

# Modeling Distances in Large-Scale Networks by Matrix Factorization

Yun Mao

Department of Computer and Information Science  
University of Pennsylvania  
maoy@cis.upenn.edu

Lawrence K. Saul

Department of Computer and Information Science  
University of Pennsylvania  
lsaul@cis.upenn.edu

## ABSTRACT

In this paper, we propose a model for representing and predicting distances in large-scale networks by *matrix factorization*. The model is useful for network distance sensitive applications, such as content distribution networks, topology-aware overlays, and server selections. Our approach overcomes several limitations of previous coordinates-based mechanisms, which cannot model sub-optimal routing or asymmetric routing policies. We describe two algorithms—singular value decomposition (SVD) and nonnegative matrix factorization (NMF)—for representing a matrix of network distances as the product of two smaller matrices. With such a representation, we build a scalable system—*Internet Distance Estimation Service (IDES)*—that predicts large numbers of network distances from limited numbers of measurements. Extensive simulations on real-world data sets show that IDES leads to more accurate, efficient and robust predictions of latencies in large-scale networks than previous approaches.

## Categories and Subject Descriptors

C.2.1 [Computer-Communication Networks]: Network Architecture and Design—*Network topology*; C.2.4 [Computer-Communication Networks]: Distributed Systems

## General Terms

Algorithms, Measurement, Performance

## Keywords

Network distance, Matrix factorization

## 1. INTRODUCTION

Wide-area distributed applications have evolved considerably beyond the traditional client-server model, in which a client only communicates with a single server. In content distribution networks (CDN), peer-to-peer distributed

hash tables (DHT) [16, 17, 18, 22], and overlay routing [2], nodes often have the flexibility to choose their communication peers. This flexibility can greatly improve performance if relevant network distances are known. For example, in a CDN, an optimized client can download Web objects from the particular mirror site to which it has the highest bandwidth. Likewise, in DHT construction, a peer can route lookup requests to the peer (among those that are closer to the target in the virtual overlay network) with the lowest latency in the IP underlay network.

Unfortunately, knowledge of network distances is not available without cost. On-demand network measurements are expensive and time-consuming, especially when the number of possible communication peers is large. Thus, a highly promising approach is to construct a model that can *predict* unknown network distances from a set of partially observed measurements [4, 6, 7, 12, 13, 20].

Many previously proposed models are based on the embedding of host positions in a low dimensional space, with network distances estimated by Euclidean distances. Such models, however, share certain limitations. In particular, they cannot represent networks with complex routing policies, such as sub-optimal routing<sup>1</sup> or asymmetric routing, since Euclidean distances satisfy the triangle inequality and are inherently symmetric. On the Internet, routing schemes of this nature are quite common [3, 10, 15], and models that do not take them into account yield inaccurate predictions of network distances.

In this paper, we propose a model based on *matrix factorization* for representing and predicting distances in large-scale networks. The essential idea is to approximate a large matrix whose elements represent pairwise distances by the product of two smaller matrices. Such a model can be viewed as a form of dimensionality reduction. Models based on matrix factorization do not suffer from the limitations of previous work: in particular, they can represent distances that violate the triangle inequality, as well as asymmetric distances. Two algorithms—singular value decomposition (SVD) and nonnegative matrix factorization (NMF)—are presented for learning models of this form. We evaluate the advantages and disadvantages of each algorithm for learning compact models of network distances.

The rest of the paper is organized as follows. Section 2

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IMC'04, October 25–27, 2004, Taormina, Sicily, Italy.  
Copyright 2004 ACM 1-58113-821-0/04/0010 ...\$5.00.

<sup>1</sup>With sub-optimal routing policies, the network distance between two end hosts does not necessarily represent the shortest path in the network. Such routing policies exist widely in the Internet for various technical, political and economic reasons.

reviews previous work based on the low dimensional embedding of host positions in Euclidean space. Section 3 presents the model for matrix factorization of network distances. The SVD and NMF algorithms for learning these models from network measurements are presented and evaluated in section 4. Section 5 proposes an architecture to estimate distances required by an arbitrary host from low dimensional reconstructions. The architecture is evaluated in section 6. Finally, section 7 summarizes the paper.

## 2. NETWORK EMBEDDINGS

One way to predict network distance between arbitrary Internet end hosts is to assign each host a “position” in a finite-dimensional vector space. This can be done at the cost of a limited number of network measurements to a set of well-positioned infrastructure nodes<sup>2</sup>, or other peer nodes. In such a model, a pair of hosts can estimate the network distance between them by applying a distance function to their positions, without direct network measurement. Most previous work on these models has represented the host positions by coordinates in Euclidean space and adopted Euclidean distance as the distance function.

We define the problem formally as follows. Suppose there are  $N$  hosts  $\mathcal{H} = \{\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_N\}$  in the network. The pairwise network distance matrix is a  $N \times N$  matrix  $D$ , such that  $D_{ij} \geq 0$  is the network distance from  $\mathcal{H}_i$  to  $\mathcal{H}_j$ .

A network *embedding* is a mapping  $H : \mathcal{H} \rightarrow \mathbb{R}^d$  such that

$$D_{ij} \approx \hat{D}_{ij} = \|H(\mathcal{H}_i) - H(\mathcal{H}_j)\|, \forall i, j = 1, \dots, N \quad (1)$$

where  $\hat{D}_{ij}$  is the estimated network distance from  $\mathcal{H}_i$  to  $\mathcal{H}_j$  and  $H(\mathcal{H}_i)$  is the position coordinate of  $\mathcal{H}_i$  as a  $d$ -dimensional real vector. We simplify the coordinate notation from  $H(\mathcal{H}_i)$  to  $\vec{H}_i = (H_{i1}, H_{i2}, \dots, H_{id})$ . The network distance between two hosts  $\mathcal{H}_i$  and  $\mathcal{H}_j$  is estimated by the Euclidean distance of their coordinates:

$$\hat{D}_{ij} = \|\vec{H}_i - \vec{H}_j\| = \left( \sum_{k=1}^d (H_{ik} - H_{jk})^2 \right)^{\frac{1}{2}} \quad (2)$$

The main problem in constructing a network embedding is to compute the position vectors  $\vec{H}_i$  for all hosts  $\mathcal{H}_i$  from a partially observed distance matrix  $D$ . A number of learning algorithms have been proposed to solve this problem, which we describe in the next section.

