RNC: A High-Precision Network Coordinate System

Jie Cheng[1] Xin Guan[2] Qiang Ye[1] Hongbo Jiang[2] Yan Dong[2]
[1]University of Prince Edward Island, Canada. China.
[2]Huazhong University of Science and Technology, China.
jcheng@upei.ca, guanxinhust@gmail.com, qye@upei.ca, hongbojiang2004@gmail.com, dongyan@mail.hust.edu.cn

***Abstract*—Network Coordinate System (NCS) has drawn much attention over the past years thanks to the increasing number of large-scale distributed systems that require the distance prediction service for each pair of network hosts. The existing schemes suffer seriously from either low prediction precision or unsatisfactory convergence speed. In this paper, we present a novel distributed network coordinate system based on Robust Principal Component Analysis, RNC, that uses a few local distance measurements to calculate high-precision coordinates without convergence process. To guarantee the non-negativity of predicted distances, we propose Robust Nonnegative Principal Component Analysis (RUN-PACE) which only involves convex optimization, consequently resulting in low computation complexity. Our experimental results indicate that RNC outperforms the state-of-the-art NCS schemes.**

I. Introduction

With the development of the Internet, many large-scale distributed systems that require the distance between each pair of network hosts have become more and more popular. A few example systems of this kind include file-sharing systems such as BitTorrent [1], multiplayer online games [2], position-based routing [3], distributed hash table (DHT) [4], content distribution network (CDN) [5], etc. In these systems, the distance can be based on a variety of different criteria, including delay, bandwidth, physical distance, etc. However, each network node always attempts to communicate with the optimal node if possible. For instance, in file sharing systems, a node prefers to download the files from a peer that enables the highest bandwidth. In multiplayer online games, a player hopes to interact with another player that leads to the shortest delay.

For small-scale systems, explicit measurement can be used to obtain the distance between each pair of network hosts. However, this approach does not scale well because the number of measurements that need to be completed exponentially increases with the number of hosts in the system. Assume that there are M hosts in a system, then M^2 explicit measurements should be finished in order to know the pairwise distance, which leads to disastrous traffic and computation overhead. To tackle the scalability problem, Network Coordinate System (NCS) has been proposed to predict the unknown distance using a small set of explicit distance measurements.

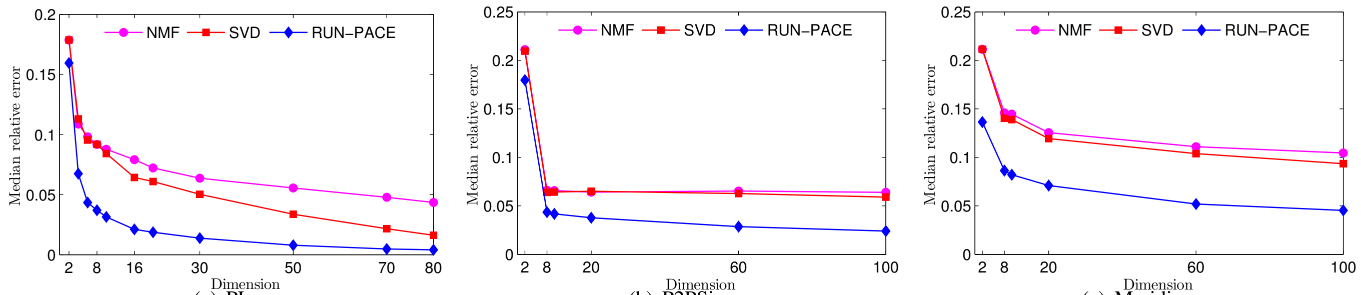
With NCS, the location of each node is mapped to a set of coordinates in a finite-dimensional space. Then the distance between each pair of network hosts can be easily calculated using the coordinates. No explicit measurement is required any more once a small set of measurements are completed. So far, there have been two different models that can be used map the location: Euclidean Distance Model [6] and Matrix Factorization Model [7], [8]. With Euclidean Distance Model, the pairwise distance is simply the Euclidean distance between two hosts. With Matrix Factorization Model, the distance is the dot product of two well-defined vectors. Thanks to the high precision of NCS, it has been used in more and more state-of-the-art applications such as cloud computing [9], [10] and social graph analysis [11], [12].

Euclidean Distance Model has two major limitations. First of all, it assumes that the distance between two hosts is symmetric, which could be incorrect in many applications. For example, the bidirectional delays between two routers in the Internet could be different due to asymmetric routing. Secondly, Euclidean Distance Model requires that the pairwise distances satisfy the triangle inequality, which is not true in some applications. Since Matrix Factorization Model does not have these limitations, it has been widely adopted in varied systems. Our research focuses on this model.

With NCS based on Matrix Factorization Model, all the distances in a distributed system can be formulated as an $M \times M$ matrix D, where M denotes the number of hosts in the system. In practice, D tends to be an approximately low-rank matrix although it is often not completely low-rank due to significant errors or outliers. The existing prediction algorithms [7], [8], [13], [14], [15] utilize rank reducing technologies, such as Singular Value Decomposition (SVD) and Nonnegative Matrix Factorization (NMF) [16], which attempt to use a completely low-rank matrix to approximate D by matching every explicit measurement indiscriminately. As a result, they suffer from corrupted entries and lead to imprecise predicted distances.

Inspired by Robust Principal Component Analysis (RPCA) [17], [18], we devised an innovative method, **Rob**u**st** **N**onnegative **P**rincip**A**l **C**ompon**E**nt Analysis (RUN-PACE), to improve the prediction accuracy. Different than the existing methods, RUN-PACE splits D into two $M \times M$ matrices, L and S, which satisfy $D = L + S$. L is a completely low-rank and nonnegative matrix while S is considered to be a matrix containing significant errors or outliers. Then only L is used to predict the unknown distance, which results in high prediction

978-1-4799-4852-9/14/$31.00 © 2014 IEEE



(a) PL (b) P2PSim (c) Meridian

Fig. 1. Comparison of reconstruction error for NMF, SVD and RUN-PACE

accuracy.

Specifically, we apply RUN-PACE to the distance matrix D to arrive at the nonnegative low rank part L, $L = \text{RUN-PACE}(D, d, \lambda)$. Then only L is used to calculate the unknown distances. Fig. 1 shows the performance of RUN-PACE in terms of the median of the relative errors. The sub-figures in Fig. 1 correspond to three different data sets: PL [19], P2PSim [20] and Meridian [21]. We can see that RUN-PACE leads to much lower prediction errors than SVD and NMF. The specific definition of median relative error, the details of RUN-PACE, and the description of the data sets will be presented in the following sections.

RUN-PACE is essentially a centralized method which does not scale well. To improve the scalability of the proposed method, we devised a distributed version of RUN-PACE, with which RUN-PACE is applied to local distance matrices instead of D. In our research, the final scheme is named **R**UN-PACE **N**etwork **C**oordinate system (RNC).

