TSMF: Network Latency Estimation using Matrix Factorization and Time Series Forecasting

Fotis Savva
School of Computer Science
University of Glasgow
Glasgow, UK
f.savva.1@research.gla.ac.uk

Christos Anagnostopoulos
School of Computer Science
University of Glasgow
Glasgow, UK
christos.anagnostopoulos@glasgow.ac.uk

Dimitrios Pezaros
School of Computer Science
University of Glasgow
Glasgow, UK
dimitrios.pezaros@glasgow.ac.uk

***Abstract*—The ability to accurately estimate end-to-end network latencies is extremely important for many services, from overlay network formation to Edge computing and 5G. Research in Network Coordinate Systems (NCS) has over the years focused on providing such estimates while conserving network resources by avoiding excessive probing. However, Internet latencies are inherently unstable and estimates produced by existing NCS's are shown to quickly become obsolete.**

In this paper, we devise TSMF, a novel NCS method based on an ensemble of Time-Series Forecasting and Matrix Factorization (MF). Fusing the two approaches results in a model that takes advantage of the low-rank structure of end-to-end latencies and temporal correlations with past measurements. In addition, TSMF can forecast future end-to-end latencies which has been impossible using existing NCS approaches.

Our results demonstrate that TSMF outperforms Euclidean and MF-based NCS's with up to $6\times$ less relative error in predicting end-to-end latencies. We also demonstrate the accuracy of TSMF in forecasting future end-to-end latencies, and its consequent suitability for services such as web-service recommendation.

I. Introduction

Knowing the evolution of latencies of end-to-end Internet paths is important for improving Quality-of-Service (QoS) in many types of networks, such as peer-to-peer and content delivery overlays. More recently, we see latency being used in identifying inter-node distances in fog/edge networks [7] and in the context of web-service recommendation [17]. End-to-end latencies are usually identified by measuring the one-way delay or round-trip time (RTT) between hosts A and B using active probing or through passively monitoring communication between hosts. However, for large number of paths, we risk overwhelming the network by having each host actively probing each other or risk never finding out about the latency between hosts that have never communicated before.

To remedy this, a number of approaches have emerged that can estimate latencies between a random pair of hosts without explicitly probing all paths. These methods exploit a limited set of latency measurements from prior communication of random hosts to construct models that can infer the latency of all other hosts not in this set. Hence, given measurements $d_{ij}, \forall (i,j) \in \Theta$, where d_{ij} is the latency between hosts i and j, and Θ is the set of pairs of nodes that have communicated previously, the model can infer $d_{ij}, \forall (i,j) \notin \Theta$. The two main categories of such methods are Euclidean Embedding methods [3], [12] and, more recently, Matrix Factorization (MF) [2], [8], [9], [11]. We commonly refer to these techniques as Network-Coordinate Systems (NCS).

Prior work on NCS's, largely ignores the temporal correlations that successive latency measurements might have. Hence, any prior measurements $d_{t-H,ij}$, where t is the latest measurement and $H \in [1, T]$ for a finite time horizon $T > 0$ indicates prior time-step/measurements, are not generally exploited for predicting latencies. Instead, both the Euclidean Embedding and MF approaches utilize *only* the latest measurements $d_{t,ij}, \forall (i,j) \in \Theta$ or an aggregate statistic like $\frac{\sum_{t=0}^{T} d_{t,ij}}{T}$. This could lead to wildly inaccurate estimates as latencies can vary significantly in the future. Incorporating prior measurements into a model could help enhance estimation accuracy. Both of these facts are demonstrated in Figure 1. Figure 1(left) measures the Coefficient-of-Variation (CoV) for measurements (varying over time) across random pair of hosts using the *Seattle* data set [9] and shows how much the measurements vary with respect to the mean. We compute CoVs of time-varying measurements $d_{t,ij}$ for each pair of hosts such that $\mu_{ij} = \frac{1}{T}\sum_t d_{t,ij}$ and $\sigma_{ij} = \sqrt{\frac{\sum_t (d_{t,ij} - \mu_{ij})^2}{T}}$. We then plot the Cumulative Distribution Function (CDF) of the CoVs which demonstrates that measurements vary significantly over time and a produced estimate at time t could be inaccurate at time $t+1$. Figure 1(right) shows how many prior measurements (denoted as time lags) act as good predictors (i.e., have strong correlation) with a measurement at time t. These are extracted using the autocorrelation function for each pair of hosts. It is evident that almost 80% of end-to-end latency estimates could benefit by, at least, the last 10 prior measurements.