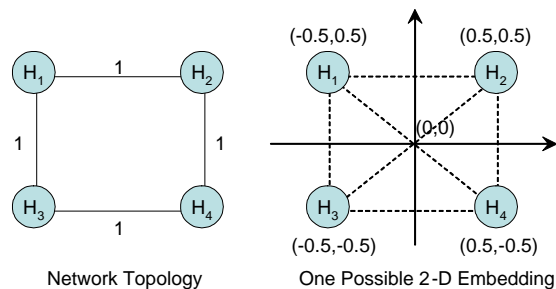
### 2.1 Previous work

The first work in the network embedding area was done by Ng and Zhang [13], whose Global Network Positioning (GNP) System embedded network hosts in a low-dimensional Euclidean space. Many algorithms were subsequently proposed to calculate the coordinates of network hosts. GNP uses a Simplex Downhill method to minimize the sum of relative errors:

$$\text{total\_err} = \sum_i \sum_j \frac{|D_{ij} - \hat{D}_{ij}|}{D_{ij}} \quad (3)$$

The drawback of GNP is that the Simplex Downhill method converges slowly, and the final results depend on the initial values of the search. PIC [4] applies the same algorithm

<sup>2</sup>referred as *landmark nodes* in this paper. They are also called beacon nodes.



**Figure 1: Four hosts  $\mathcal{H}_1 - \mathcal{H}_4$  in a simple network topology**

to the sum of squared relative errors and studies security-related issues.

Cox, Dabek *et al.* proposed the Vivaldi algorithm [5, 6] based on an analogy to a network of physical springs. In this approach, the problem of minimizing the sum of errors is related to the problem of minimizing the potential energy of a spring system. Vivaldi has two main advantages: it is a distributed algorithm, and it does not require landmark nodes.

Lim *et al.* [12] and Tang *et al.* [20] independently proposed models based on Lipschitz embeddings and Principal Component Analysis (PCA). These models begin by embedding the hosts in an  $N$ -dimensional space, where the coordinates of the host  $\mathcal{H}_i$  are given by its distances  $(D_{i1}, \dots, D_{iN})$  to  $N$  landmark nodes. This so-called Lipschitz embedding has the property that hosts with similar distances to other hosts are located nearby in the  $N$ -dimensional space. To reduce the dimensionality, the host positions in this  $N$ -dimensional space are then projected into the  $d$ -dimensional subspace of maximum variance by PCA. A linear normalization is used to further calibrate the results, yielding the final host positions  $\vec{H}_i \in \mathbb{R}^d$ .

### 2.2 Limitations

Euclidean distances are inherently symmetric; they also satisfy the triangle inequality. Thus, in any network embedding,

$$\begin{aligned} \hat{D}_{ij} &= \hat{D}_{ji} & \forall i, j \\ \hat{D}_{ij} + \hat{D}_{jk} &\geq \hat{D}_{ik} & \forall i, j, k \end{aligned}$$

These two properties are inconsistent with observed network distances. On the Internet, studies indicate that as many as 40% of node pairs of real-world data sets have a shorter path through an alternate node [3, 20]. Another study shows that asymmetric routing is quite common [15]; even for the same link, the upstream and downstream capacities may be very different [10].

In addition to these limitations, low-dimensional embeddings of host positions cannot always model distances in networks where there are pairs of nodes that do not have a direct path between them, even if the distances are symmetric and satisfy triangle inequality. Figure 1 illustrates a simple network topology in which four hosts in different autonomous systems are connected with unit distance to their neighbors. An intuitive two-dimensional embedding is also shown. In the given embedding, the estimated distances are  $\hat{D}_{14} = \hat{D}_{23} = \sqrt{2}$ , but the real distances are  $D_{14} = D_{23} = 2$ .

It is provable that there exists no Euclidean space embedding (of any dimensionality) that can exactly reconstruct the distances in this network. Similar cases arise in networks with tree-like topologies.

### 3. DISTANCE MATRIX FACTORIZATION

The limitations of previous models lead us to consider a different framework for compactly representing network distances. Suppose that two nearby hosts have similar distances to all the other hosts in the network. In this case, their corresponding rows in the distance matrix will be nearly identical. More generally, there may be many rows in the distance matrix that are equal or nearly equal to linear combinations of other rows. Recall from linear algebra that an  $N \times N$  matrix whose rows are not linearly independent has rank strictly less than  $N$  and can be expressed as the product of two smaller matrices. With this in mind, we seek an approximate factorization of the distance matrix, given by:

$$D \approx XY^T,$$

where  $X$  and  $Y$  are  $N \times d$  matrices with  $d \ll N$ . From such a model, we can estimate the network distance from  $\mathcal{H}_i$  to  $\mathcal{H}_j$  by  $\hat{D}_{ij} = \vec{X}_i \cdot \vec{Y}_j$ , where  $\vec{X}_i$  is the  $i$ th row vector of the matrix  $X$  and  $\vec{Y}_j$  is the  $j$ th row vector of the matrix  $Y$ .

More formally, for a network with distance matrix  $D_{ij}$ , we define a *distance matrix factorization* as two mappings

$$\begin{aligned} X &: \mathcal{H} \rightarrow \mathbb{R}^d, \\ Y &: \mathcal{H} \rightarrow \mathbb{R}^d, \end{aligned}$$

and an approximate distance function computed by

$$\hat{D}_{ij} = X(\mathcal{H}_i) \cdot Y(\mathcal{H}_j).$$

As shorthand, we denote  $X(\mathcal{H}_i)$  as  $\vec{X}_i$  and  $Y(\mathcal{H}_i)$  as  $\vec{Y}_i$ , so that we can write the above distance computation as:

$$\hat{D}_{ij} = \vec{X}_i \cdot \vec{Y}_j = \sum_{k=1}^d X_{ik} Y_{jk}. \quad (4)$$

Note that in contrast to the model in section 2, which maps each host to one position vector, our model associates *two* vectors with each host. We call  $\vec{X}_i$  the *outgoing vector* and  $\vec{Y}_i$  the *incoming vector* for  $\mathcal{H}_i$ . The estimated distance from  $\mathcal{H}_i$  to  $\mathcal{H}_j$  is simply the dot product between the outgoing vector of  $\mathcal{H}_i$  and the incoming vector of  $\mathcal{H}_j$ .

Applying this model of network distances in distributed applications is straightforward. For example, consider the problem of mirror selection. To locate the closest server among several mirror candidates, a client can retrieve the outgoing vectors of the mirrors from a directory server, calculate the dot product of these outgoing vectors with its own incoming vector, and choose the mirror that yields the smallest estimate of network distance (i.e., the smallest dot product).

Our model for representing network distances by matrix factorization overcomes certain limitations of models based on low dimensional embeddings. In particular, it does not require that network distances are symmetric because in general  $\hat{D}_{ij} = \vec{X}_i \cdot \vec{Y}_j \neq \vec{X}_j \cdot \vec{Y}_i = \hat{D}_{ji}$ . Distances computed in this way also are not constrained to satisfy the triangle inequality. The main assumption of our model is that many rows in the distance matrix are linearly dependent, or nearly so. This is likely to occur whenever there are

clusters of nearby nodes in the network which have similar distances to distant nodes. In this case, the distance matrix  $D$  will be well approximated by the product of two smaller matrices.