The detailed contributions of this paper are summarized as follows:

- We propose the Robust Nonnegative Principal Component Analysis (RUN-PACE) method, which innovatively ignores significant errors or outliers and uses the low-rank component of the distance matrix to match the assigned coordinates. Since the impact of significant errors or outliers is eliminated, RUN-PACE leads to high prediction accuracy. In addition, RUN-PACE guarantees the non-negativity of the low-rank component, which is compatible with most NCS applications.
- The RUN-PACE method utilizes Alternative Direction Method which only involves convex optimization. Consequently, RUN-PACE can generate the predicted distances at a very fast rate.
- We propose a coordinate system based on RUN-PACE, RNC, which utilizes a few local distance measurements to predict the unknown distance. It avoids the convergence process and achieves high prediction precision by introducing a small set of virtual landmarks, which scales very well.

The rest of the paper is organized as follows. Section II includes the details of RPCA and the models used by NCS. The network coordinate system problem is formulated mathematically in Section III. Section IV describes the details of RNC and Section V presents the RUN-PACE method. Our simulation results are included in Section VI. Finally, Section VII concludes the paper.

II. Preliminaries

In this section, we describe the advantages of NCS schemes with and without landmark hosts. In addition, an introduction to RPCA is presented.

A. NCS Schemes With Landmark Hosts

An important group of NCSs, such as GNP [6], ICS [22], PIC [23] and IDES [7], [8], utilize a small set of landmark hosts to obtain coordinates for individual hosts. In these systems, all hosts are separated to two categories: landmark hosts and ordinary hosts. Specifically, the coordinate system first selects a few hosts as landmark hosts and assigns them the coordinates for each one according to pair-wise distances between them. Then, if any normal host want to join the coordinate system, it can either measure the distances between landmark hosts and itself, and fetch the coordinates of landmark hosts as well. Based on these information, every single normal host can calculate the coordinates of itself.

Compared to complete distributed schemes [24], [14], [13], [15], landmark based NCSs have sufficient reasons to attract our attention continually. Complete distributed schemes often cost considerable time to converge to precise coordinates [24], [14], [13], [15]. And for the convergence of whole system, each host must join the communication and calculation process passively. On the contrary, landmark host based schemes achieve precise coordinates through communication with landmarks and local computation once, therefore, save overheads on network traffic and local computation extremely.

Furthermore, similar to other P2P schemes, distributed NCSs can not totally avoid the utilization of server or Rendezvous Point (RP). First of all, hosts with these schemes often have to connect server or RP at least once for bootstrapping to obtain a list of existing hosts as their reference host candidates. Secondly, while the system is initializing with small scale and no enough reference hosts to be selected, a centralized scheme often is proceeded on the server or RP to calculated the coordinates of hosts in the system [13].

In addition, distributed NCSs are often vulnerable to certain security attacks. Since every node in the system refers the information from other nodes which response at their own discretion, malicious nodes hiding in the system can falsify coordinates or delay the response deliberately for the sake of disrupting the system.

An apparent limitation of landmark based NCSs are the scalability nature since these systems rely on these landmark hosts, which could easily become the bottleneck. However, in practice, even without communication and measurement to all landmarks, the predicted distances still can be accurate, since normal hosts can act as the landmark hosts either after they have joined the system and calculated the accurate coordinates of their own. Such an improvement allows these schemes to extend to large scale networks easily [7], [8]. Actually, plenty of distributed schemes [14], [13], [15] are developed based on landmark based schemes directly [7], [8].

B. Robust Principal Component Analysis

Candès et al. [17] and Wright et al. [25] proposed robust principal component analysis (RPCA) method which aims to extract the "principal component" of a data matrix D. The principal component is denoted as L and the remaining component is represented as S. In addition, $D = L + S$. RPCA can precisely recover L from D by solving the convex optimization problem summarized in Notation (1):

$$\min_{L,S} \|L\|_* + \lambda\|S\|_1, \text{ subject to } D = L + S \tag{1}$$

where $\|\cdot\|_*$ denotes the nuclear norm of a matrix (i.e., the sum of its singular values), $\|\cdot\|_1$ denotes the sum of the absolute values of matrix entries, and λ is a positive weight parameter. The advantage of RPCA lies in the ability to exactly extract the low-rank component from a data matrix containing large errors or outliers. Different than RPCA, traditional principal component analysis suffers from corrupted entries, even if the only one entry is arbitrarily corrupted. Several applications of RPCA, such as background modeling from surveillance video, have been demonstrated by Candès et al. [17] and Wright et al. [25].

We were inspired to use RPCA to recover the low rank component L of the distance matrix D. Specifically, we attempted to reduce the prediction errors by matching the assigned coordinates to the distances in L. So far, there have been few studies that apply RPCA to the NCS problem because it is too expensive for RPCA to work with the distance matrix involving all the networks hosts (especially in a distributed manner). In addition, traditional RPCA does not consider the nonnegative constraint of the low-rank component L.

C. Solving of RPCA problem

Over the past years, there have been a variety of methods to solve the RPCA problem, such as Augmented Lagrangian Multiplier (ALM) method [26] and Alternating Direction Methods (ADM) [27]. Among these methods, Lin *et al.* proposed the ALM method which converges fast and leads to precise prediction [26]. However, all of the existing methods have not considered the nonnegative constraints of L.

For the problems involving inequality constraints using ALM method, [28] suggested that an inequality constraint could be transformed to an equality constraint by introducing a slack variable. LANCELOT introduced a practical example of ALM solution when both the equality and inequality constraints exist concurrently [29]. The Lagrangian function used in LANCELOT incorporates only the equality constraints, while inequality constraints are enforced explicitly in the subproblem. However, the LANCELOT method requires that the objective function is second-order differentiable. For the RPCA problem, the objective function is much less smooth.

Among previous mentioned solving schemes for RPCA, YUAN *et al.* proposed to solve RPCA problem using the ADM method, which is a practical improvement of the traditional ALM method for solving convex programming problem with linear constraints [30]. Versus classic ALM method, ADM performs minimization of Lagrangian function with respect to two variable matrices alternatively followed by the update of Lagrange multipliers. However, the classic ALM method requires a minimization of Lagrangian function with respect to two variable matrices jointly. Recently, several works have introduced ADM to solve the problems of nonnegative matrix factorization and nonnegative matrix completion involving both equality and inequality constraints [16].

III. Problem Formulation

In this section, we formulate the problem of NCS based on matrix factorization mathematically. Specifically, there are M hosts in the network under investigation. Each of the hosts is assigned a global ID in the range of 1 to M. D denote the distance matrix between host-pairs in the network. In our research, each host H_i is equipped an outgoing vector X_i and an incoming Y_i vector which are both d-dimension row vectors. The predicted distance from H_i to H_j is defined as the dot product between the outgoing vector X_i of H_i and the incoming vector Y_j of H_j, and vice versa. Following this definition, the outgoing vector X_i and incoming vector Y_i are the coordinates of H_i.

In our research, we propose to first decompose D into the summary of low rank part L and error part S, that is, $D = L + S$. Then, let coordinates of hosts in the network match the distances in low rank part L formulated as:

$$L(i,j) = X_i \cdot Y_j^T \tag{2}$$

Assume U and V be the coordinate matrices, in which the i-th row are the outgoing and incoming vector of host H_i respectively. We can formulate Eq. (2) as matrix representation in Eq. (3):

$$L = U \cdot V^T \tag{3}$$

However, individual host does not realize the whole D in a distributed manner but only local information. So we must first construct or approximate corresponding distance matrix. Then we elaborate to extract nonnegative low rank part based on local information precisely. Finally, we can utilize proper optimization algorithms to find the precise coordinates based on distance in nonnegative low rank part.

