distance matrix D_l ($1 \leq l \leq k$) has been factorized into the product two $n \times m$ ($m \ll n$) matrixes X_l and Y_l through the adaptive training in (b). Finally, the missing distance d_{ij} has been derived from Eqs. (8) and (12) with a confidence interval $[\mu_{ij}, \nu_{ij}]$ in (c).

k distances between nodes i and j, *i.e.*, $[d_{ij}]_1$, $[d_{ij}]_2, \cdots$, $[d_{ij}]_k$ can be obtained, built on Eq. (6). Following Eq. (12), we can find out the upper and lower bounds, *i.e.*, μ_{ij} and ν_{ij}, of the predicted distance d_{ij} with the available measurement data, referring to nodes i, j, and more than $(n-2)$ other nodes randomly selected in the network. This means that unknown distance d_{ij} between nodes i and j can be represented the confidence interval $[\mu_{ij}, \nu_{ij}]$.

3 TNDP: Tensor-based Network Distance Prediction

This section mainly introduces TNDP in detail for network distance prediction with confidence intervals. First, the architecture of TNDP is provided, and then both the daptive SGD-based network distance prediction and its variant with confidence intervals are presented.

3.1 Architecture Overview

The main idea of TNDP is to predict the unknown network distances with confidence intervals by introducing the distance matrix tensor and adaptive SGD-based matrix factorization. Specifically, a number of network measurement among a random set of nodes, which are reachable to the desired two nodes, have been implemented to establish a network distance tensor, and adaptive SGD-based solutions have been leveraged to factorize it such that the unknown distances can be inferred approximatively from it in the form of confidence intervals.

Fig. 4 illustrates the operations of TNDP. It works as follows. Initially, TNDP begins from establishing a network distance tensor $\mathbb{R}^{D_1 \times D_2 \times \ldots \times D_k}$, which has k $n*n$ distance matrices D_1,D_2,...,D_k while missing some elements, by performing no more than k network measurements among more than n nodes. Then, each network matrix has been factorized into two smaller network matrices to find the approximate locations for the desired nodes, with the SGD-based learning solutions by introducing weight matrix, regularization coefficient, and minibatch gradient descent with the exponential decay rates. Finally, the unknown distance d_{ij} can be obtained in the form of a confidence interval $[\mu_{ij}, \nu_{ij}]$, which are derived from Eqs. (8) and (12). This can ensure that the confidence interval of an unknown distance is derived for the many-to-one services distributed applications. The details of TNDP are presented below.

3.2 Distance Tensor Establishment

In order to establish an $n \times n \times k$ network distance tensor $d_{ij} \in \mathbb{R}^{D_1 \times D_2 \times \ldots \times D_k}$, node i first sends a probe packet to the unreachable node j via broadcast, where direct network measurement between them cannot be performed. Once receiving this packet, each node l, who is its p-hop neighbor, adds the arriving time $t_{l-1,l}$ to the packet header, and then forwards it to its neighbors and source node i. Meanwhile, it initiates a new packet to desired node j. Each forwarder, who receives this packet, will repeat this action as node l.

Through this way, there will be more than n nodes that participate in the network measurement. Finally, each forwarder, including node i, will send the feedback packet to a server, which is responsible for matrix factorization and confidence interval calculations for distance prediction. Once receiving this packet, this server will begin to establish an $n{\times}n{\times}k$ network distance tensor $d_{ij}{\in}\ \mathbb{R}^{D_1\times D_2\times\ldots\times D_k}$. Each D_l ($1{\le}l{\le}k$) will consist of $n{*}n$ distances among n nodes. Let $S(D_i, D_j)$ denote the number of the same nodes included in both D_i and D_j. In order to reduce the measurement overhead, two adjacent distance matrices D_i and D_{i+1} satisfy

$$\begin{cases} 0 < S(D_i, D_{i+1}) \le n-1, \\ D_i \neq D_{i+1}, \\ d_{ij} \notin D_i \cup D_{i+1}. \end{cases} \tag{14}$$

This means that both D_i and D_{i+1}, illustrated in Fig. 4(a), will include most same nodes and distances, mainly depending on the node density, to infer the unknown distances. A same column (row) of two different matrices D_i and D_{i+1} within $\mathbb{R}^{D_1\times D_2\times\ldots\times D_k}$ is supposed to correspond to the network distances among the same nodes at different time epochs, when D_i and D_{i+1} have been established, respectively. It is noticed that each $a_{ii}{\in}D_l$ ($1{\le}i{\le}n$) is set to be 0, similar to [12], [13], [8], while without the need to be measured.

3.3 Adaptive SGD-based Matrix Factorization

Matrix factorization is concerned with the ability to constantly learn its current results based on previous ones following the design rules for solving various problems [24]. The solutions available for matrix factorization include SGD [22], Momentum [25], Adam [26], etc. As the basis of most matrix factorization schemes, all of them have salient features, including randomly choosing one training sample to update its parameter along the negative gradients of the chosen sample, and the ability to process large-scale measurement data dynamically. However, each of them cannot achieve our desired goal due to overfitting and slow convergence. Therefore, we propose our adaptive SGD-based matrix factorization in an alternating manner, and present it as follows.

Once receiving the created network distance tenser $\mathbb{R}^{D_1\times D_2,\ldots,\times D_k}$, the server begins to factorize each distance matrix D_i into the production of X_i and Y_i. Each iteration for inferring the missing distance d_{ij} follows the negative gradient directions of all chosen samples from node i to node j [27]. It is unlikely to happen that the outputs keep the same as the inputs at each iteration. Therefore, we introduce an objective function described in Eq. (5) that can quantify the error between the output and input. In order to minimize the error, some adjustable parameters like the weight matrix W and regularization coefficient λ are also introduced.