### 4. DISTANCE RECONSTRUCTION

In this section we investigate how to estimate outgoing and incoming vectors  $\vec{X}_i$  and  $\vec{Y}_i$  for each host  $\mathcal{H}_i$  from the distance matrix  $D$ . We also examine the accuracy of models that approximate the true distance matrix by the product of two smaller matrices in this way.

The distance matrix  $D$  can be viewed<sup>3</sup> as storing  $N$  row-vectors in  $N$ -dimensional space. Factoring this matrix  $D \approx XY^T$  is essentially a problem in linear dimensionality reduction, where  $Y$  stores  $d$  basis vectors and  $X$  stores the linear coefficients that best reconstruct each row vector of  $D$ . We present two algorithms for matrix factorization that solve this problem in linear dimensionality reduction.

#### 4.1 Singular value decomposition

An  $N \times N$  distance matrix  $D$  can be factored into three matrices by its singular value decomposition (SVD), of the form:

$$D = USV^T,$$

where  $U$  and  $V$  are  $N \times N$  orthogonal matrices and  $S$  is an  $N \times N$  diagonal matrix with nonnegative elements (arranged in decreasing order). Let  $A = US^{\frac{1}{2}}$  and  $B = S^{\frac{1}{2}}V$ , where  $S_{ii}^{\frac{1}{2}} = \sqrt{S_{ii}}$ . It is easy to see that  $AB^T = US^{\frac{1}{2}}(VS^{\frac{1}{2}})^T = US^{\frac{1}{2}}S^{\frac{1}{2}}V^T = D$ . Thus SVD yields an exact factorization  $D = AB^T$ , where the matrices  $A$  and  $B$  are the same size as  $D$ .

We can also use SVD, however, to obtain an approximate factorization of the distance matrix into two smaller matrices. In particular, suppose that only a few of the diagonal elements of the matrix  $S$  are appreciable in magnitude. Define the  $N \times d$  matrices:

$$X_{ij} = U_{ij} \sqrt{S_{jj}}, \quad (5)$$

$$Y_{ij} = V_{ij} \sqrt{S_{jj}}, \quad (6)$$

where  $i = 1 \dots N$  and  $j = 1 \dots d$ . The product  $XY^T$  is a low-rank approximation to the distance matrix  $D$ ; if the distance matrix is itself of rank  $d$  or less, as indicated by  $S_{jj} = 0$  for  $j > d$ , then the approximation will in fact be exact. The low-rank approximation obtained from SVD can be viewed as minimizing the squared error function

$$\sum_i \sum_j (D_{ij} - \vec{X}_i \cdot \vec{Y}_j)^2 \quad (7)$$

with respect to  $X_i \in \mathbb{R}^d$  and  $Y_j \in \mathbb{R}^d$ . Eqs. (5) and (6) compute the *global minimum* of this error function.

Matrix factorization by SVD is related to principal component analysis (PCA) [9] on the row vectors. Principal

<sup>3</sup>Note that  $D$  does not have to be a square matrix of pairwise distances. It can be the distance matrix from one set of  $N$  hosts  $\mathcal{H}$  to another set of  $N'$  hosts  $\mathcal{H}'$ , which may or may not overlap with each other. In this case,  $X \in \mathbb{R}^{N \times d}$  contains the outgoing vectors for  $\mathcal{H}$  and  $Y \in \mathbb{R}^{d \times N'}$  contains the incoming vectors for  $\mathcal{H}'$ . For simplicity, though, we consider the case  $N = N'$  in what follows.

components of the row vectors are obtained from the orthogonal eigenvectors of their correlation matrix; each row vector can be expressed as a linear combination of these eigenvectors. The diagonal values of  $S$  measure the significance of the contribution from each principal component. In previous work on embedding of host positions by PCA, such as ICS [12] and Virtual Landmark [20], the first  $d$  rows of the matrix  $U$  were used as coordinates for the hosts, while discarding the information in the matrices  $S$  and  $V$ . By contrast, our approach uses  $U$ ,  $S$  and  $V$  to compute outgoing and incoming vectors for each host.

We use the topology in Figure 1 as an example to show how the algorithm works. The distance matrix is

$$D = \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 1 \\ 1 & 2 & 0 & 1 \\ 2 & 1 & 1 & 0 \end{bmatrix}$$

We obtain the SVD result as

$$U = \begin{bmatrix} -0.5 & 0 & \frac{1}{\sqrt{2}} & 0.5 \\ -0.5 & -\frac{1}{\sqrt{2}} & 0 & -0.5 \\ -0.5 & \frac{1}{\sqrt{2}} & 0 & -0.5 \\ -0.5 & 0 & -\frac{1}{\sqrt{2}} & 0.5 \end{bmatrix}, S = \begin{bmatrix} 4 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.5 & 0 & -\frac{1}{\sqrt{2}} & -0.5 \\ -0.5 & \frac{1}{\sqrt{2}} & 0 & 0.5 \\ -0.5 & -\frac{1}{\sqrt{2}} & 0 & 0.5 \\ -0.5 & 0 & \frac{1}{\sqrt{2}} & -0.5 \end{bmatrix}$$

Note that  $S_{44} = 0$ . Therefore, an exact  $d = 3$  factorization exists with:

$$X = \begin{bmatrix} -1 & 0 & 1 \\ -1 & -1 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & -1 \end{bmatrix}, Y = \begin{bmatrix} -1 & 0 & -1 \\ -1 & 1 & 0 \\ -1 & -1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

One can verify in this case that the reconstructed distance matrix  $XY^T$  is equal to the original distance matrix  $D$ .

## 4.2 Non-negative matrix factorization

Non-negative matrix factorization (NMF) [11] is another form of linear dimensionality reduction that can be applied to the distance matrix  $D_{ij}$ . The goal of NMF is to minimize the same error function as in Eq. (7), but subject to the constraint that  $X$  and  $Y$  are non-negative matrices. In contrast to SVD, NMF guarantees that the approximately reconstructed distances are nonnegative:  $\hat{D}_{ij} \geq 0$ . The error function for NMF can be minimized by an iterative algorithm. Compared to gradient descent and the Simplex Downhill method, however, the algorithm for NMF converges much faster and does not involve any heuristics, such as choosing a step size. The only constraint on the algorithm is that the true network distances must themselves be nonnegative,  $D_{ij} \geq 0$ ; this is generally true and holds for all the examples we consider. The algorithm takes as input initial (random) matrices  $X$  and  $Y$  and updates them in an alternating fashion. The update rules for each iteration are:

$$X_{ia} \leftarrow X_{ia} \frac{(DY)_{ia}}{(XY^TY)_{ia}}$$

$$Y_{ja} \leftarrow Y_{ja} \frac{(X^TD)_{aj}}{(X^TXY^T)_{aj}}$$

It is known that these update rules converge monotonically to stationary points of the error function, Eq. (7). Our experience shows that two hundred iterations suffice to converge to a local minimum.