In our research, the hosts are classified into two groups: landmark nodes and normal hosts. Landmark nodes are a set of N distributed hosts with accurate coordinates. We assume that the pair-wise distances between every host in the set are available to the information server of RNC. Then landmark hosts' coordinates can be calculated with RNC. And we can add landmark hosts into a host set N_c in which every host have been assigned coordinates. e.g., by the King method [21] if the metric is RTT. An ordinary host is an arbitrary end node in the Internet, which is identified by a valid IP address.

Moveover, each normal host H_i randomly selects N hosts as its reference hosts from positioned node set N_c. With RNC, each normal host H_i in the network measures the distances between itself and its reference hosts. In addition, the coordinates of reference host are collected by host H_i either. Then each normal host uses these information to determine its own coordinates using RNC locally. We first define the notations of host sets, coordinate matrices and distance matrices involved in our scheme. Formally, we use the following notations:

- N_a denotes the host set of landmark hosts.
- $N = |N_a|$ denotes the number of reference hosts.
- U_a and V_a denote the outgoing and incoming coordinate matrices of landmark hosts with scale of $N \times d$ respectively. The according rows of U_a and V_a are the outgoing and incoming vectors of one landmark host respectively. V_a denotes the incoming coordinate matrix of landmark hosts with scale of $N \times d$ either.
- Q_a denotes the distance matrix containing all pair-wise distances between landmark hosts in N_a with scale of $N \times N$.
- N_c denotes the host set, host in which is assigned coordinates using our scheme. The reference hosts are random selected from this set. Note that N_c only contains N_a while the system is initialized.
- U_r and V_r denote the outgoing and incoming coordinate matrices of normal host H_i's reference hosts with scale of $N \times d$ respectively. The according rows of U_r and V_r are the outgoing and incoming vectors of one of the H_i's reference hosts respectively. Note that U_r and V_r are available on H_i.
- N_b denotes the host set including N reference hosts of normal host H_i and H_i itself. $N + 1 = |N_b|$ denotes the number of hosts in N_b.
- D_{out} denotes the measured distance vector containing all distances from normal host H_i to its N reference hosts with scale of $1 \times N$. Similarly, D_{in} contains the distances from N reference hosts to H_i with scale of $1 \times N$ either. Both D_{out} and D_{in} are available in host H_i.
- Q_b denotes the distance matrix containing all pair-wise distances between hosts in N_b with scale of $(N + 1) \times (N + 1)$. The Q_b is unavailable on H_i but could be approximated using U_r, V_r, D_{out} and D_{in}.

IV. RNC: A High-Precision Algorithm

Our scheme can be separated into two stages in which landmark hosts are well-positioned first, and then normal can be located precisely with the assistant of landmarks. The details of both stages are extended by estimating the submatrix of low rank part matrix, L. Once we estimate the submatrix of L involving target hosts, landmark hosts or normal hosts, precisely, we can utilize a proper optimization algorithm to find the precise coordinates for them.

First of all, the pair-wise distances between every host in the landmark set N_a can be measured and the results are reported to the information server. In this way, the pair-wise distance matrix Q_a is available to the information server of RNC. Based on Q_a, RNC will find precise coordinates for all landmark hosts. Then, each land mark host can fetch its coordinates from the information server in case the normal host's request. After that, we can add landmark hosts into host set N_c in which every host have been assigned its own coordinates.

Secondly, each normal host H_i randomly selects N hosts as its reference hosts from positioned host set N_c. While system is just initialized, only landmark hosts can been selected. With RNC, each normal host H_i in the network measures the distances between itself and its reference hosts in a distributed manner. In addition, the coordinates of reference host are collected by host H_i either. Then each normal host uses the information from its reference hosts to determine its own coordinates locally. Based on these information, we derive each normal host H_i's coordinates following three steps sequentially.

When we use RNC, the second stage can be carried out to calculate the coordinates on other normal hosts. Other hosts with calculated coordinates can be added to N_c either. Along with every host in the network calculates its coordinates, the whole system is located precisely. The details of the two stages involved in RNC are presented as follows.

A. Determine landmark hosts' coordinates

The first stage of RNC is to determine the coordinates of landmark nodes on the information server. As mentioned previously, the the pair-wise distance matrix Q_a involving all distances between landmark hosts is available to the information server of RNC. First of all, as mentioned previously, the the pair-wise distance matrix Q_a is available to the information server of RNC. Since the errors/outliers in distances affect the construction precision of coordinates, we strive to eliminate the impact of them. Therefore, we utilized RUN-PACE scheme to recover nonnegative low rank part A_a from Q_a for eliminating the compact of significant errors and outliers. As the recovered A_a only contains nonnegative distances, we can apply NMF algorithm directly over A_a to obtain landmark outgoing and incoming vectors X_i and Y_i in dimension d for each landmark host.

Extract Nonnegative Low Rank Part: To eliminate the impact of errors and outliers, we utilize RUN-PACE method to recover the low rank part from original distance matrix. Specifically, we intend to extract nonnegative low rank part A_a from distance matrix Q_a for landmark hosts using RUN-PACE algorithm, $A_a =$ RUN-PACE(Q_a, d, λ). The RUN-PACE algorithm guarantees every element of A_a, $A_a(i_{row}, j_{col}) \geq 0$, that is, ensures the nonnegative distances between landmark hosts. To solve this problem, we propose the RUN-PACE algorithm which is specified in Section V and its pseudocodes are outlined in Algorithm 3.

Optimize Landmark Hosts' Coordinates: As the proper distance A_a is available, we apply NMF algorithm on it directly to find precise coordinates for landmark hosts. The NMF is to solve following optimization problem as in Eq. (4).

$$\begin{aligned} \min_{U_a, V_a} \quad & \|A_a - U_a \cdot V_a^T\|_F^2 \\ \text{subject to} \quad & U_a \geq 0, V_a \geq 0 \end{aligned} \tag{4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix and $U_a \geq 0$ and $V_a \geq 0$ constrains each element of U_a and V_a to be greater than or equal to 0. The solving of NMF in Eq. (4) can refer [16]. After landmark hosts have been well-positioned, we can join them into the host set N_c.

In this paper, the pursuit of our idea is to find proper distance between hosts actively, instead accept the given distances passively. Due to the errors/outliers in the measured distances, we expect to extract the proper distances from them previously. So, we propose to recover the nonnegative low rank matrix A_a from $Q_a = A_a + E_a$ using our RUN-PACE method. The A_a is our desired ''proper" distance matrix between landmark hosts, which eliminates the influence of errors/outliers having been collected in E.

The first stage of our RNC algorithm is specified in Algorithm 1. The procedure RUN-PACE is to extract nonnegative low rank part which is further explained in Algorithm 3. To solve Eq. (4), we adopt NMF method which are implemented by procedure NMF.