To obtain two best X_l and Y_l over D_l, we define the regularized losses $[f_{ij}]_l$ and $[f_{ji}]_l$ with unreachable nodes i and j as follows:

$$\begin{cases} [f_{ij}]_l = f([d_{ij}]_l, x_{l,i}y_{l,j}^T) + \lambda x_{l,i}x_{l,i}^T, \\ [f_{ji}]_l = f([d_{ji}]_l, x_{l,j}y_{l,i}^T) + \lambda y_{l,i}y_{l,i}^T. \end{cases} \tag{15}$$

Thus, the gradients of $[f_{ij}]_l$ and $[f_{ji}]_l$ can be computed as

$$\begin{cases} \dfrac{\partial [f_{ij}]_l}{\partial x_{l,i}} = \dfrac{\partial f([d_{ij}]_l, x_{l,i}y_{l,j}^T)}{\partial x_{l,i}} + \lambda x_{l,i}, \\ \dfrac{\partial [f_{ji}]_l}{\partial y_{l,i}} = \dfrac{\partial f([d_{ji}]_l, x_{l,j}y_{l,i}^T)}{\partial y_{l,i}} + \lambda y_{l,i}. \end{cases} \tag{16}$$

Along with the negative of $\frac{\partial [f_{ij}]_l}{\partial x_{l,i}}$ and $\frac{\partial [f_{ji}]_l}{\partial y_{l,i}}$, both $[f_{ij}]_l$ and $[f_{ji}]_l$ can be minimized, respectively, such that Eq. (5) or Eq. (6) can be achieved.

To infer the missing elements more accurately, it requires to train more samples simultaneously for each parameter update as far as possible. Therefore, we introduce the minibatch gradient descent rules w.r.t of f_{ij} and f_{ji} and define as follows:

$$\begin{cases} \dfrac{\partial f_{ij}}{\partial x_i} \leftarrow \displaystyle\sum_{j=1}^{n} w_{ij} \dfrac{\partial f_{ij}}{\partial x_i}, \\ \dfrac{\partial f_{ij}}{\partial y_i} \leftarrow \displaystyle\sum_{j=1}^{n} w_{ij} \dfrac{\partial f_{ij}}{\partial y_i}. \end{cases} \tag{17}$$

This means that the chosen samples to infer the missing distance d_{ij} refer to all the distances to nodes i and j in the given distance matrices.

Built on Eq. (4), the gradients of $[f_{ij}]_l$ and $[f_{ji}]_l$ are

$$\begin{cases} \dfrac{\partial [f_{ij}]_l}{\partial x_{l,i}} = -([d_{ij}]_l - x_{l,i}y_{l,j}^T)y_{l,j}, \\ \dfrac{\partial [f_{ji}]_l}{\partial y_{l,i}} = -([d_{ji}]_l - x_{l,j}y_{l,i}^T)x_{l,i}. \end{cases} \tag{18}$$

Then, the minibatch gradient descent rules w.r.t $[f_{ij}]_l$ and $[f_{ji}]_l$ can be computed as

$$\begin{cases} ([d_{ij}]_l - x_{l,i}y_{l,j}^T)y_{l,j} \leftarrow \displaystyle\sum_{j=1}^{n} w_{ij}([d_{ij}]_l - x_{l,i}y_{l,j}^T)y_{l,j}, \\ ([d_{ji}]_l - x_{l,j}y_{l,i}^T)x_{l,i} \leftarrow \displaystyle\sum_{j=1}^{n} w_{ji}([d_{ji}]_l - x_{l,j}y_{l,i}^T)x_{l,j}. \end{cases} \tag{19}$$

In order to know the missing elements quickly, how to choose a proper *stepsize*, l_s, *i.e.*, *learning rate*, which controls the speed of learning, becomes a challenge to be considered. Notice that too large or small stepsize of learning would result in data overflow and converge slowly, therefore it is required to design an adaptive stepsize to speed up the learning. One innovative approach is to gradually decrease the value of l_s so as to make sure it converges quickly, by introducing the exponential decay rates, similar to Adam [26]. Let β_1 and β_2 stand for two exponential decay rates w.r.t. the gradient and the squared gradient, and denote β_1^m and β_2^m as the power of β_1 and β_1, respectively, through m learning iterations. Then, the adaptive stepsize l_s can be computed as

$$l_s = \frac{l_0\sqrt{1-\beta_2^m}}{1-\beta_1^m+\epsilon}, \tag{20}$$

where ϵ is an extremely small smoothing term to prevent the denominators of Eq. (20) from zero, and l_0 denotes the default setting of l_s. The upper bounds of l_s are $|l_s|\leq\frac{l_0\sqrt{1-\beta_2}}{1-\beta_1}$ in the case of $(1-\beta_1)>\sqrt{1-\beta_2}$ and $|l_s|\leq l_0$ otherwise. By introducing l_s, we can ensure that the adaptive learning rates can be obtained to update parameters.

Then, along the negative gradient directions, the locations $(x_{l,i},\ y_{l,i})$ of node i over D_l comply with the following rules

$$\begin{cases} x_{l,i} \leftarrow \lambda x_{l,i} + \dfrac{l_0\sqrt{1-\beta_2^m}}{1-\beta_1^m+\epsilon}\displaystyle\sum_{j=1}^{n} w_{ij}([d_{ij}]_l - x_{l,i}y_{l,j}^T)y_{l,j}, \\ y_{l,i} \leftarrow \lambda y_{l,i} + \dfrac{l_0\sqrt{1-\beta_2^m}}{1-\beta_1^m+\epsilon}\displaystyle\sum_{j=1}^{n} w_{ji}([d_{ji}]_l - x_{l,j}y_{l,i}^T)x_{l,j}. \end{cases} \tag{21}$$

to be updated in an alternating manner. The default settings for l_0, β_1, β_2 and ϵ depend on empirical value and real-world applications, and can follow the suggestions in [26]. The pseudocode of this operation is given in Algorithm 1.