One major advantage of NMF over SVD is that it is straightforward to modify NMF to handle missing entries in the distance matrix  $D$ . For various reasons, a small number of elements in  $D$  may be unavailable. SVD can proceed with missing values if we eliminate the rows and columns in  $D$  that contain them, but doing so will leave the corresponding host positions unknown.

NMF can cope with missing values if we slightly change the update rules. Suppose  $M$  is a binary matrix where  $M_{ij} = 1$  indicates  $D_{ij}$  is known and  $M_{ij} = 0$  indicates  $D_{ij}$  is missing. The modified update rules are:

$$X_{ia} \leftarrow X_{ia} \frac{\sum_k D_{ik} M_{ik} Y_{ka}}{\sum_k (XY^T)_{ik} M_{ik} Y_{ka}} \quad (8)$$

$$Y_{ja} \leftarrow Y_{ja} \frac{\sum_k (X^T)_{ak} D_{kj} M_{kj}}{\sum_k (X^T)_{ak} (XY^T)_{kj} M_{kj}} \quad (9)$$

These update rules converge to local minima of the error function,  $\sum_{ij} M_{ij} |D_{ij} - \hat{X}_i \cdot \hat{Y}_j|^2$ .

## 4.3 Evaluation

We evaluated the accuracy of network distance matrices modeled by SVD and NMF and compared the results to those of PCA from the Lipschitz embeddings used by Virtual Landmark [20] and ICS [12]. We did not evaluate the Simplex Downhill algorithm used in GNP because while its accuracy is not obviously better than Lipschitz embedding, it is much more expensive, requiring hours of computation on large data sets [20]. Accuracies were evaluated by the modified relative error,

$$\text{relative\_error} = \frac{|D_{ij} - \hat{D}_{ij}|}{\min(D_{ij}, \hat{D}_{ij})} \quad (10)$$

where the min-operation in the denominator serves to increase the penalty for underestimated network distances.

### 4.3.1 Data sets

We used the following five real-world data sets in simulation. Parts of the data sets were filtered out to eliminate missing elements in the distance matrices (since none of the algorithms except NMF can cope with missing data).

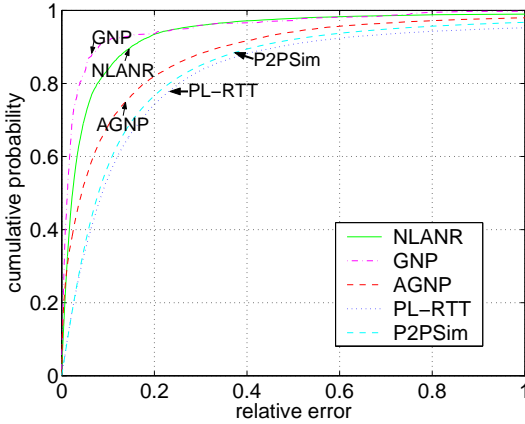
The network distances in the data sets are round-trip time (RTT) between pairs of Internet hosts. RTT is symmetric between two end hosts, but it does violate the triangle inequality and also give rise to other effects (described in Section 2.2) that are poorly modeled by network embeddings in Euclidean space.

- **NLANR**: The NLANR Active Measurement Project [1] collects a variety of measurements between all pairs of participating nodes. The nodes are mainly at NSF supported HPC sites, with about 10% outside the US. The data set we used was collected on January 30, 2003, consisting of measurements of a  $110 \times 110$  clique. Each host was pinged once per minute, and network distance was taken as the minimum of the ping times over the day.
- **GNP** and **AGNP**: The GNP project measured minimum round trip time between 19 active sites in May

2001. About half of the hosts are in North America; the rest are distributed globally. We used GNP to construct a symmetric  $19 \times 19$  data set and AGNP to construct an asymmetric  $869 \times 19$  dataset.

- **P2PSim**: The P2Psim project [14] measured a distance matrix of RTTs among about 2000 Internet DNS servers based on the King method [8]. The DNS servers were obtained from an Internet-scale Gnutella network trace.
- **PL-RTT**: Obtained from PlanetLab pairwise ping project [19]. We chose the minimum RTT measured at 3/23/2004 0:00 EST. A  $169 \times 169$  full distance matrix was obtained by filtering out missing values.

#### 4.3.2 Simulated Results



**Figure 2: Cumulative distribution of relative error by SVD over various data sets,  $d = 10$**

Figure 2 illustrates the cumulative density function (CDF) of relative errors of RTT reconstructed by SVD when  $d = 10$ , on 5 RTT data sets. The best result is over GNP data set: more than 90% distances are reconstructed within 9% relative error. This is not too surprising because the GNP data set only contains 19 nodes. However, SVD also works well over NLANR, which has more than 100 nodes: about 90% fraction of distances are reconstructed within 15% relative error. Over P2PSim and PL-RTT data sets, SVD achieves similar accuracy results: 90 percentile relative error is 50%. We ran the same tests on NMF and observed similar results. Therefore, we chose NLANR and P2PSim as two representative data sets for the remaining simulations.

Figure 3 compares the reconstruction accuracy of three algorithms: matrix factorization by SVD and NMF, and PCA applied to the Lipschitz embedding. The algorithms were simulated over NLANR and P2PSim data sets. It is shown that NMF has almost exactly the same median relative errors as SVD on both data sets when the dimension  $d < 10$ . Both NMF and SVD yield much more accurate results than Lipschitz: the median relative error of SVD and NMF is more than 5 times smaller than Lipschitz when  $d = 10$ . SVD is slightly better than NMF when  $d$  is large. The reason for this may be that the algorithm for NMF is only guaranteed to converge to local minima. Considering that the hosts in the data sets come from all over the

Internet, the results show that matrix factorization is a scalable approach to modeling distances in large-scale networks. In terms of maintaining a low-dimensional representation,  $d \approx 10$  appears to be a good tradeoff between complexity and accuracy for both SVD and NMF.

## 5. DISTANCE PREDICTION

The simulation results from the previous section demonstrate that pairwise distances in large-scale networks are well modeled by matrix factorization. In this section we present the *Internet Distance Estimation Service (IDES)* — a scalable and robust service based on matrix factorization to estimate network distances between arbitrary Internet hosts.