In this paper, we propose TSMF, an ensemble methodology based on Time Series forecasting and Matrix Factorization to achieve better latency estimation. We exploit the *low-rank* structure of a matrix $\mathbf{D}$ that holds the limited number of measurements from set Θ and the temporal correlations that measurements have, to fuse together a model that is an ensemble of MF based on Stochastic Gradient Descent (SGD)

ISBN 978-3-903176-39-3

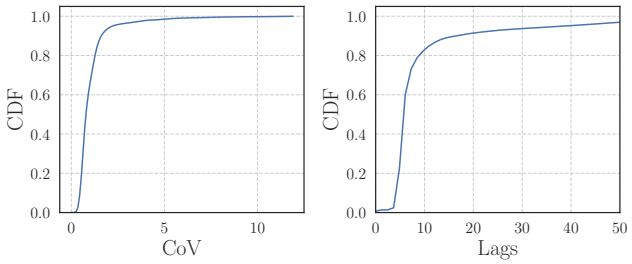

Fig. 1. (Left) CDF of CoV indicating that measurements fluctuate with respect to time; (Right) Number of prior measurements that can act as predictors

[1] and Time-series forecasting based on Simple Exponential Smoothing (SES) [4]. The primary purpose of TSMF is to estimate any missing latencies $d_{ij} \notin \Theta$. However, it can also forecast any future latencies $d_{t+H,ij}$ as it utilizes SES. In addition, we leverage the associated uncertainty of TSMF to construct a heuristic that proposes nodes (i, j) that should actively probe each other to identify their end-to-end latency. By forcing nodes to communicate, more recent measurements become available making TSMF more accurate.

The main contributions of this work are:

- A novel model (TSMF) for network latency estimation based on time-series forecasting and matrix factorization.
- A method to identify the subset of nodes which should communicate to improve overall latency estimation.
- a comprehensive performance evaluation and comparative assessment of TSMF against state-of-the-art NCS approaches.

Our results demonstrate that TSMF has up to $6\times$ less relative error in predicting end-to-end latencies, compared to standard approaches based on MF and Euclidean Embedding. In addition, we showcase TSMF's ability to perform forecasting and web-service recommendation along with exhibiting how to enhance accuracy by selective probing.

The rest of the paper is organized as follows: Section II presents an overview of current approaches. In Section III we give formal definitions for the task at hand as well as formulating the basic methodology behind MF. We then present TSMF in Section IV and how its use can inform decisions for selective probing at Section V. Section VI presents the evaluation of TSMF over different tasks.

II. Background

In Euclidean Embedding approaches such as Vivaldi [3], each host i is modeled as a set of coordinates in euclidean space such that $\mathbf{x}_i \in \mathbb{R}^d$. The latency between a pair of hosts is then estimated by $\hat{d}_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$, where $d_{ij} \approx \hat{d}_{ij}$ and $\|\mathbf{x}\|_2$ is the Euclidean (L2) norm of vector $\mathbf{x}$. Each available measurement $d_{ij} \in \Theta$ is used in a learning process to adjust the coordinates of every host proportionally with respect to the error $e = d_{ij} - \hat{d}_{ij}$. However, a big drawback of Euclidean Embedding methods, is the widespread Triangle Inequality Violations (TIVs) in network latencies [14]. A property of metric spaces such as the one described above is the Triangle Inequality which states that $d_{ij} \leq d_{ik} + d_{jk}$. However, this property can be violated in a number of settings where routing might be sub-optimal or follows diverse paths because of policy configurations [14], hence the large number of TIVs. This leads to inaccurate estimations and the suitability of such systems has been questioned in a number of surveys [6].