Algorithm 1: Adaptive SGD-based matrix factorization

```
1  Input: $d_{ij}\in\mathbb{R}^{D_1\times D_2\times\ldots\times D_k}$, $l_0$, $\beta_1$, $\beta_2$, and $k$
2  Output: $D_1 = X_1Y_1^T$,$\cdots$,$D_k = X_kY_k^T$
3  for $\forall D_l$ $(1\leq l\leq k)$ do
4      compute $\frac{\partial[f_{ij}]_l}{\partial x_{l,i}}$ and $\frac{\partial[f_{ij}]_l}{\partial x_{l,i}}$ ;
5      $\frac{\partial[f_{ij}]_l}{\partial x_{l,i}} \leftarrow \sum_{j=1}^{n} w_j^i \frac{\partial[f_{ij}]_l}{\partial x_{l,i}}$;
6      $\frac{\partial[f_{ji}]_l}{\partial y_{l,i}} \leftarrow \sum_{j=1}^{n} w_j^i \frac{\partial[f_{ji}]_l}{\partial y_{l,i}}$;
7      if $[f_{ij}]_l$ or $[f_{ji}]_l$ not converged do
8          $l_s\leftarrow\frac{l_0\sqrt{1-\beta_2^m}}{1-\beta_1^m+\epsilon}$;
9      update $X_l$ and $Y_l$ built on Eq. (21);
10 end for
```

3.4 Distance Prediction with Confidence Intervals

There will exist $2k$ small matrices $X_1, X_2, \ldots, X_k$ and Y_1, Y_2,$\ldots$, Y_k referring to nodes i and j through adaptive SGD-based matrix factorization. Each $x_{l,i}\in X_l$ and $y_{l,j}\in Y_l$ can approximately act as the incoming vector and outgoing vector of nodes i and j, respectively. This means that the missing entry $d_{ij}\in\mathbb{R}^{D_1\times D_2\times\ldots\times D_k}$ can be derived built on Eq. (22) and will change in a certain extent, following Gaussian distribution $N(\bar{d}_{ij},\ \sigma_{ij}^2)$.

Let $x_{l,i}$=$\{[x_{i1}]_l$,$[x_{i2}]_l, \ldots, [x_{im}]_l\}$ and $y_{l,i}$=$\{[y_{i1}]_l$,$[y_{i2}]_l$, ..., $[y_{im}]_l\}$, respectively. Then, the l-th predicted distance $[d_{ij}]_l$ from nodes i to j derived from X_l and Y_l over D_l can be computed as

$$[d_{ij}]_l \approx x_{l,i}y_{l,j}^T = \sum_{k=1}^{m}[x_{ik}]_l[y_{jk}]_l. \tag{22}$$

With the known distances $[d_{ij}]_1$, $[d_{ij}]_2$, $\cdots$, and $[d_{ij}]_k$, we can obtain $\bar{d}_{ij}$ built on Eq. (8). Both $\bar{d}_{ij}$ and variance σ_{ij}^2 are the essential statistical characteristics to predict d_{ij} in the form of confidence intervals. Suppose that the confidence level is $\theta\%$ and let η be $(1-\theta\%)$, then we have

$$P(\mu_{ij} < d_{ij} < \nu_{ij}) = \theta\%. \tag{23}$$

Given $u_{\eta/2}$, from the table of Gaussian distribution, we can know that

$$u_{\eta/2} = \chi. \tag{24}$$

Built upon Eq. (12), the exact $\hat{\mu}_{ij}$ and $\hat{\nu}_{ij}$ can be given by

$$\begin{cases} \mu_{ij} = \bar{d}_{ij} - \chi\sqrt{\dfrac{\sigma_{ij}^2}{n}}, \\ \nu_{ij} = \bar{d}_{ij} + \chi\sqrt{\dfrac{\sigma_{ij}^2}{n}}. \end{cases} \tag{25}$$

Given any missing entry, its confidence interval can be figured out by Eq. (25). To facilitate a better understanding, the pseudocode of distance prediction with confidence intervals is described in Algorithm 2.

Algorithm 2: Network distance prediction in the form of confidence intervals

```
1  Input: $D_1 = X_1Y_1^T$,$\cdots$,$D_k = X_kY_k^T$
2  Output: each missing entry $d_{ij}\in[\mu_{ij},\nu_{ij}]$
3  for $x_{l,i}\in X_l$, $y_{l,j}\in Y_l$ and $l\leq k$ do
4      $[d_{ij}]_l$=$x_{l,i}y_{l,j}$;
5  end for
6  foreach missing entry $d_{ij}$ do
7      $\bar{d}_{ij} \leftarrow \frac{\sum_{l=1}^{k}[d_{ij}]_l}{k}$;
8      $\sigma^2 \leftarrow \frac{\sum_{l=1}^{k}([d_{ij}]_l-\bar{d}_{ij})^2}{k}$;
9      calculate $[\mu_{ij},\nu_{ij}]$ in Eq.(25) with $\bar{d}_{ij}$ and $\sigma^2$;
10 end foreach
```

4 Performance Evaluation

In this section, we evaluate the performance of TNDP via simulations in Matlab R2020b over two available data sets, and compare it with three famous network distance prediction solutions: IDES [12], DMF [13], and DMFSGD [8]. First, we introduce the simulation environments, and then present the evaluation metrics. Finally, we provide the details of simulation results and conduct performance analysis.

4.1 Simulation Environments

We implemented the experiment on a desktop computer equipped with Intel i5-9400 CPU and 16GB RAM. Similar to [13], two publicly well-known data sets, collected from the real Internet measurements, are used in all experiments.

- P2PSim1953 [21]: It was acquired from the P2PSim project that includes static RTT measurements among 1953 Internet DNS servers. We used the raw data that each distance pairs consists of a series of five samples, resulting in a 1953×1953×5 tensor.
- PlanetLab490 [18]: It was collected from the PlanetLab Pairwise Ping Project that includes static RTT measurements among 490 nodes with nine days. Each pairwise measurement sample is taken between a 14.7-hour interval and aligned to a tensor of 490×490×18.

It is supposed that such original data sets include no erroneous results. Note that these data sets are obtained from the real-world networks, therefore, there are many TIV edges. These are allowed in IDES, DMF, and DMFSGD, following the real routing policies such as sub-optimal routing and asymmetric routing, while cannot exist in Euclidean embedding solutions. Some network distances in all these data sets were not measured and represented with the length of -1 for facilitating matrix factorization, similar to [12], [13], [8], but will be derived with our solution in the form of confidence intervals, enabling to infer the unknown network distances more accurately.