Algorithm 1 Landmark hosts' coordinates determination using RNC

```
Input: $N_a, Q_a, d, \lambda$.
1: $A_a$ = RUN-PACE($Q_a, d, \lambda$).
2: $(U_a, V_a)$ = NMF($A_a, d$).
3: $N_c = N_a$.
Output: $(U_a, V_a, N_c)$.
```

B. Determine normal host's coordinates

As mentioned previously, each normal host H_i communicates with selected N reference hosts and saves the U_r, V_r, D_{out} and D_{in} locally. Based on these information, our scheme calculates H_i's coordinates following three steps sequentially. Similar to coordinates determination of landmark hosts, we expect to derive the proper distance between normal host and its reference hosts. Then we let the normal host's coordinates to match the proper distances according to the coordinates of reference hosts.

First of all, we first approximate the pair-wise distance matrix Q_b between normal host and its reference hosts using variables U_r, V_r, D_{out} and D_{in}. Then we use RUN-PACE again to recover the low rank part L_b from Q_b, which contains our expected proper distances after eliminating impact of the errors and outliers from Q_b. Finally, with referring to the coordinates of reference hosts, we find coordinates of H_i by matching the proper distances to its reference hosts with the help of Non-Negative Least Squares (NNLS) method. Once normal host H_i realize it own coordinates, the H_i can join into the located host set N_c and act as a reference host candidate.

Construct Local Target Matrix: In this part, we describe the steps for approximating the distance matrix Q_b of H_i and its reference hosts using local available U_r, V_r, D_{out} and D_{in}. Since the basic idea of matrix factorization model suggests that the distances can be indicated by the product of coordinate matrix, the distance between reference hosts can be approximated by $U_r V_r^T$. Furthermore, the measured distance vector D_{out} and D_{in} can be filled in according entries that indicates the distances to and from reference hosts. Without loss of generality, the Q_b can be approximated as in Eq. (5):

$$Q_b = \begin{bmatrix} U_r V_r^T & D_{in}^T \\ D_{out} & c \end{bmatrix} \tag{5}$$

where c is the distance from H_i to itself. Here we arrange H_i the last host in N_b for explanation convenience but the hosts in N_b can be arranged in sequence with their ID numbers.

Extract Nonnegative Low Rank Part: Similar to recovering of A_a, we intend to recover nonnegative low rank part A_b from Q_b by RUN-PACE again, A_b = RUN-PACE(Q_b, d, λ). The result low rank part A_b is our expect proper distances between hosts in N_b. Once it's available, we can find the H_i's coordinates by match according distances in it. The problem can be solved by the RUN-PACE algorithm either.

Find normal host's Coordinates: The result A_b in last step is the precise approximation of proper distance matrix of hosts in N_b. The distance vector $A_b(N+1, 1:N)$ and $A_b(1:N, N+1)$ are the according proper distances to and from its reference hosts, which eliminate the impact of errors and outliers. Here $A_b(N+1, 1:N)$ and $A_b(1:N, N+1)$ indicate the last row (except entry $A_b(N+1, N+1)$) of A_b and last column (except entry $A_b(N+1, N+1)$) of A_b respectively.

As the basic idea of matrix factorization model, a normal host predicts distance to and from its reference hosts by the product of its outgoing and incoming vector of itself and incoming and outgoing vector of its reference hosts. Therefore, based on the existed coordinates of reference hosts, we find H_i's accurate coordinates by matching proper distance vector $A_b(N+1, 1:N)$ and $A_b(1:N, N+1)$. Specifically, we can decide normal host's coordinates using following optimization problem in Eq. (6).

$$\begin{aligned} X_i &= \arg\min_{X \in \mathbb{R}_+^d} \|X \cdot V_r^T - A_b(N+1, 1:N)\|_2^2 \\ Y_i &= \arg\min_{Y \in \mathbb{R}_+^d} \|Y \cdot U_r^T - (A_b(1:N, N+1))^T\|_2^2 \end{aligned} \tag{6}$$

where $\|\cdot\|_2$ denotes the 2-norm of a vector and $X \in \mathbb{R}_+^d$ constrains each element of X to be greater than or equal to 0. The problem formulated in Eq. (6) can be solved using nonnegative least squares method [31].

Once the coordinates of normal host has been calculated, we can add it into the positioned host set N_c. The basic architecture limits the scalability of NCS which requires a normal host to measure network distances to all landmark hosts. Our design permits normal hosts choose the reference hosts in the positioned host set N_c, which improves the scalability and robustness of RNC. This design proposes the ordinary hosts which have already computed their vectors only can act as reference hosts. The reference host candidates can be landmark nodes, or other positioned normal hosts.

In this paper, the pursuit of our idea is to find proper distance between hosts actively, instead accept the given distances passively. Since the errors or outliers crawl in the measured distances, we propose to extract the proper distances from them. So, we propose to recover the nonnegative low rank matrix A_a and A_b from Q_a and Q_b using our RUN-PACE method. The A_a and A_b are our desired ''proper" distance matrices between hosts in N_a and N_b, which eliminates the influence of errors and outliers having been collected in E_a and E_b respectively.

Our RNC algorithm is specified in Algorithm 2. The APPR procedure is to approximate distance matrix Q_b. The procedure RUN-PACE is to extract nonnegative low rank part which is further explained in Algorithm 3. To solve Eq. (6), we adopt nonnegative least squares methods which are implemented by procedure NNLS.

Algorithm 2 Normal host's coordinates determination using RNC

```
Input: $N_b, U_r, V_r, D_{out}, D_{in}, d, \lambda$.
1: $Q_b = \text{APPR}(U_r, V_r, D_{out}, D_{in})$
2: $A_b = \text{RUN-PACE}(Q_b, d, \lambda)$.
3: $X_i = \text{NNLS}(A_b(N+1, 1:N), V_r)$;
4: $Y_i = \text{NNLS}(A_b(1:N, N+1), U_r)$.
5: $N_c = N_c \bigcup \{H_i\}$
Output: $(X_i, Y_i, N_c)$.
```

V. Robust Nonnegative Principal Component Analysis

In this section, we present the RUN-PACE method used to recover the nonnegative low rank matrix A from $Q = A + E$. In short, recovering the nonnegative low rank part from D focuses on the convex optimization problem formulated in Eq. (7):

$$\begin{aligned} \min_{A,E} \quad & \|A\|_* + \lambda\|E\|_1, \\ \text{subject to} \quad & Q = A + E, \\ & A \geq 0 \end{aligned} \tag{7}$$

Compared to the classic problem in Eq. (1), Eq. (7) attempts to recover the low-rank part from Q under the nonnegative inequality constraints of low rank part. However, the original problem cannot be solved by ADM due to the existence of inequality constraints. In our research, we propose to import a slack variable B and transform the inequality constraints on B by enforcing $A = B$ and $B \geq 0$. Then we utilize two equation constraints $Q = A + E$ and $A = B$ to introduce two multipliers W and Z. And ADM performs minimization with respect to A, E and B alternatively followed by the update of W and Z. Through the ADM, the RUN-PACE method can recover the nonnegative low rank part A from matrix Q.