Alternative approaches based on Matrix Factorization (MF) [8], [9], [16], [17] do not suffer from this shortcoming. All available measurements from a set Θ are placed in a matrix $\mathbf{D} \in \mathbb{R}^{n\times p}$, where n is the number of hosts and p is the number of services. In a context where only the latencies amongst a fixed set of hosts is needed, we usually have $p = n$, so $\mathbf{D}$ is a square matrix. In the remainder and without loss of generality we assume $p = n$. Then, for each $D_{ij} = d_{ij}, \forall (ij) \in \Theta$, while we notate $D_{ij} = \texttt{missing}, \forall (i, j) \notin \Theta$. In cases where d_{ij} is the RTT between hosts i and j then $\mathbf{D}$ is also symmetric with $d_{ij} = d_{ji}$. MF methods, operate under the assumption that matrix $\mathbf{D}$ is *low-rank* so they learn a decomposition of the matrix $\mathbf{D} = \mathbf{XY}$, where $\mathbf{X} \in \mathbb{R}^{n\times r}$ and $\mathbf{Y} \in \mathbb{R}^{r\times n}$, where r is assumed to be the rank of matrix $\mathbf{D}$. The matrices $\mathbf{X}, \mathbf{Y}$ are learned from the available set of measurements. To estimate latencies between hosts i and j we can assign two vectors to each host $(\mathbf{x}_i, \mathbf{y}^i)$ and $(\mathbf{x}_j, \mathbf{y}^j)$ where the subscript i denotes the i^{th} row and the superscript i denotes the i^{th} column of a matrix. The estimation for measurement d_{ij} is then produced by $\hat{d}_{ij} = \mathbf{x}_i \mathbf{y}^j$.

III. Problem Formulation & Definitions

Let $\mathbf{D} \in \mathbb{R}^{n\times n}$ be a square matrix with

$$D_{ij} = \begin{cases} d_{ij} & \text{if} \quad (i,j) \in \Theta \\ \texttt{missing} & \text{if} \quad (i,j) \notin \Theta \end{cases}$$

where $d_{ij} \in \mathbb{R}$ indicates that a measurement of RTT delay exists between hosts i and j. The general objective of any NCS is then to find $\hat{\mathbf{D}}^*$ such that it approximates $\mathbf{D}$

$$\hat{\mathbf{D}}^* = \arg\min_{\hat{\mathbf{D}}} \|\mathbf{D} - \hat{\mathbf{D}}\|_F \tag{1}$$

where $||\cdot||_F$ is the Frobenius Norm. We define (1) as the general *loss* function. Due to a number of `missing` entries in $\mathbf{D}$, we approximate the general *loss* function as the squared-loss over all available measurements:

$$\min_{\hat{\mathbf{D}}} \sum_{i,j\in\Theta} (D_{ij} - \hat{D}_{ij})^2. \tag{2}$$

By exploiting the *low-rank* nature of $\mathbf{D}$, MF methods approximate $\mathbf{D}$ by a decomposition such that $\mathbf{D} \approx \hat{\mathbf{D}}$ and $\hat{\mathbf{D}} = \mathbf{XY}$. Where $\mathbf{X} \in \mathbb{R}^{n\times r}$ and $\mathbf{Y} \in \mathbb{R}^{r\times n}$ and r is the $\texttt{rank}(\mathbf{D})$.

A standard MF approach of estimating matrices $\mathbf{X}$ and $\mathbf{Y}$ is by optimizing a variant of (2) using SGD. Specifically, following this approach we have to minimize:

$$J(\mathbf{X}, \mathbf{Y}) = \sum_{i,j\in\Theta} (D_{ij} - \mathbf{x}_i \mathbf{y}^j)^2 + \lambda \|\mathbf{X}\|_F + \lambda \|\mathbf{Y}\|_F, \tag{3}$$

where the last two terms are regularization terms with $\lambda > 0$. As MF approaches suffer from the well-known overfitting

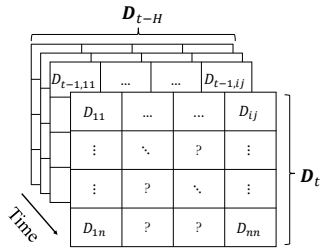

Fig. 2. Matrices $\mathbf{D}_t$, holding host pairs latency measurements, at different time-steps t.