Each distance matrix $D_l \in \mathbb{R}^{D_1\times D_2\times\ldots\times D_k}$ consists of the most distances among 1953 nodes (n=1953) and 490 nodes (n=490) for TNDP, IDES, DMF, and DMFSGD over data sets P2PSim1953 and PlanetLab490, respectively. The network distance among all nodes for IDES, DMF, DMFSGD and our solution are represented with the latency in the form of RTT. Table 3 lists 20 widely existing unknown distances, which will be predicted in our simulations, in TNDP, IDES, DMF, and DMFSGD over these two data sets. There are more than 5 created distance matrices for TNDP (*i.e.*, $k\geq5$), used for establishing $\mathbb{R}^{D_1\times D_2\times\ldots\times D_k}$, and only one for IDES, DMF, and DMFSGD. The 95% confidence interval has been calculated to predict network distances with the created matrices. Unless specified, the regularization coefficient λ and l_0 are set to be 0.01 and 0.1, while the good initial settings [26] for β_1, β_2 and ϵ are 0.9, 0.999 and 10^{-8}, respectively. The default rank ζ of two $n\times m$ factorized matrices X and Y over D is set to be 10, and can be considered equivalent to m since $m\ll n$ =1953 or n =490.

There are five basic simulation scenarios designed to evaluate our approach.

- **Different neighbor density scenario:** Each node is distributed in different locations and maintains alterable neighbor densities in the network. Given node n_i, the neighbor density, denoted by ρ_i, refers to the number of its one-hop neighbors, whom it can directly send the packets to within its maximum transmission range, and can be represented as
$$\rho_i = \{p|n_1, n_2, \cdots, n_p\}. \tag{26}$$
Each node which participates in establishing distance tensor $\mathbb{R}^{D_1\times D_2\times\ldots\times D_k}$, will measure all distance information between it and p neighbors n_1, n_2, $\cdots$, and n_p as far as possible.
- **Different neighbor set scenario:** Each node will keep the same neighbor densities, while select different nodes to form its neighbor set in each experiment. Given node n_i, the neighbor set, denoted by N_i, is defined as the total neighbors of its, and can be given by
$$N_i = \{n_1, n_2, \cdots, n_p\}. \tag{27}$$
- **Different dimensional rank scenario:** The low-rank requirement is the inherent characteristics of matrix factorization. Thus, the rank, denoted by ζ, of two small factorized matrices X and Y over D is set in the range of $[\zeta_1, \zeta_2]$ in all experiments.
- **Different learning rate scenario:** The default setting of learning rate l_0 in Eq. (20) has been changed in the range of [l_1, l_2] to accelerate the convergence of learning rate l_s. With different default settings and increasing learning iterations, the adaptive l_s can be achieved.
- **Different regularization coefficient scenario:** The regularization coefficient λ, referring to Eqs. (5) and (21), will vary from the range [λ_1, λ_2] to control the norm of X and Y to avoid overfitting. This will help to reduce errors on the unseen data used in learning due to overfitting.

4.2 Evaluation Metrics

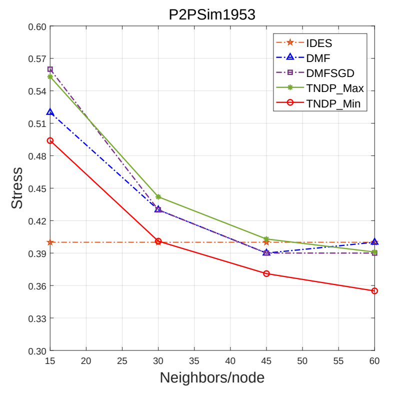
For the sake of evaluating the effectiveness of our solution, two following criteria are used in our simulations.

Stress: The stress is defined as
$$\sqrt[2]{\frac{\sum_{ij}(d_{ij}-\overline{d}_{ij})^2}{\sum_{i\neq j} d_{ij}^2}}. \tag{28}$$

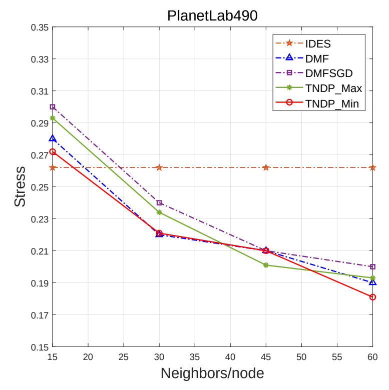
It is used to estimate the global fitness between D and $\hat{D}$. Considering that our results are represented as intervals, we use the minimum and maximum of the interval, defined as TNDP_Min and TNDP_Max, respectively, to assess the stress.

Relative Error (REE): The relative error, denoted by e_r, is defined as
$$e_r = \frac{|d_{ij}-\overline{d}_{ij}|}{\min[d_{ij}, \overline{d}_{ij}]}. \tag{29}$$

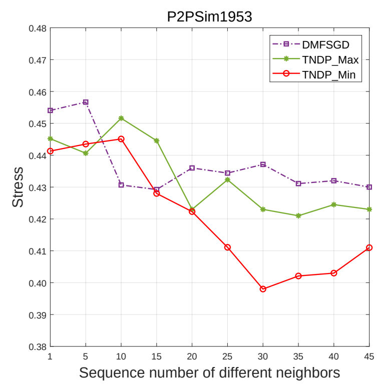
This metric is used to estimate the accuracy of network distance prediction.