Introduce slack variable B: Since it is hard to optimize A under low rank and nonnegative constraints concurrently in Eq. (7), we first introduce slack variable B to relax constraints of $A \geq 0$. After that, we can utilize ADM to solve RPCA problem under nonnegative conditions of A. Specifically, we transform the original problem to the problem formulated in Eq. (8):

$$\begin{aligned} \min_{A,E,B} \quad & \|A\|_* + \lambda\|E\|_1, \\ \text{subject to} \quad & Q = A + E, \\ & A = B, \\ & B \geq 0 \end{aligned} \tag{8}$$

Solve the problem with ADM: By introducing Lagrangian multiplier W and Z, the augmented Lagrangian function $f(\cdot)$ of Eq. (8) is formulated in Eq. (9):

$$\begin{aligned} \min_{A,E,B} f(A,E,B,W,Z,\mu,\nu) = & \|A\|_* + \lambda\|E\|_1 + \langle W, Q-A-E\rangle \\ & + \frac{\mu}{2}\|Q-A-E\|_F^2 + \langle Z, A-B\rangle + \frac{\nu}{2}\|A-B\|_F^2 \\ \text{subject to } & B \geq 0 \end{aligned} \tag{9}$$

where $\mu > 0$ and $\nu > 0$ are the penalty parameters and $\langle\cdot\rangle$ is the Frobenius inner product of two matrices which is defined as the sum of the products of the corresponding components of two matrices having the same size.

The augmented Lagrangian function in Eq. (9) only incorporate the equality constraint $Q = A + E$ and $A = B$. The bound constraints $B \geq 0$ are incorporated explicitly in the subproblem. In this way, we can derive the solution of Eq. (8) by minimizing augmented Lagrangian function $f(\cdot)$ with inequality constraints.

The ADM for Eq. (8) is derived by successively minimization $f(\cdot)$ with respect to E, A, B, one at a time while fixing others at their most recent values. Note that the multiplier estimations W, Z and penalty parameters μ, ν are fixed during each iteration. With the fixed W, Z, μ and ν, the ADM update E, A, B by minimizing subproblem Eq. (9) alternatively. We will introduce the minimization of E, A, B sequently in following paragraphs.

Update error and outlier part E: The updating of E in Eq. (9) is to find the optimal solution of Eq. (10):

$$\min_E \quad \lambda\|E\|_1 + \langle W, Q-A-E\rangle + \frac{\mu}{2}\|Q-A-E\|_F^2 \tag{10}$$

For convenience, we introduce the soft-thresholding (shrinkage) operator $s_\tau(\cdot)$ in Eq. (11):

$$s_\tau(x) = \begin{cases} x - \tau & \text{if } x > \tau \\ x + \tau & \text{if } x < -\tau \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

where $x \in \mathbb{R}$ and $\tau > 0$. This operator can be extended to vectors and matrices by applying it element-wise.

Specifically, the updating of E can follow the result described in [32]. We formulate the result in Eq. (12):

$$s_\tau(C) = \arg\min_E \ \tau\|E\|_1 + \frac{1}{2}\|E - C\|_F^2 \tag{12}$$

where C is a matrix and optimal solving $\widehat{E} = s_\tau(C)$ is a matrix whose each element is calculated with soft-thresholding operator: $\widehat{E}(i_{row}, j_{col}) = s_\tau(C(i_{row}, j_{col}))$. Following the optimization formula, the optimal solving $\widehat{E}$ of Eq. (10) is:

$$\widehat{E} = s_{\lambda\mu^{-1}}(Q - A + \mu^{-1}W) \tag{13}$$

Update Low Rank Part A: This section is to solve A in Eq. (9) with E, B, W, Z, μ and ν fixed. We can formulate this optimization problem in Eq. (14):

$$\begin{aligned} \min_A \;\; & \|A\|_* + \langle W, Q-A-E\rangle + \frac{\mu}{2}\|Q-A-E\|_F^2 \\ & + \langle Z, A-B\rangle + \frac{\nu}{2}\|A-B\|_F^2 \end{aligned} \tag{14}$$

By rearranging the Eq. (14), we transform it to following equivalent problem:

$$\begin{aligned} & \min_A \;\; (\mu+\nu)^{-1}\|A\|_* + \frac{1}{2}\|A-R\|_F^2 \\ & R = (\mu+\nu)^{-1}\left[\mu(Q - E + \mu^{-1}W) + \nu(B - \nu^{-1}Z)\right] \end{aligned} \tag{15}$$

To solve Eq. (15), let's recall the analysis in [33] with Eq. (16)

$$U_C s_\tau(S_C) V_C^T = \arg\min_A \;\; \tau\|A\|_* + \frac{1}{2}\|A - C\|_F^2 \tag{16}$$

where $U_C S_C V_C^T$ is the SVD of C and $s_\tau(\cdot)$ is the soft-thresholding operator. Following the theorem, we can figure out the solving of A in Eq. (17):

$$\widehat{A} = U_R s_{(\mu+\nu)^{-1}}(S_R) V_R^T \tag{17}$$

where $U_R S_R V_R^T$ is the SVD of R.

Update Nonnegative Slack Variable B: The updating of B in Eq. (9) is to solve Eq. (18):

$$\begin{aligned} \min_B \;\; & \langle Z, A-B\rangle + \frac{\nu}{2}\|A-B\|_F^2 \\ \text{subject to} \;\; & B \geq 0 \end{aligned} \tag{18}$$

It's well known that the optimal solving of B is:

$$\widehat{B} = q_+(A + \nu^{-1}Z) \tag{19}$$

Here $q_+(\cdot)$ is a operator which satisfies $q_+(x) = max\{x, 0\}$ where $x \in \mathbb{R}$. By applying the operator element-wise, it can be extended to matrices which satisfies $(q_+(C))(i_{row}, j_{col}) = max\{C(i_{row}, j_{col}), 0\}$, where C is a matrix.

Algorithm 3 RUN-PACE using the ADM

```
Input: Target matrix $Q \in \mathbb{R}^{N_1 \times N_2}, d, \lambda$.
 1: procedure RUN-PACE($Q, d, \lambda$)
 2:     $W_0 = Q/h(Q); Z_0 = 0; A_0 = B_0 = 0;$
 3:     $\mu_0 = \nu_0 > 0; \rho > 1; k = 0.$
 4:     while not converged do
 5:         // Line 6 solves $E_{k+1} = \text{argmin}_E f(A_k, E, W_k, \mu_k)$.
 6:         $E_{k+1} = s_{\lambda\mu_k^{-1}}(Q - A_k + \mu_k^{-1}W_k)$.
 7:         // Line 9-10 solve
 8:         // $A_{k+1} = \text{argmin}_A f(A, B_k, E_{k+1}, W_k, Z_k, \mu_k, \nu_k)$
 9:         $(U_R, S_R, V_R) = \text{SVD}(R|_{B_k, E_{k+1}, W_k, Z_k, \mu_k, \nu_k})$;
10:         $A_{k+1} = U_R s_{(\mu_k+\nu_k)^{-1}}(S_R) V_R^T$.
11:         //Line 12 solves $B_{k+1} = \text{argmin}_B f(A_{k+1}, B, Z_k, \nu_k)$.
12:         $B_{k+1} = q_+(A_{k+1} + \nu_k^{-1} Z_k)$.
13:         // Line 14 updates $W_{k+1}$
14:         $W_{k+1} = W_k + \mu_k(Q - A_{k+1} - E_{k+1})$.
15:         // Line 16 updates $Z_{k+1}$
16:         $Z_{k+1} = Z_k + \nu_k(A_{k+1} - B_{k+1})$.
17:         // Line 18 updates $\mu_{k+1}$ and $\nu_{k+1}$
18:         $\mu_{k+1} = \rho\mu_k$ ; $\nu_{k+1} = \rho\nu_k$.
19:         $k = k + 1$.
20:     end while
21: end procedure
Output: $B_k$.
```