### 5.1 Basic architecture

We classify Internet hosts into two categories: landmark nodes and ordinary hosts. Landmark nodes are a set of well-positioned distributed hosts. The network distances between each of them is available to the *information server* of IDES. We assume that landmarks can measure network distances to others and report the results to the information server. The information server can also measure the pairwise distances via indirect methods without landmark support, *e.g.* by the King method [8] if the metric is RTT. An ordinary host is an arbitrary end node in the Internet, which is identified by a valid IP address.

Suppose there are  $m$  landmark nodes. The first step of IDES is to gather the  $m \times m$  pairwise distance matrix  $D$  on the information server. Then, we can apply either SVD or NMF algorithm over  $D$  to obtain landmark outgoing and incoming vectors  $\vec{X}_i$  and  $\vec{Y}_i$  in  $d$  dimensions,  $d < m$ , for each host  $\mathcal{H}_i$ . As before, we use  $X$  and  $Y$  to denote the  $d \times m$  matrices with  $\vec{X}_i$  and  $\vec{Y}_i$  as row vectors. Note that NMF can be used even when  $D$  contains missing elements.

Now suppose an ordinary host  $\mathcal{H}_{\text{new}}$  wants to gather distance information over the network. The first step is to calculate its outgoing vector  $\vec{X}_{\text{new}}$  and incoming vector  $\vec{Y}_{\text{new}}$ . To this end, it measures the network distances to and from the landmark nodes. We denote  $D_i^{\text{out}}$  as the distance to landmark  $i$ , and  $D_i^{\text{in}}$  as the distance from landmark  $i$  to the host. Ideally, we would like the outgoing and incoming vectors to satisfy  $D_i^{\text{out}} = \vec{X}_{\text{new}} \cdot \vec{Y}_i$  and  $D_i^{\text{in}} = \vec{X}_i \cdot \vec{Y}_{\text{new}}$ . The solution with the least squares error is given by:

$$\vec{X}_{\text{new}} = \arg \min_{\vec{U} \in \mathbb{R}^d} \sum_{i=1}^m (D_i^{\text{out}} - \vec{U} \cdot \vec{Y}_i)^2 \quad (11)$$

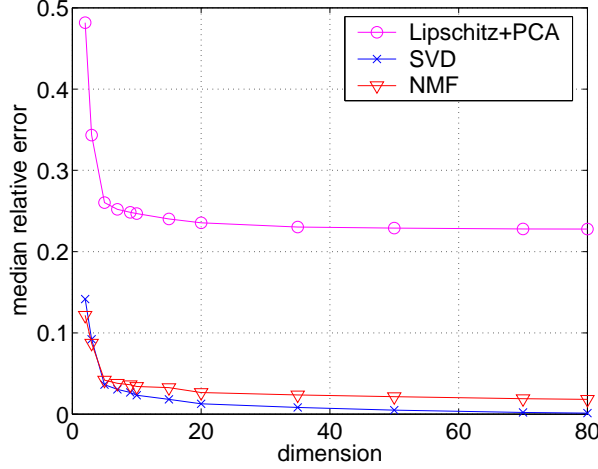
$$\vec{Y}_{\text{new}} = \arg \min_{\vec{U} \in \mathbb{R}^d} \sum_{i=1}^m (D_i^{\text{in}} - \vec{X}_i \cdot \vec{U})^2 \quad (12)$$

The global minima of these error functions, computed by simple matrix operations, have the closed form:

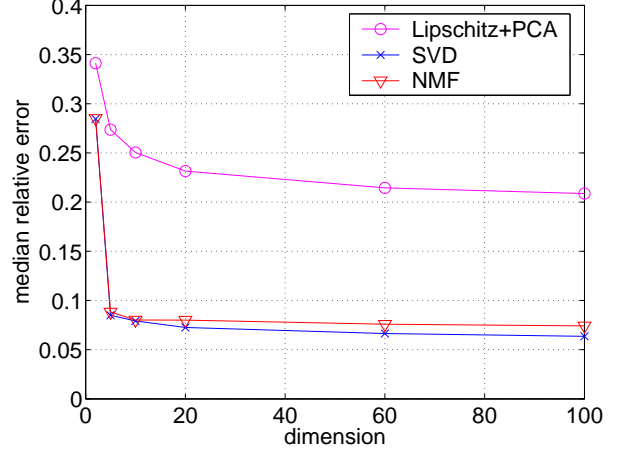
$$\vec{X}_{\text{new}} = (D^{\text{out}} Y)(Y^T Y)^{-1} \quad (13)$$

$$\vec{Y}_{\text{new}} = (D^{\text{in}} X)(X^T X)^{-1} \quad (14)$$

Eqs. (13–14) assume that the optimizations are unconstrained. Alternatively, one can impose nonnegativity constraints on  $\vec{X}_{\text{new}}$  and  $\vec{Y}_{\text{new}}$ ; this will guarantee that the predicted distances are themselves nonnegative (assuming that the landmark distance matrix was also modeled by NMF). The least squared error problems in Eqs. (11–12) can be solved with nonnegativity constraints, but the solution is somewhat more complicated. Our simulation results did



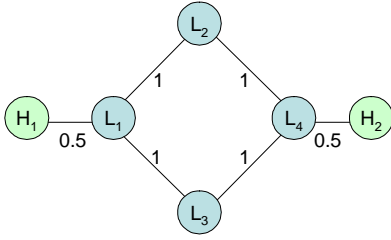
(a) Comparison over NLANR data set



(b) Comparison over P2PSim data set

**Figure 3: Reconstruction error comparison of SVD, NMF and Lipschitz over NLANR and P2PSim data set**

not reveal any significant difference between the prediction accuracies of least squares solutions with and without non-negativity constraints; thus, in what follows, we focus on the simpler unconstrained solutions in Eqs. (13–14).



**Figure 4: Four landmark nodes  $L_1 - L_4$  and two ordinary hosts  $\mathcal{H}_1, \mathcal{H}_2$  interconnected by a simple network topology**

We give a simple example of this procedure in Figure 4. The network is an enlarged version of the network in Figure 1, with the four original nodes serving as landmarks and two new nodes introduced as ordinary hosts. The first step is to measure inter-landmark distances and calculate landmark incoming and outgoing vectors. We used SVD to factor the landmark distance matrix in this example. The result is the same as the example in section 4:

$$X = \begin{bmatrix} -1 & 0 & 1 \\ -1 & -1 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & -1 \end{bmatrix}, Y = \begin{bmatrix} -1 & 0 & -1 \\ -1 & 1 & 0 \\ -1 & -1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

Note that SVD can be substituted by NMF and the following steps are identical.