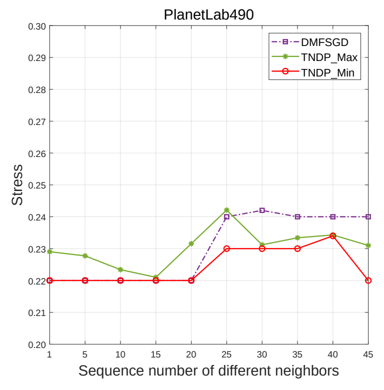
(a) Stress with varied neighbor number



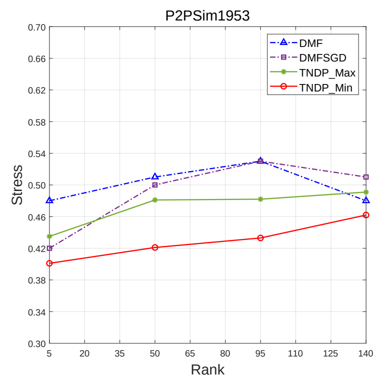
(b) Stress with varied neighbor number



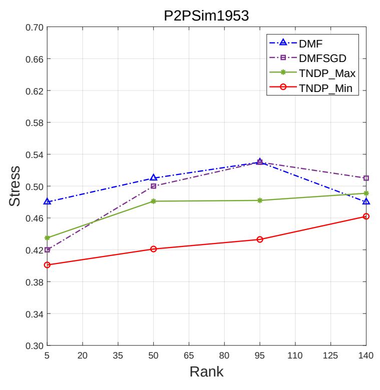
(c) Stress with varied neighbor sets



(d) Stress with varied neighbor sets



(e) Stress with varied dimension



(f) Stress with varied dimension

Fig. 5: Simulation results on stress with varied neighbor densities, neighbor sets and dimensional rank, which run over two publicly available data sets P2PSim1953 and PlanetLab490, respectively.

4.3 Simulation Results

4.3.1 Impact of neighbor densities

In this subsection, the neighbor densities vary from 15 neighbors/node to 60 neighbors/node, mainly by changing the maximum transmission ranges of nodes who involves in distance prediction, to estimate the stress of IDES, DMF, DMFSGD and TNDP. The rank ζ of two small factorized matrices X and Y is set to be 10, the initial learning rate l_0 is equal to 0.5, and λ=1.

Figs. 5(a)-(b) indicate the stress of IDES, DMF, DMFSGD and TNDP with different neighbor densities. It is observed that increasing neighbors are beneficial to decrease the stress for all of them, because more nodes involved in distance prediction can help to reconstruct more real network distances. Among four of them, IDES has changeless stress with the different neighbor densities, since it is built on the fixed landmarks to work (with 10% nodes as landmarks in our simulations). Unlike DMF and DMFSGD, TNDP represents the stress with confidence interval in the range of [0.35, 0.56] and [0.18, 0.30] over datasets P2PSim1953 and PlanetLab490, respectively, with different neighbor densities, and thus can more accurately figure out the unknown distances.

To facilitate better understanding, Table 3 has listed 20 widely existing missing distances for four approaches over these two data sets and given their predicted network distance under these settings when their neighbor densities are set to be 45 neighbors/node. The missing distance $d_{500,42}$, for example, is represented with confidence interval [107.97, 114.19] for TNDP, while only an unique number 196.87, 219.56, 162.73 for DMFSGD, DMF, and IDES, respectively, over data set P2PSim1953. It shows that TNDP can infer the missing distances more effectively with confidence intervals than three of them.

4.3.2 Impact of different neighbor sets

In this subsection, we keep the neighbor densities of nodes to be 30 neighbors/node, and randomly select different nodes as the neighbors of each node which involves in distance prediction. There are 25 neighbor sets, where 30 neighbors are selected randomly for assessing the stress of our TNDP.

Figs. 5(c)-(d) illustrate the stress of TNDP and DMFSGD with different neighbor sets. It shows that the stress of them is different even if all of them keep the same neighbor densities. This is because different neighbors played distrinct roles to reconstruct the missing network distances. The results from Figs. 5(c)-(d) indicate that TNDP has -3.4%-9.1% and 0-8.3% reduction than DMFSGD in the stress over data sets P2PSim1953 and PlanetLab490, respectively.

TABLE 3: Network distance prediction for missing distances in four approaches over two data sets

Solutions	Data sets	Network distance prediction for missing distance (s) between nodes i and j									
		$d_{500,42}$	$d_{993,44}$	$d_{1054,49}$	$d_{1284,54}$	$d_{1465,58}$	$d_{1121,62}$	$d_{1729,66}$	$d_{590,69}$	$d_{1206,71}$	$d_{880,76}$
		$d_{284,57}$	$d_{166,41}$	$d_{443,69}$	$d_{316,97}$	$d_{485,9}$	$d_{299,15}$	$d_{92,17}$	$d_{109,26}$	$d_{260,37}$	$d_{387,56}$
TNDP_Max	P2PSim1953	114.19	39.56	105.54	76.10	227.89	43.27	126.70	180.50	55.64	320.55
	PlanetLab490	83.17	104.25	292.63	129.75	327.47	55.52	305.59	148.01	156.51	40.54
TNDP_Min	P2PSim1953	107.97	32.87	88.88	59.19	186.64	36.31	86.43	162.35	43.28	303.80
	PlanetLab490	77.27	100.43	283.69	124.15	310.91	46.45	290.19	143.13	150.85	34.93
DMFSGD	P2PSim1953	196.87	46.15	42.45	78.09	167.48	92.43	88.15	267.34	42.57	219.32
	PlanetLab490	77.09	100.15	276.89	129.09	314.46	52.11	323.12	142.81	159.27	46.87
DMF	P2PSim1953	219.56	26.80	43.54	69.49	237.93	63.41	78.18	393.61	45.43	273.76
	PlanetLab490	74.88	96.93	275.98	136.88	319.46	27.63	290.84	141.90	154.99	47.62
IDES	P2PSim1953	162.73	51.71	41.75	83.35	114.98	89.95	101.93	341.67	51.16	170.57
	PlanetLab490	68.40	103.14	267.01	118.30	271.77	29.46	216.33	161.57	153.11	38.85

4.3.3 Impact of different dimensional rank

In this subsection, the rank ζ of two factorized matrices X and Y over D is set in the range of $[5, 140]$ in all experiments. The dimension of locations $(x_{l,i},\ y_{l,i})$ of node i over D_l is equal to 5. The settings of α and λ are 0.5 and 1, respectively.

Figs. 5(e)-(f) elaborate that stress of IDES, DMF, DMFSGD and TNDP as the rank ζ changes from 5 to 140. Different from IDES, DMF, and DMFSGD, TNDP considers the predicted network distances as confidence intervals built on matrix factorization, thus has a varied stress with different rank. The results from Figs. 5(e)-(f) show that it has achieved 1.7%-19.3% and -8.9%-13.8% reduction than DMF and DMFSGD over data set P2PSim1953, and 4.8%-18.5% and 3.1%-18.4% reduction than DMF and DMFSGD over data set PlanetLab490, respectively.