In summary, the alternative direction method in our research update E, A and B iteratively. In detail, it updates E, A and B sequentially by minimizing subproblem Eq. (9) and keeping μ and ν fixed. It updates the E, A and B alternatively, that is, one at a time while fixing the other at its most recent value. The solving of E can follow the result in Eq. (13). The updating of A is following the formulation in Eq. (17). We update B according to the result in Eq. (19). Then the violation amount of the constraints $Q = A + E$ and $A = B$ are used to update Lagrangian multiplier W and Z, respectively. And the penalty parameter μ and ν are increased by a factor of $\rho > 1$. Finally, it outputs nonnegative matrix B_k as the optimal value of A.

In our research, we use $A_0 = B_0 = 0$ as the initial values of A and B. Furthermore, we use $W_0 = Q/h(Q)$ as the initial value of W where $h(Q)$ is defined in Eq. (20):

$$h(Q) = \max\{\|Q\|_2, \lambda^{-1}\|Q\|_\infty\} \tag{20}$$

Here $\|\cdot\|_2$ and $\|\cdot\|_\infty$ are 2-norm and infinity norm of a matrix. And we use $Z_0 = 0$ as the initial value of another Lagrangian multiplier Z.

In addition, we set $\lambda = 1/\sqrt{\max\{N_1, N_2\}}$ as the weight parameter of E, where N_1 and N_2 are the scale of Q. And we set penalty parameters $\mu_0 = \nu_0 = 1.25/\|Q\|_2$ which are increased by $\rho = 1.5$. The stopping criteria is $\|Q - A_k - E_k\|_F/\|Q\|_F < 10^{-4}$ and $\|A_k - B_k\|_F/\|A_k\|_F < 10^{-4}$. The maximal iteration times is set to 100. The SVD function presents the singular value decomposition operation upon a matrix. Since an efficient SVD operation often needs the specified rank, we set the rank as the dimension of coordinates, d, intuitively. The algorithm is outlined in Algorithm 3.

Through the entire RUN-PACE algorithm, we transform original problem in the problem which can be solved by ADM by introducing a slack variable B. It results in precise solution with low computation overhead. Potentially, it guarantees the nonnegative of distance estimations concurrently.

VI. Simulation

In this section, we compare the performance of RNC with the state-of-the-art NCS schemes: Vivaldi [24], IDES [8], DMF [14], Phoenix [13] and DMFSGD [15]. In our research, we implemented RNC, Vivaldi and IDES using Matlab. In addition,we adopted the Matlab implementation of DMF, Phoenix and DMFSGD from their authors. Our simulations were executed on a 64-bit Windows 7 machine with Intel Core i7-3770 CPU, 16GB memory, and 8MB cache.

Since the errors of predicted distance matrix using Euclidean distance model are normally greater than that of matrix factorization model, we only considered one method based on the Euclidean distance model, Vivaldi. Except that, other schemes are all based on matrix factorization model. DMFSGD supports both Euclidean distance model and matrix factorization model. In our research, we adopt the implementation of DMFSGD based on matrix factorization model. IDES can use both NMF and SVD technologies. We use the implementation of IDES that is based on NMF which guarantees the nonnegativity of prediction distances. Note that only the DMF scheme may result in negative predicted distances and all other schemes only work with nonnegative predicted distances.

Among all schemes, IDES and RNC need virtual landmark hosts which are assigned a coordinates and other hosts in the network calculate their coordinates according to the position of landmark hosts. The selection of landmark hosts in IDES and RNC is totally random and we always select 32 landmark hosts for all data sets. For the other schemes, we randomly select $K = 32$ reference hosts for each host. Potentially, IDES and RNC do not require a convergence process. We run IDES and RNC 10 times randomly. Except IDES and RNC, other schemes need to converge to their optimal position in a period of time slots. For guaranteeing sufficient convergence and removing random impact, we evaluate other schemes through 10 time slots with 10 random runs. Unless otherwise stated, we observe their performance at the 10-th time slot. Without exception, all the schemes generate $d = 8$ dimension coordinates for every single host.

We use three real-world data sets in our simulations. The network distances in the data set are RTTs between pairs of Internet hosts. The detail of the data sets are scribed following:

- PL data set [19] includes the RTTs among 169 PlanetLab hosts all over the world. The data are the minimum RTTs measured at 3/23/2004 0:00 EST.
- P2PSim data set [20] is measuring the latencies between 1740 DNS servers using King method [34].
- Meridian data set [21] performs host to host latency measurement between 2500 hosts between 5/13/2004 using King method [34] technology as well.

To characterize the performance, we consider the following metrics:

- Median Relative Error (MRE) is defined as the median value of relative error. The definition of Relative error is defined as $RE = (|D(i,j) - \widehat{D}(i,j)|)/D(i,j)$.
- Cumulative Distribution Function (CDF) of the relative errors.
- 90th Percentile Relative Error (NPRE) is defined as the smallest relative error whose CDF value is greater than 0.9.

A. Prediction Accuracy

Fig. 2 illustrates the CDF of relative errors for RNC and other schemes in different data sets. First of all, the RNC outperforms other schemes in all data sets. Take results in Fig. 4(a) for example, at least 90% relative errors, i.e., NPRE, are less than 0.502 using RNC. The median value of relative error, i.e., MRE, is only 0.087. Compare to Phoenix, which is the most closest approach on performance, has 0.523 of NRPE and 0.1 of MRE. In Fig. 4(b) and 4(c), RNC is more close to Phoenix but still outperforms other schemes significantly.