problem in Machine Learning (ML), we have to heavily regularize to avoid this from happening. Overfitting in this case would cause poor accuracy in predicting the missing entries in $\mathbf{D}$. Using SGD, $\mathbf{X}, \mathbf{Y}$ are updated as:

$$\Delta \mathbf{x}_i = -\gamma \nabla_{\mathbf{x}} J(\mathbf{X}, \mathbf{Y}), \tag{4}$$

$$\Delta \mathbf{y}^j = -\gamma \nabla_{\mathbf{y}} J(\mathbf{X}, \mathbf{Y}), \tag{5}$$

with learning rate $\gamma \in (0, 1)$. The update rules at (4) and (5) are applied iteratively for random available measurements in the set Θ for a number of iterations until the matrices have reached a convergence threshold. Hence, the gradient is approximated by individual measurements:

$$\nabla_{\mathbf{x}_i} J(\mathbf{X}, \mathbf{Y}) = (D_{ij} - \mathbf{x}_i \mathbf{y}^j)\mathbf{y}^j + \lambda \mathbf{x}_i, \tag{6}$$

$$\nabla_{\mathbf{y}^j} J(\mathbf{X}, \mathbf{Y}) = (D_{ij} - \mathbf{x}_i \mathbf{y}^j)\mathbf{x}^i + \lambda \mathbf{y}^j. \tag{7}$$

Once the matrices $\mathbf{X}, \mathbf{Y}$ have been estimated, the missing entries can be predicted using $D_{ij} \approx \mathbf{x}_i \mathbf{y}^j, \quad \forall (i,j) \notin \Theta$. Although this approach is powerful, it lacks the opportunity of leveraging any prior communication that nodes have. It is usually the case that $\mathbf{D}$ is not the only matrix available but, instead we also have previously incomplete matrices $\mathbf{D}_{t-H}$. Thus, in the following section we introduce TSMF that exploits this fact.

IV. The TSMF Methodology

A. Forecasting Missing Latency Measurements

Figure 2 shows an example of how measurements are structured in matrix $\mathbf{D}$. Entries with "*?*", indicate `missing` entries such that $(i,j) \notin \Theta$. In addition, previous matrices are abbreviated as $\mathbf{D}_{t-H}$. As witnessed in Figure 1(right) at least 10 previous measurements are significantly autocorrelated with the measurements observed at time t. Using past measurements, one can build models to forecast the missing measurements at time-step t. Formally, given prior measurements $(d_{t-H,ij}, \ldots, d_{t-1,ij})$, the objective is to forecast the value of $d_{t,ij}$ leveraging all or a subset of the past measurements. Usually, the last L measurements are the most informative so standard forecasting models typically focus only on those. A popular forecasting model is the SES, which provides forecasts based on a weighted average of L previous measurements. The weights decrease exponentially with recent measurements being assigned more weight, i.e.,

$$d_{t,ij} = \sum_{k=0}^{L-1} \beta(1-\beta)^k d_{t-1-k,ij}, \tag{8}$$

with $\beta \in (0, 1)$ empirically set or optimised using a training period.

To approximate the complete matrix $\mathbf{D}$, we can stack all of the host pairs and obtain their previous measurement by matrix $\tilde{\mathbf{D}} \in \mathbb{R}^{n^2 \times L}$. We can also construct matrix $\mathbf{B} \in \mathbb{R}^{n^2 \times L}$ which holds the associated weight for each measurements computed using the produced coefficient at (8), such that for a series of L measurements we have vector $\mathbf{b} = [\beta(1-\beta)^L, \ldots, \beta(1-\beta)^0]^T$. Then, the general loss function in (1) is expressed as:

$$J(\tilde{\mathbf{D}}, \mathbf{B}) = \mathbb{E}[\|vec(\mathbf{D}) - diag(\tilde{\mathbf{D}}\mathbf{B}^\top)\|_2] \tag{9}$$

where $vec(\cdot)$ is the vectorized form of matrix $\mathbf{D}$, and $diag(\cdot)$ extracts the diagonal elements of a matrix.