4.3.4 Impact of different learning rates

In this subsection, we evaluate the stress of learning rate l_s by changing the default learning rate l_0 and increasing learning iterations m, described in Eq. (21). Because only both DMFSGD and TNDP have introduced adaptive learning rate, we just listed the stress of them with different learning rates.

Table 4 indicates the learning time comparison between DMFSGD and TNDP for matrix factorization over two available data sets. It shows that TNDP is much more efficient than DMFSGD, with a 0.58s-1.46s reduction from 5 matrices and a 0.23s-0.34s reduction from 18 matrices over data set P2PSim1953 and PlanetLab490, respectively, to learn locations of nodes. This is because DMFSGD exploits the constant rate to learn all locations, which converges slowly to local minima. Figs. 6(a)-(b) demonstrate that the stress of DMFSGD and TNDP as the default learning rate changes from 0.01 to 1.00. It is shown that TNDP is less insensible to the learning rate than DMFSGD along the gradient descent of training data. This is because TNDP has introduced adaptive, rather than the pre-existing, learning rates to fit the locations of nodes. Different training data requires individual learning rates. The learning rates of TNDP in training data will be adjusted automatically to ensure that the optimized predicted results can be achieved as far as possible.

4.3.5 Impact of different regularization coefficient

In this subsection, we evaluate the stress of regularization coefficient λ and change it in the range of [0.01, 1.00]. Notice that only DMFSGD and TNDP have introduced regularization coefficient, we just listed the stress of them with different regularization coefficient.

Figs. 6(c)-(d) demonstrate the stress of DMFSGD and TNDP as the regularization coefficient changes from 0.01 to 1.00. It can be seen that the stress of TNDP is improved when λ increases much larger due to low overfitting. Therefore, TNDP can be more applied to improve the precision of network distance prediction than DMFSGD.

4.3.6 Cumulative Distribution Function of REE

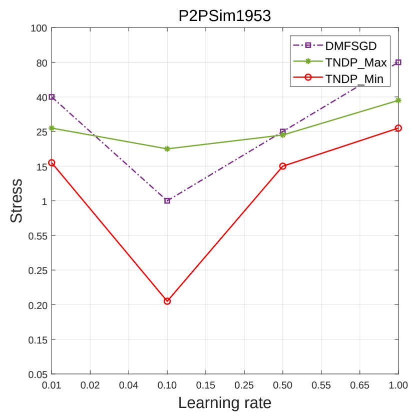
In this subsection, we evaluate the cumulative distribution function (CDF) of IDES, DMF, DMFSGD and TNDP as REE changes in the range of [0, 1.0]. In order to better demonstrate differences among IDES, DMF, DMFSGD and TNDP, the line markers in Figs. 6 (e)-(f) have been deleted, which first exist in Fig. 5 and Figs. 6 (a)-(d).

Figs. 6(e)-(f) illustrate the CDF of relative errors of IDES, DMF, DMFSGD and TNDP with different REE over datasets P2PSim1953 and PlanetLab490, respectively. The results indicate that four of them have similar accuracy of predicted distances. Furthermore, both TNDP and DMFSGD have similar moderate improvements in CDF, with less relative errors, due to introducing the SGD-based learning.

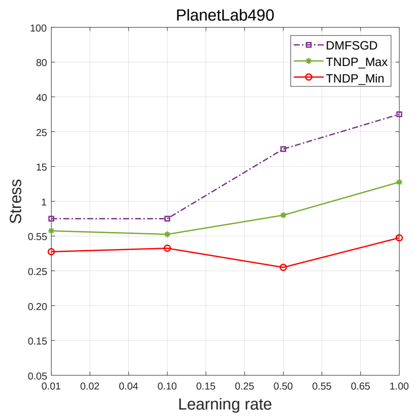
5 Related Work

There have been numerous network distance prediction approaches proposed for the various distributed applications with network coordinates [28], [11], [29], [30], [31]. A detailed investigation on them is given in [7] and [4]. On the basis of the embedded network coordinates, these approaches can be divided into two categories: (a) Euclidean embedding and (b) non-Euclidean embedding.

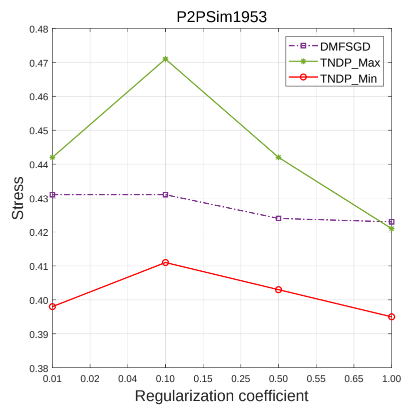
Euclidean embedding: This idea of these solutions is to embed network nodes into a Euclidean space to



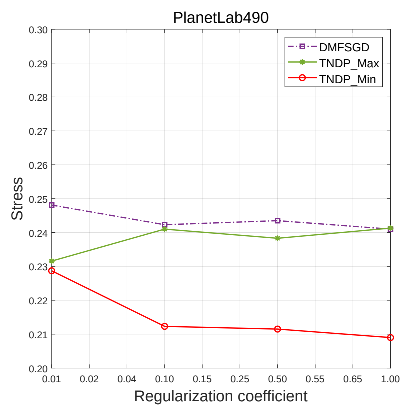
(a) Stress with varied learning rate



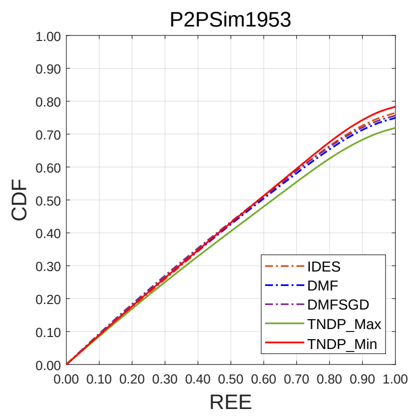
(b) Stress with varied learning rate



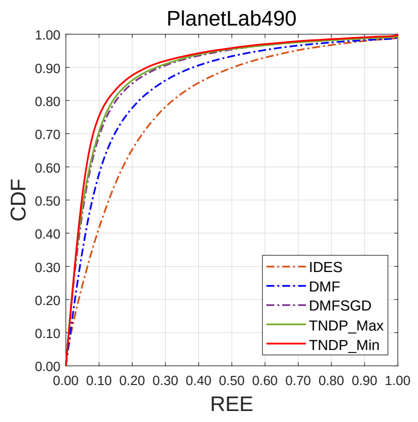
(c) Stress with varied regularization coefficient



(d) Stress with varied regularization coefficient



(e) CDF of REE over data set P2PSim1953



(f) CDF of REE over data set PlanetLab490

Fig. 6: Simulation results on stress and REE with varied learning rate and regularization coefficient, which run over two publicly available data sets P2PSim1953 and PlanetLab490, respectively.