Second, we measure the distance vectors for the ordinary hosts:  $D^{\text{out}} = D^{\text{in}} = [0.5 \ 1.5 \ 1.5 \ 2.5]$  for ordinary host  $\mathcal{H}_1$ . According to Eqs. (13 – 14),  $\vec{X}_{\mathcal{H}_1} = [-1.5 \ 0 \ 1]$ ,  $\vec{Y}_{\mathcal{H}_1} = [-1.5 \ 0 \ -1]$ . Similarly, we obtain the distance vector of  $\mathcal{H}_2$

as  $[2.5 \ 1.5 \ 1.5 \ 0.5]$ , and calculate its outgoing and incoming vectors:  $\vec{X}_{\mathcal{H}_2} = [-1.5 \ 0 \ -1]$ ,  $\vec{Y}_{\mathcal{H}_2} = [-1.5 \ 0 \ 1]$ . One can verify that distances between ordinary hosts and landmarks are exactly preserved. The distance between two ordinary hosts is not measured, but can be estimated as  $\vec{X}_{\mathcal{H}_1} \cdot \vec{Y}_{\mathcal{H}_2} = \vec{X}_{\mathcal{H}_2} \cdot \vec{Y}_{\mathcal{H}_1} = 3.25$ , while the real network distance is 3.

## 5.2 Optimization

The basic architecture requires an ordinary host to measure network distances to all landmarks, which limits the scalability of IDES. Furthermore, if some of the landmark nodes experience transient failures or a network partition, an ordinary host may not be able to retrieve the measurements it needs to solve Eqs. (13–14).

To improve the scalability and robustness of IDES, we propose a relaxation to the basic architecture: an ordinary host  $\mathcal{H}_{\text{new}}$  only has to measure distances to a set of  $k$  nodes with pre-computed outgoing and incoming vectors. The  $k$  nodes can be landmark nodes, or other ordinary hosts that have already computed their vectors. Suppose the outgoing vectors of those  $k$  nodes are  $\vec{X}_1, \vec{X}_2, \dots, \vec{X}_k$  and the incoming vectors are  $\vec{Y}_1, \vec{Y}_2, \dots, \vec{Y}_k$ . We measure  $D_i^{\text{out}}$  and  $D_i^{\text{in}}$  as the distance from and to the  $i$ th node, for all  $i = 1, \dots, k$ . Calculating the new vectors  $\vec{X}_{\text{new}}$  and  $\vec{Y}_{\text{new}}$  for  $\mathcal{H}_{\text{new}}$  is done by solving the least squares problems:

$$\vec{X}_{\text{new}} = \arg \min_{\vec{U} \in \mathbb{R}^d} \sum_{i=1}^k (D_i^{\text{out}} - \vec{U} \cdot \vec{Y}_i)^2 \quad (15)$$

$$\vec{Y}_{\text{new}} = \arg \min_{\vec{U} \in \mathbb{R}^d} \sum_{i=1}^k (D_i^{\text{in}} - \vec{X}_i \cdot \vec{U})^2 \quad (16)$$

The solution is exactly the same form as described in Eq. (13) and Eq. (14). The constraint  $k \geq d$  is necessary (and usually sufficient) to ensure that the problem is not singular. In general, larger values of  $k$  lead to better prediction results, as they incorporate more measurements of network distances involving  $\mathcal{H}_{\text{new}}$  into the calculation of the vectors  $\vec{X}_{\text{new}}$  and  $\vec{Y}_{\text{new}}$ .

We use the topology in Figure 4 again to demonstrate how the system works. As in the basic architecture, the first step is to measure inter-landmark distances and calculate landmark outgoing and incoming vectors. Secondly, the ordinary host  $\mathcal{H}_1$  measures the distances to  $L_1$ ,  $L_2$  and  $L_3$  as  $[0.5 \ 1.5 \ 1.5]$ . By Eq. (13) and Eq. (14), the vectors are  $\vec{X}_{\mathcal{H}_1} = [-1.5 \ 0 \ 1]$ ,  $\vec{Y}_{\mathcal{H}_1} = [-1.5 \ 0 \ -1]$ . Note that we did not measure the distance between  $\mathcal{H}_1$  and  $L_4$ , but it can be estimated as  $\vec{X}_{\mathcal{H}_1} \cdot \vec{Y}_{L_4} = [-1.5 \ 0 \ 1] \cdot [-1 \ 0 \ 1] = 2.5$ , which is in fact the true distance. Finally, the ordinary host  $\mathcal{H}_2$  measures the distances to  $L_2$ ,  $L_4$  and  $\mathcal{H}_1$  as  $[1.5 \ 0.5 \ 3]$ . Because all of them already have pre-computed vectors,  $\mathcal{H}_2$  can compute its own vectors by Eq. (13) and Eq. (14). The results are  $\vec{X}_{\mathcal{H}_2} = [-1.4 \ 0.1 \ -0.9]$ ,  $\vec{Y}_{\mathcal{H}_2} = [-1.4 \ -0.1 \ 0.9]$ . The distances between ordinary host  $\mathcal{H}_2$  and  $L_1/L_3$  are not measured directly, but can be estimated as  $\vec{X}_{\mathcal{H}_2} \cdot \vec{Y}_{L_1} = [-1.4 \ 0.1 \ -0.9] \cdot [-1 \ 0 \ -1] = 2.3$  and  $\vec{X}_{\mathcal{H}_2} \cdot \vec{Y}_{L_3} = [-1.4 \ 0.1 \ -0.9] \cdot [-1 \ -1 \ 0] = 1.3$ .

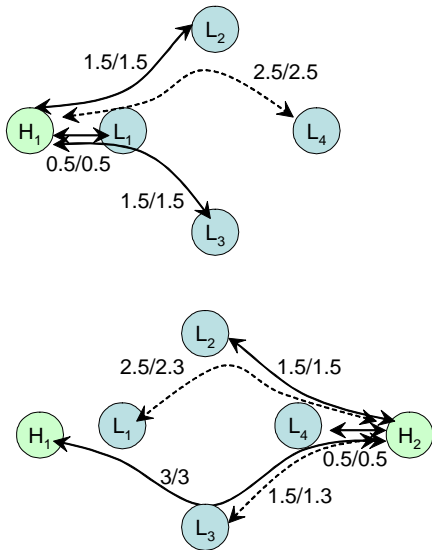


Figure 5: Learning outgoing and incoming vectors for two ordinary hosts. Solid lines indicate that real network measurement is conducted. Each edge is annotated with (real network distance / estimated distance).

This example illustrates that even without measurement to all landmarks, the estimated distances can still be accurate. In this example, most of the pairwise distances are exactly preserved; the maximum relative error is 15% when predicting the distance between  $\mathcal{H}_2$  and  $L_2$ . In the example, the load is well distributed among landmarks. As shown in Figure 5, distances to  $L_2$  are only measured twice during this estimation procedure. Such a scheme allows IDES to scale to a large number of ordinary hosts and landmarks. It is also robust against partial landmark failures.