B. Matrix Factorization & Time Series Forecasting Ensemble

Although a forecasting approach to this problem is powerful, it fails to account for correlations between entries in $\mathbf{D}$. In addition, using forecasting models alone might be inappropriate in cases where no previous measurements were observed for a particular pair of hosts. Therefore, we propose TSMF, a method that is a combination of MF with a Time-Series forecasting component fused together in an ensemble learning framework. Ensemble learning models have been proven powerful in ML modelling as they allow the combination of several statistical/ML models to increase predictive power. Using TSMF, a missing measurement is estimated as:

$$d_{t,ij} = \alpha \mathbf{x}_i \mathbf{y}^j + (1-\alpha) \sum_{k=0}^{L-1} \beta(1-\beta)^k d_{t-1-k,ij}. \tag{10}$$

In (10), we call the first term the MF component and the second term the Time-Series (TS) component, hence, the name TSMF. The parameter α is a weight placed on each component. It represents the trustworthiness of the estimates produced by the individual components. The MF and TS components can be estimated separately. The MF component is estimated by the update rules outlined at (4),(5). The TS component requires no estimation as the β parameter can be set heuristically as will be demonstrated in our experiments discussed later. Provisional values for α can be set using the training loss of the MF and TS components obtained from (3) and (9), respectively, where in both equations, the training loss is measured by using only the known measurements $(i,j) \in \Theta$. Abbreviating (3) by J_{MF} and (9) by J_{TS}, α can be set by $\alpha = 1 - \frac{J_{MF}}{J_{MF}+J_{TS}}$. This value is an approximation of how trustworthy each component is, since the training loss can be used as a proxy for how accurate each component (individually) will be at estimating missing entries.

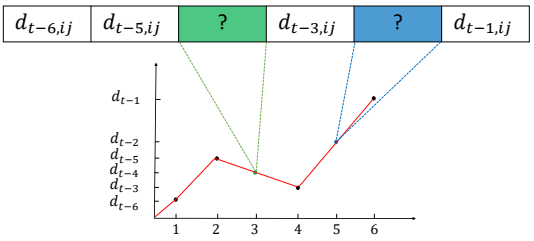

Fig. 3. Missing value imputation using Linear Interpolation for a sequence of measurements.

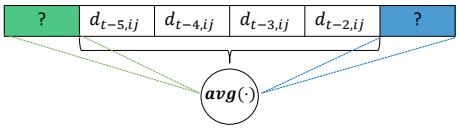

Fig. 4. Handling edge case (a-b) for missing value imputation by the arithmetic mean.

C. Estimating Previous Missing Measurements

A core challenge with the TS component is the existence of missing entries in previous measurements. As the TS component uses the matrix $\tilde{\mathbf{D}}$ which could contain missing entries in each row, we need an auxiliary method to impute any missing values. Formally, the TS component makes estimations using L values, where $\tilde{\mathbf{d}}_z$ are the previous L measurements for a pair of hosts (i, j). As the vector $\tilde{\mathbf{d}}_z$ contains $(d_{t-L,ij}, \ldots, d_{t-1,ij})$, it may very well be the case that we have a number of missing entries $\mathcal{M} = \{k | d_{t-k,ij} = \texttt{missing} \quad \forall k \in L\}$. To alleviate this, we use Linear Interpolation (LI), where a missing value is imputed by drawing a line between the missing value's adjacent entries.

A visual representation of this method is shown in Figure 3. In this example, we have two missing entries at time-steps $k = 4$ and $k = 2$. The values for $d_{t-4,ij}$ and $d_{t-2,ij}$ are imputed by their adjacent entries at time-steps $5, 3$ and $3, 1$ respectively using LI. The exact formula is shown at (11).

$$d_{k,ij} = \frac{1}{2}(d_{k-1,ij} + d_{k+1,ij}), \quad \forall k \in \mathcal{M}. \tag{11}$$