TABLE 4: Learning time comparison of solutions to learn each location over two data sets

Solutions	Data sets	Learning times used in each matrix (s)																	
		D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	D_{16}	D_{17}	D_{18}
TNDP	P2PSim1953	2.53	2.51	2.39	2.47	2.45	-	-	-	-	-	-	-	-	-	-	-	-	-
	PlanetLab490	0.88	0.83	0.79	0.89	0.82	0.79	0.83	0.90	0.91	0.79	0.98	0.89	0.91	0.89	0.94	0.79	0.80	0.89
DMFSGD	P2PSim1953	3.10	3.76	3.43	3.93	3.64	-	-	-	-	-	-	-	-	-	-	-	-	-
	PlanetLab490	1.19	1.12	1.12	1.15	1.10	1.13	1.12	1.15	1.18	1.07	1.21	1.14	1.20	1.18	1.20	1.06	1.11	1.18

predict network distances. More specifically, it firstly selects some landmarks and then measures the network distances to and from them. Thus, their locations can be derived by minimizing the error between the actual measure distances and the predicted distances. Each ordinary nodes can obtain its own coordinate once knows the distances to and from landmarks. Example of these include Global Network Positioning (GNP) [10], Virtual Landmarks [32], Vivaldi [28], PIC [11], NPS [29], ICS [30], BBS [33]. With the learn coordinates, the distance between any node-pairs can be directly computed in Euclidean distance. However, all of them cannot represent the real distance properties such as suboptimal routing or asymmetric routing, since Euclidean distances satisfy the triangle inequality and are inherently symmetric.

Non-Euclidean embedding: The basic idea of these approaches is to map nodes into non-Euclidean space, for example, matrix factorization [12], [8], Hyperbolic embedding [34], [35], and network geography [14], and model the network distances in non-Euclidean curves. Generally, most of them can more accurately figure out network distances than Euclidean embedding since the predicted network distances are persistent in the real networks [8], [34]. For instance, the extensive TIVs and asymmetric paths [36], which follow the real transmission paths of traffic, have been allowed to reconstruct the network distances. Example of these approaches include IDES [12], DMF [13], DMFSGD [8], Phoenix [37], and RMF [38]. All these usually are designed for stable networks and reference nodes [10], [12], [13], [8], where each network distance between two nodes has been represented in exact fixed value. This holds on most of networks where distances among reachable nodes only have small time-varying fluctuations, however cannot tolerate network change in flows, paths, and topologies, etc. In reality, the distance between two nodes will be change with time, meaning that their network distance is not an exact fixed value always but changes in a range. Therefore, unlike the previous work, the network distances in our proposed TNDP have been represented in the form of confidence intervals to more accurately infer the unknown distances. Furthermore, similar to DMFSGD, our TNDP is built on the distributed matrix-factorization to infer unknown network distances.

6 Conclusions

This paper has presented TNDP, an adaptive tensor-based approach for network distance prediction with confidence intervals. With a small set of network measurements among nodes selected randomly, a distance matrix tensor has been established and factorized with the adaptive learning to infer unknown distances, by introducing the important determinants, like weight matrix, regularization coefficient, the data volume of simultaneously learning, and gradient descent with the exponential decay rates. Benefiting from these novel solutions, TNDP converges fast, has high accuracy and can deal with dynamic measurements in large-scale networks. Extensive simulations on the real-world datasets show that TNDP outperforms the previous approaches on large-scale network distance prediction.

References

[1] H. Yin, X. Zhang, S. Zhao, Y. Luo, C. Tian, and V. Sekar, "Trade-offs between cost and performance for CDN provisioning based on coordinate transformation," *IEEE Transactions On Multimedia*, vol. 19, no. 11, pp. 2583–2596, 2017.

[2] W. Du, Y. Liao, N. Tao, P. Geurts, X. Fu, and G. Leduc, "Rating network paths for locality-aware overlay construction and routing," *IEEE/ACM Transactions on Networking*, vol. 23, no. 5, pp. 1661–1673, 2015.

[3] O. Bilgen and A. B. Wagner, "A new stable peer-to-peer protocol with non-persistent peers: The group suppression protocol," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 614–632, 2020.

[4] H. Huang, H. Yin, G. Min, D. O. Wu, Y. Wu, T. Zuo, and K. Li, "Network distance prediction for enabling service-oriented applications over large-scale networks," *IEEE Communications Magazine*, vol. 53, no. 8, pp. 166–174, 2015.

[5] X. Zhang, H. Yin, D. O. Wu, H. Huang, G. Min, and Y. Zhang, "SSL: A surrogate-based method for large-scale statistical latency measurement," *IEEE Transactions on Services Computing*, vol. 13, no. 5, pp. 958–968, 2020.

[6] M. Shahzad and A. X. Liu, "Accurate and efficient per-flow latency measurement without probing and time stamping," *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, pp. 3477–3492, 2016.

[7] B. Donnet, B. Gueye, and M. A. Kaafar, "A survey on network coordinates systems, design, and security," *IEEE Communications Surveys Tutorials*, vol. 12, no. 4, pp. 488–503, 2010.

[8] 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.

[9] D. Mirkovic, G. Armitage, and P. Branch, "A survey of round trip time prediction systems," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 1758–1776, 2018.

[10] T. E. Ng and H. Zhang, "Predicting internet network distance with coordinates-based approaches," in *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, 2002, pp. 170–179.