## 6. EVALUATION

In this section we evaluate IDES, using SVD and NMF algorithms to learn models of network distances, and compare them to the GNP [13] and ICS [12] systems.

The experiments were performed on a Dell Dimension 4600 with Pentium 4 3.2GHz CPU, 2GB RAM. The GNP

implementation was obtained from the official GNP software release written in C. We implemented IDES and ICS in MatLab 6.0.

We identify four evaluation criteria:

- *Efficiency*  
We measure efficiency by the total running time required by a system to build its model of network distances between all landmark nodes and ordinary hosts.
- *Accuracy*  
The prediction error between  $D_{ij}$  and  $\hat{D}_{ij}$  should be small. We use the modified relative error function in Eq. (10) to evaluate accuracy, which is also used in GNP and Vivaldi. Note that predicted distances are computed between ordinary hosts that have not conducted any network measurements of their distance. Predicted distance errors are different than reconstructed distance errors (where actual network measurements are conducted).
- *Scalability*  
The storage requirements are  $O(d)$  for models based on network embeddings (with one position vector for each host) and matrix factorizations (with one incoming and outgoing vector for each host). In large-scale networks, the number of hosts  $N$  is very large. The condition  $d \ll N$  allows the model to scale, assuming that reasonable accuracy of predicted distances is maintained. Also, to support multiple hosts concurrently, it is desirable to distribute the load—for instance, by only requiring distance measurements to partial sets of landmarks.
- *Robustness*  
A robust system should be resilient against host failures and temporary network partitioning. In particular, partial failure of landmark nodes should not prevent the system from building models of network distances.

### 6.1 Efficiency and accuracy

We use three data sets for evaluating accuracy and efficiency.

- GNP: 15 out of 19 nodes in the symmetric data set were selected as landmarks. The rest of the 4 nodes and the 869 nodes in the AGNP data set were selected as ordinary hosts. Prediction accuracy was evaluated on  $869 \times 4$  pairs of hosts.
- NLANR: 20 out of 110 nodes were selected randomly as landmarks. The remaining 90 nodes were treated as ordinary hosts. The prediction accuracy was evaluated on  $90 \times 90$  pairs of hosts.
- P2PSim: 20 out of 1143 nodes were selected randomly as landmarks. The remaining 1123 nodes were treated as ordinary hosts. The prediction accuracy was evaluated on  $1123 \times 1123$  pairs of hosts.

Although deliberate placement of landmarks may yield more accurate results, we chose the landmarks randomly since in general they may be placed anywhere on the Internet. A previous study also shows that random landmark selection is fairly effective if more than 20 landmarks are employed [21].

data set	IDES/SVD	IDES/NMF	ICS	GNP
GNP	0.10s	0.12s	0.02s	1min 19s
NLANR	0.01s	0.02s	0.01s	4min 44s
P2PSim	0.16s	0.17s	0.03s	2min 30s

**Table 1: Efficiency comparison on IDES, ICS and GNP over four data sets**

To ensure fair comparisons, we used the same set of landmarks for all four algorithms. We also repeated the simulation several times, and no significant differences in results were observed from one run to the next.

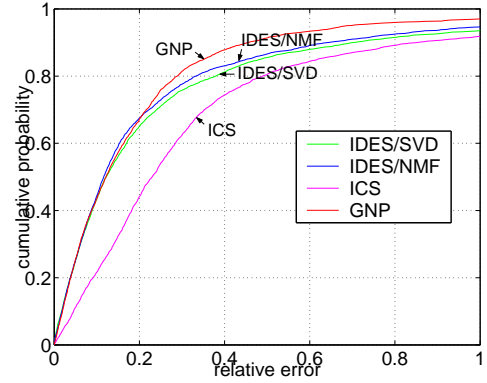
Table 1 illustrates the running time comparison between IDES, ICS and GNP. GNP is much more inefficient than the IDES and ICS. This is because GNP uses Simplex Downhill method, which converges slowly to local minima. Both IDES and ICS have running time less than 1 second, even when the data sets contain thousands of nodes. It is possible to reduce the running time of GNP by sacrificing the accuracy, but the parameters are hard to tune, which is another drawback of Simplex Downhill method.

Figure 6 plots the CDF of prediction errors for IDES using SVD, IDES using NMF, ICS and GNP over the three data sets respectively. In Figure 6(a), the GNP system is the most accurate system for the GNP data set. IDES using SVD and NMF are as accurate as GNP for 70% of the predicted distances. The GNP data set is somewhat atypical, however, in that the predicted distance matrix has many more columns (869) than rows (4). Figure 6(b) and 6(c) depict the CDF of prediction errors over NLANR and P2PSim data sets, which are more typical. In both cases, IDES has the best prediction accuracy. On the NLANR data set, IDES yields better results than GNP and ICS: the median relative error of IDES using SVD is only 0.03. Its 90 percentile relative error is about 0.23. The accuracy is worse for all three systems in P2PSim data set than in NLANR data set. However, IDES (with either SVD or NMF) is still the most accurate system among the three. The better prediction results on the NLANR data set may be due to the fact that 90% of the hosts in NLANR are in North America and the network distances, computed from minimum RTT over a day, are not affected much by queuing delays and route congestion. These properties make the data set more uniform, and therefore, more easily modeled by a low dimensional representation.

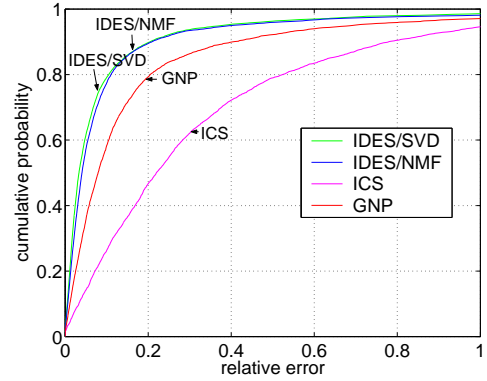
## 6.2 Scalability and robustness

In the previous subsection, we showed that IDES can accurately model the network distances in low dimensions  $d \leq 10$ , which is fundamental to make the system scale to large-scale networks. In this subsection, we study the impact of partially observed landmarks on the accuracy of IDES. Measuring the distances to only a subset of landmark nodes reduces the overall load and allows the system to support more ordinary hosts concurrently. It also makes the system robust to partial landmark failures.