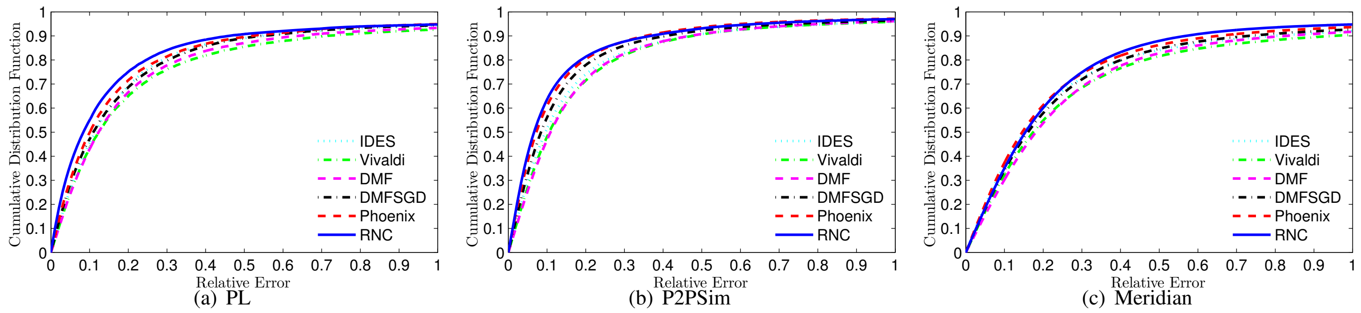
Secondly, the prediction accuracy is worse for all schemes in PL data set than in P2PSim and Meridian data sets. And RNC outperforms other schemes with more advantage in PL data set. The worse prediction in the PL data set may be due to the fact that distances in PL are not symmetric, which greatly increase the difficulty of prediction. However, our scheme continue to be the most accuracy scheme among all schemes. On the contrary, distances in terms of RTTs in P2PSim and Meridian data sets are symmetric between two end hosts.

B. Efficiency

Table I illustrates the running time of different schemes. Except IDES and RNC, the other schemes need a convergence proceeding of 10 time slots. Therefore, the running time of Vivaldi, DMF, DMFSGD and Phoenix are sum in 10 time slots which is in accord with the performance comparison. The running time of these schemes can be divided to three groups. The fastest group consists of IDES and DMF which mainly utilize nonnegative least squares algorithm and offer imprecise coordinates. The slower group includes Vivaldi, Phoenix and RNC, in which elementary arithmetic, nonnegative least square and SVD calculation dominate the computation time respectively and offer better accuracy. The slowest group is DMFSGD which involves matrix multiplications. In spite of the fast computations for IDES and DMF, they only offer much poorer prediction accuracy compared to RNC. Though Vivaldi pay similar computation overhead, they can not provide small prediction error like RNC. Phoenix behaves a little bit efficient than RNC and relatively close performance, but Phoenix need a convergence proceeding which results in more delay and traffic. Compared to DMFSGD, RNC can provide more precise prediction coordinates with much faster pace.

C. Scalability

To investigate the performance of RNC in practice, we compare RNC with IDES in the scenario where landmark hosts



(a) PL (b) P2PSim (c) Meridian

Fig. 2. Comparison of CDF for Vivaldi, IDES, DMF, DMFSGD, Phoenix and RNC

TABLE I
COMPUTATION TIME (MILLISECONDS)

Schemes	IDES	Vivaldi	DMF	DMFSGD	Phoenix	RNC
PL	0.45	10.37	0.35	71.11	6.55	12.32
P2PSim	0.88	12.95	0.42	62.74	6.89	12.39
Meridian	1.05	14.28	0.46	60.07	6.87	12.11

are partially failure. It results in that distance measurements to and from a portion of landmark hosts are unavailable. However, it reduces the communication overhead and allows the NCS support more hosts concurrently. Furthermore, it makes the system more robust while landmark hosts are partially failing.

In Fig. 3, we investigate the performance variation of IDES and RNC along with the fraction of unobserved landmark hosts increases. The x axis denotes the fraction of unobserved landmark hosts. The y axis indicates the MRE property. The unobserved reference hosts for each normal host are randomly selected from 32 reference hosts at independence. From this figure, we can see that RNC always outperforms IDES in variable fraction of unobserved landmarks. Both schemes' performance degrades along with the unobserved fraction increases.

D. Robustness

To further demonstrate the performance of RNC in practice, we also compare the MRE property of RNC with IDES while distance measurements contain errors. Toward this end, we add errors artificially in the accurate distance measurements and compare the performance of different schemes. In detail, we randomly select a ratio of p entries among all the $M \times (M-1)$ distances between M hosts in the network. Then we increase these selected distances to λ times of the original values. Then we observe the performance of different schemes at $\lambda = 3$ and $\lambda = 10$ with ratio p increasing from 0% to 5%. We repeat the simulation at 100 random runs and investigate the average median relative errors.

From Fig. 4, we can see that RNC outperforms IDES significantly. Even when $p = 5\%$ for RNC with both $\lambda = 3$ and $\lambda = 10$ in all traces, RNC is still more accurate than IDES at $p = 0\%$, i.e., the case of distance measurements containing no any errors. That means RNC is much more robust than IDES to distance measurement errors. The reason may stem from the idea of removing the errors and outliers in the distance measurement for RNC, which always extract the low rank part to determine the coordinates.

VII. CONCLUSIONS

In this paper, we present RNC, an innovative scheme for network coordinate system based on the assumption that distance measurement results in the network include errors or outliers. The assigned coordinates should be consistent with the distances after the impact of the errors or outliers is eliminated. First of all, a few landmark hosts are chosen randomly and their coordinates are determined using our scheme. Then other normal hosts determine their coordinates using our scheme by choosing a small amount of reference hosts and measuring the distances to and from them. In both of these steps, a local target distance matrix based on only a few measured distances is constructed and the RUNPACE method is applied to the target distance matrix to extract the nonnegative low-rank component. Then, the low-rank component is used to determine the coordinates. By eliminating the impact of the errors or outliers, each individual node can calculate its coordinates precisely. Through extensive simulations, we found that RNC outperforms the state-of-the-art NCS schemes in terms of prediction error and computation efficiency. In addition, RNC leads to a system that is more scalable to landmark host failures and more robust to distance measurement errors.

REFERENCES

[1] Azureus bittorrent. http://sourceforge.net/projects/azureus/.

[2] Sharad Agarwal and Jacob R Lorch. Matchmaking for online games and other latency-sensitive p2p systems. In *ACM SIGCOMM Computer Communication Review*, volume 39, pages 315–326. ACM, 2009.

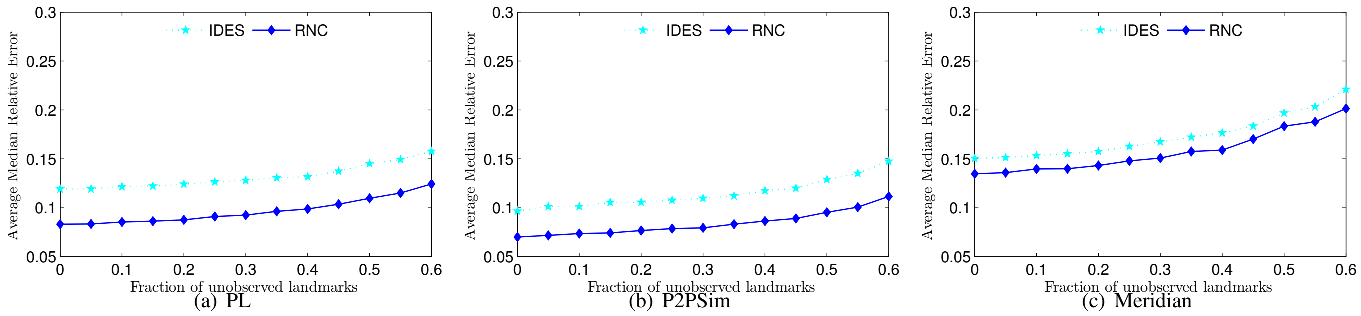
[3] Martin Mauve, J. Widmer, and Hannes Hartenstein. A survey on position-based routing in mobile ad hoc networks. *Network, IEEE*, 15(6):30–39, 2001.