[11] M. Costa, M. Castro, R. Rowstron, and P. Key, "PIC: Practical Internet coordinates for distance estimation," in *24th International Conference on Distributed Computing Systems, 2004. Proceedings.*, 2004, pp. 178–187.

[12] Y. Mao, L. K. Saul, and J. M. Smith, "IDES: An Internet distance estimation service for large networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2273–2284, 2006.

[13] Y. Liao, P. Geurts, and G. Leduc, "Network distance prediction based on decentralized matrix factorization," in *Proc. of IFIP Netw. Conf.*, 2010, pp. 15–26.

[14] A. Jain and J. Pasquale, "Internet distance prediction using node-pair geography," in *2012 IEEE 11th International Symposium on Network Computing and Applications*, 2012, pp. 71–78.

[15] 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. 25, no. 2, pp. 724–737, 2017.

[16] G. Wang, B. Zhang, and T. Ng, "Towards network triangle inequality violation aware distributed systems," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, 2007, pp. 175–188.

[17] T. Bouchoucha, C. Chuah, and Z. Ding, "Topology inference of unknown networks based on robust virtual coordinate systems," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 405–418, 2019.

[18] 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. 25, no. 2, pp. 724–737, 2017.

[19] E. J. Candes and Y. Plan, "Matrix completion with noise," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 925–936, 2010.

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

[21] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating latency between arbitrary internet end hosts," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 5–18.

[22] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[23] A. M. Buchanan and A. W. Fitzgibbon, "Damped newton algorithms for matrix factorization with missing data," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 316–322.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[25] N. Qian, "On the momentum term in gradient descent learning algorithms," vol. 12, no. 1, 1999, pp. 145–151.

[26] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[27] R. K. Kolla, K. Jagannathan, and A. Gopalan, "Collaborative learning of stochastic bandits over a social network," *IEEE/ACM Transactions on Networking*, vol. 26, no. 4, pp. 1782–1795, 2018.

[28] F. Dabek, R. Cox, F. Kaashoek, and R. Morris, "Vivaldi: A decentralized network coordinate system," in *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, 2004, pp. 15–26.

[29] T. E. Ng and H. Zhang, "A network positioning system for the internet." in *USENIX Annual Technical Conference, General Track*, 2004, pp. 141–154.

[30] H. Lim, J. C. Hou, and C.-H. Choi, "Constructing internet coordinate system based on delay measurement," *IEEE/ACM Transactions on Networking*, vol. 13, no. 3, pp. 513–525, 2005.

[31] J. Cheng, Y. Liu, Q. Ye, H. Du, and A. V. Vasilakos, "Discs: A distributed coordinate system based on robust nonnegative matrix completion," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 934–947, 2017.

[32] L. Tang and M. Crovella, "Virtual landmarks for the internet," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, 2003, pp. 143–152.

[33] Y. Shavitt and T. Tankel, “Big-bang simulation for embedding network distances in euclidean space,” *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 993–1006, 2004.

[34] ——, “Hyperbolic embedding of internet graph for distance estimation and overlay construction,” *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 25–36, 2008.

[35] W. Zeng, R. Sarkar, F. Luo, X. Gu, and J. Gao, “Resilient routing for sensor networks using hyperbolic embedding of universal covering space,” in *2010 Proceedings IEEE INFOCOM*, 2010, pp. 1–9.

[36] C. Busch, R. Kannan, and A. V. Vasilakos, “Approximating congestion + dilation in networks via ”quality of routing” games,” *IEEE Transactions on Computers*, vol. 61, no. 9, pp. 1270–1283, 2012.

[37] 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 Transactions on Network and Service Management*, vol. 8, no. 4, pp. 334–347, 2011.

[38] Y. Fu and X. Xu, “Self-stabilized distributed network distance prediction,” *IEEE/ACM Transactions On Networking*, vol. 25, no. 1, pp. 451–464, 2017.

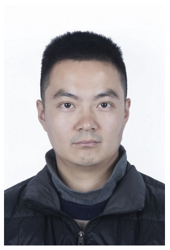
Haojun Huang is an Associate Professor in the School of Electronic Information and Communications at Huazhong University of Science and Technology, China. He received his PhD degree in Communication and Information Engineering from the University of Electronic Science and Technology of China in 2012, and the BS degree in Computer Science from Wuhan University of Technology in 2005. His current research interests include Internet of Things, Network Function Virtualization, Software-Defined Networking, and Artificial Intelligence for networking.

Li Li is currently pursuing a PhD degree in Information and Communication Engineering at Huazhong University of Science and Technology, Wuhan, China. His research interests include Network Traffic Prediction and Artificial Intelligence for networking.

Geyong Min is a Professor of High Performance Computing and Networking in the Department of Computer Science within the College of Engineering, Mathematics and Physical Sciences at the University of Exeter, United Kingdom. He received the PhD degree in Computing Science from the University of Glasgow, United Kingdom, in 2003, and the BS degree in Computer Science from Huazhong University of Science and Technology, China, in 1995. His research interests include Computer Networks, Wireless Communications, Parallel and Distributed Computing, Ubiquitous Computing, Multimedia Systems, Modelling and Performance Engineering.

Wang Miao is currently a Postdoctoral Research Associate in the Department of Computer Science at the University of Exeter, United Kingdom. He received his PhD degree in Computer Science from the University of Exeter, United Kingdom in 2017. His research interests focus on Network Function Virtualization, Software-Defined Networking, Unmanned Aerial Networks, Wireless Communication Networks, Wireless Sensor Networks, and Edge Artificial Intelligence.

Yingying Zhu is currently a Postdoctoral Researcher at Huazhong University of Science and Technology, Wuhan, China (HUST). She received the PhD degree in Communication and Information Engineering and the BS degree in Communication Engineering from HUST in 2018 and 2011, respectively. Her research interests include Computer Vision and Machine Learning.



Yangming Zhao is a research scientist with University at Buffalo. He received the BS degree in Communication Engineering and the PhD degree in Communication and Information System from University of Electronic Science and Technology of China in 2008 and 2015, respectively. His research interests include Network Optimization, Data Center Networks, Edge Computing and Transportation Systems.