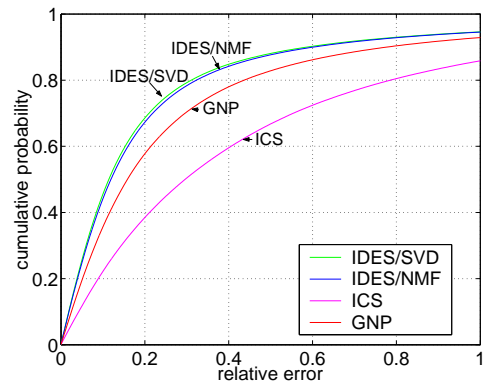
We simulated partially observed landmark scenarios in IDES using SVD to model partial distance matrices from the NLANR and P2PSim data sets. For each data set, we experimented with two settings: 20 random landmarks and 50 random landmarks. The simulation results are shown in Figure 7. The x-axis indicates the fraction of unobserved



(a) CDF of relative error over GNP data set, 15 landmarks

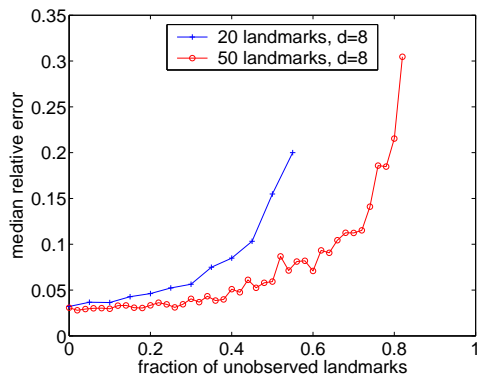


(b) CDF of relative error over NLANR data set, 20 landmarks

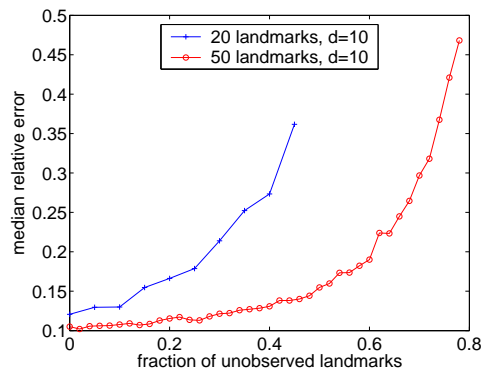


(c) CDF of relative error over P2PSim data set, 20 landmarks

**Figure 6: Accuracy comparison on IDES using SVD and NMF, ICS, and GNP,  $d = 8$**



(a) over NLANR data set



(b) over P2PSim data set

**Figure 7: the correlation between accuracy and landmark failures on IDES using SVD algorithm.**

landmarks. The unobserved landmarks for each ordinary host were independently generated at random. When the number of landmarks is less than twice the model dimensionality  $d$ , the accuracy appears sensitive to the fraction of unobserved landmarks. However, as the number of landmarks increases, the system tolerates more failure: for example, not observing 40% of the landmarks has little impact on the system accuracy when 50 landmarks are used in the test.

## 7. SUMMARY

In this paper, we have presented a model based on matrix factorization for predicting network distances between arbitrary Internet hosts. Our model imposes fewer constraints on network distances than models based on low dimensional embeddings; in particular, it can represent distances that violate the triangle inequality, as well as asymmetric network distances. Such a model is more suitable for modeling the topology and complex routing policies on the Internet. Based on this model, we proposed the IDES system and two learning algorithms, SVD and NMF, for factoring matrices of network distances between arbitrary Internet hosts. Simulations on real world data sets have shown that IDES is computationally efficient, scalable to large-scale networks, more accurate than previous models, and resilient to temporary landmark failures.

## 8. ACKNOWLEDGMENTS

We are grateful to Jonathan M. Smith (UPenn) for helpful comments on the manuscript, and Frank Dabek (MIT) for sharing the P2PSim data set. This material is based upon work supported by the National Science Foundation under Grant No. 0238323 and DARPA under contract F30602-99-1-0512.

## 9. REFERENCES

- [1] The NLANR active measurement project. <http://amp.nlanr.net/active/>.
- [2] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proceedings of 18th ACM SOSP*, 2001.
- [3] Suman Banerjee, Timothy G. Griffin, and Marcelo Pias. The interdomain connectivity of planetlab nodes. In *Proceedings of The 5th annual Passive and Active Measurement Workshop (PAM 2004)*, Antibes Juan-les-Pins, France, April 2004.
- [4] Manuel Costa, Miguel Castro, Antony Rowstron, and Peter Key. PIC: Practical internet coordinates for distance estimation. In *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS)*, Tokyo, Japan, March 2004.
- [5] Russ Cox, Frank Dabek, Frans Kaashoek, Jinyang Li, and Robert Morris. Practical, distributed network coordinates. In *Proceedings of HotNets-II*, Cambridge, MA, Nov 2003.
- [6] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of ACM SIGCOMM Conference*, Aug 2004.
- [7] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A Global Internet Host Distance Estimation Service. *IEEE/ACM Transactions on Networking*, Oct 2001.
- [8] Krishna P. Gummadi, Stefan Saroiu, and Steven D. Gribble. King: Estimating latency between arbitrary internet end hosts. In *Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002)*, Marseille, France, November 2002.
- [9] I.T. Jolliffe. *Principal component analysis*. Springer-Verlag, New York, 1986.
- [10] Karthik Lakshminarayanan and Venkata Padmanabhan. Some Findings on the Network Performance of Broadband Hosts. In *Proceedings of the Internet Measurement Conference*, Oct 2003.
- [11] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 556–562, 2000.
- [12] Hyuk Lim, Jennifer Hou, and Chong-Ho Choi. Constructing internet coordinate system based on delay measurement. In *Proceedings of the Internet Measurement Conference*, Oct 2003.

- [13] T. S. Eugene Ng and Hui Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *Proceedings of INFOCOM 2002*, New York City, NY, June 2002.
- [14] The p2psim project.  
<http://www.pdos.lcs.mit.edu/p2psim>.
- [15] Vern Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, 1997.
- [16] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, August 2001.
- [17] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, November 2001.
- [18] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the ACM SIGCOMM '01 Conference*, San Diego, California, August 2001.
- [19] Jeremy Stribling. All pairs of ping data for PlanetLab.  
[http://www.pdos.lcs.mit.edu/~strib/pl\\_app](http://www.pdos.lcs.mit.edu/~strib/pl_app).
- [20] Liying Tang and Mark Crovella. Virtual Landmarks for the Internet. In *Proceedings of the Internet Measurement Conference*, Oct 2003.
- [21] Liying Tang and Mark Crovella. Geometric exploration of the landmark selection problem. In *Proceedings of The 5th annual Passive and Active Measurement Workshop (PAM 2004)*, Antibes Juan-les-Pins, France, April 2004.
- [22] Ben Y. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report CSD-01-1141, U. C. Berkeley, Apr 2001.