[4] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. pages 161–172, 2001.

[5] Rupa Krishnan, Harsha V. Madhyastha, Sridhar Srinivasan, Sushant Jain, Arvind Krishnamurthy, Thomas Anderson, and Jie Gao. Moving beyond end-to-end path information to optimize cdn performance. In *Proceedings of ACM SIGCOMM Internet Measurement Conference*, pages 190–201. ACM, 2009.

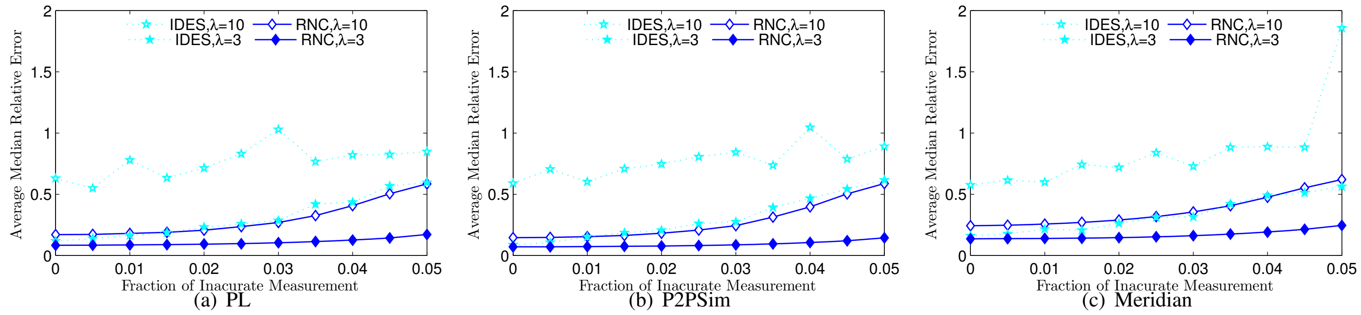
[6] TS Eugene Ng and Hui Zhang. Predicting internet network distance with coordinates-based approaches. In *Proceedings of IEEE INFOCOM*, volume 1, pages 170–179. IEEE, April 2002.

[7] Yun Mao and Lawrence K. Saul. Modeling distances in large-scale networks by matrix factorization. In *Proceedings of the ACM SIGCOMM conference on Internet measurement*, pages 278–287. ACM, October 2004.



(a) PL (b) P2PSim (c) Meridian

Fig. 3. The impact of landmark failures on the accuracy of IDES and RNC



(a) PL (b) P2PSim (c) Meridian

Fig. 4. The impact of measurement errors on the accuracy of IDES and RNC

[8] Yun Mao, Lawrence K Saul, and Jonathan M Smith. Ides: An internet distance estimation service for large networks. *IEEE Journal on Selected Areas in Communications*, 24(12):2273–2284, December 2006.

[9] Patrick Wendell, Joe Wenjie Jiang, Michael J Freedman, and Jennifer Rexford. Donar: decentralized server selection for cloud services. In *ACM SIGCOMM Computer Communication Review*, volume 40, pages 231–242. ACM, 2010.

[10] Cong Ding, Yang Chen, Tianyin Xu, and Xiaoming Fu. Cloudgps: a scalable and isp-friendly server selection scheme in cloud computing environments. In *Proceedings of the IEEE International Workshop on Quality of Service (IWQoS)*, page 5. IEEE, 2012.

[11] Xiaohan Zhao, Alessandra Sala, Christo Wilson, Haitao Zheng, and Ben Y Zhao. Orion: shortest path estimation for large social graphs. In *Proceedings of the Third USENIX Workshop on Online Social Networks*, page 5, 2010.

[12] Xiaohan Zhao, Alessandra Sala, Haitao Zheng, and Ben Y Zhao. Efficient shortest paths on massive social graphs. In *Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, pages 77–86. IEEE, 2011.

[13] Yang Chen, Xiao Wang, Cong Shi, Eng Keong Lua, Xiaoming Fu, Beixing Deng, and Xing Li. Phoenix: A weight-based network coordinate system using matrix factorization. *IEEE Transactions on Network and Service Management*, 8(4):334–347, 2011.

[14] Yongjun Liao, Pierre Geurts, and Guy Leduc. Network distance prediction based on decentralized matrix factorization. In *9th International IFIP TC 6 Networking Conference*, pages 15–26. Springer, May 2010.

[15] Yongjun Liao, Wei Du, Pierre Geurts, and Guy Leduc. Dmfsgd: A decentralized matrix factorization algorithm for network distance prediction. *IEEE/ACM Transactions on Networking*, 21(5):1511–1524, December 2012.

[16] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization.

[17] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11, 2011.

[18] John Wright, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems*, pages 2080–2088, 2009.

[19] Jeremy Stribling. "all pairs of ping data for planetlab", http://www.pdos.lcs.mit.edu/ strib/pl_app.

[20] The P2PSim project. http://pdos.csail.mit.edu/p2psim/.

[21] The Meridian data. http://www.cs.cornell.edu/people/egs/meridian/data.php.

[22] Hyuk Lim, Jennifer C Hou, and Chong-Ho Choi. Constructing internet coordinate system based on delay measurement. In *Proceedings of the ACM SIGCOMM conference on Internet measurement*, pages 129–142. ACM, 2003.

[23] Manuel Costa, Miguel Castro, R Rowstron, and Peter Key. Pic: Practical internet coordinates for distance estimation. In *Proceedings of International Conference on Distributed Computing Systems*, pages 178–187. IEEE, July 2004.

[24] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 15–26. ACM, August 2004.

[25] John Wright, Arvind Ganesh, Shankar Rao, and Yi Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization, 2009. submitted to Journal of the ACM.

[26] Zhouchen Lin, Minming Chen, and Yi Ma. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *UIUC Technical Report UILU-ENG-09-2214*, October 2010.

[27] Xiaoming Yuan and Junfeng Yang. Sparse and low-rank matrix decomposition via alternating direction methods. *Pacific Journal of Optimization*, 9(1):167–180, 2013.

[28] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer verlag, 1999.

[29] A.R. Conn, N.I.M. Gould, and P.L. Toint. *LANCELOT: a FORTRAN package for large-scale nonlinear optimization (Release A), Vol. 17 in Springer Series in Computational Mathematics*. Springer-Verlag, December 1992.

[30] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[31] Charles L. Lawson and Richard J. Hanson. *Solving least squares problems*. Prentice-Hall, 1974.

[32] Elaine T. Hale, Wotao Yin, and Yin Zhang. Fixed-point continuation for ℓ_1-minimization: Methodology and convergence. *SIAM Journal on Optimization*, 19(3):1107–1130, 2008.

[33] Jian-Feng Cai, Emmanuel J. Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[34] Krishna P Gummadi, Stefan Saroiu, and Steven D Gribble. King: Estimating latency between arbitrary internet end hosts. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, pages 5–18. ACM, November 2002.