However, using LI, there are edge cases that we have to handle. Specifically, there are three cases where this method might be ineffective : (a-b) The first or/and last measurements are missing, where the first value is when $k = L$ and last $k = 1$ (c) All measurements are missing $|\mathcal{M}| = L$. For handling edge cases (a-b), we can simply use the arithmetic mean of all non-missing entries such that $d_{k,ij} = \frac{1}{L-|\mathcal{M}|} \sum_{d_{k,ij} \notin \mathcal{M}} d_{k,ij}$. A visual representation of this approach is shown at Figure 4, where the first and last entries are missing and the rest of the values are averaged to obtain an estimation of the missing values. For the third edge case (c), no imputation can be performed and the TS component cannot effectively provide an estimation for $d_{t,ij}$. However, a provisional prediction can be made using $avg(\hat{\mathbf{d}}_{t,i}, \hat{\mathbf{d}}_t^j)$, where the value for $d_{t,ij}$ is simply the expected value given all predictions made at row i and column j of matrix $\hat{\mathbf{D}}$, generated by the TS component.

D. Forecasting Future Measurements

A salient feature of TSMF is that it is also able to forecast future measurement values based on current knowledge. So far, our focus has been on estimating missing latency measurements $d_{t,ij}, \quad \forall (i, j) \notin \Theta$. Given these estimations, matrix $\mathbf{D}_t$ is now complete. Therefore, the TS component can incorporate the known measurements in $\tilde{\mathbf{D}}$ by shifting the window of L previous measurements by 1 such that any measurement at time-step t is now considered as a previous measurement. Given this information, forecasts for $\mathbf{D}_{t+1}$ can be produced by the TS component. The MF component in this scenario provides the same estimation values as the previous time-step such that $\hat{\mathbf{D}}_t = \mathbf{XY} = \hat{\mathbf{D}}_{t+1}$. Within the time-series forecasting context, this is the same as the Naive forecasting method [10] in which the last known value of a time-series at time step t is the produced forecast for time step $t+1$. In essence, TSMF is an ensemble technique combining a Naive forecasting model with SES. By forecasting future measurements, we allow NCS to make optimal choices for future communication without incurring additional communication overhead.

V. Enhanced Latency Estimation by Selective Probing

In current NCSs, an assumption is made on how known measurements become available. This can happen in an active probing setting or by collecting passive measurements betweeen randomly communicating nodes. In an active probing case, a static set of hosts are called to exchange messages that carry the end-to-end latency of the communicating hosts. However, some end-to-end latencies might be harder to predict than others. Formally, each predicted measurement $\hat{d}_{ij}$ is associated with an error ϵ, such that $d_{ij} = \hat{d}_{ij} + \epsilon$, where ϵ is a random variable. It could very well be the case that ϵ varies for different measurements with the source of error for TSMF coming from both the MF component and the TS component. Some explanations to this is that the measurement d_{ij} might be less correlated with other measurements such that the MF component is unable to produce a 'good' prediction. With respect to the TS component, a time-series of measurements for host-pair (i, j) might be more uncertain than others.

We conduct an experiment in which we randomly select the missing entries for a matrix $\mathbf{D}$, produce predictions and repeat for a number of times/initializations. The results of this experiment are shown in Figure 5. At each initialization, we randomly select a number of entries in matrix $\mathbf{D}$ and set them as `missing`. Matrix $\mathbf{D}$ is then approximated by a deterministic MF method such as Singular Value Decomposition (SVD) to avoid the stochastic nature of our own MF component. We use SVD for this experiment since a

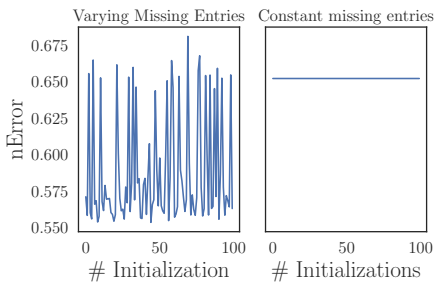

Fig. 5. Error variation by approximating matrix $\mathbf{D}$, each time with different missing values. (Left) At each *Initialization*, missing entries of $\mathbf{D}$ are different causing variation in accuracy. (Right) At each *Initialization* missing entries are constant.

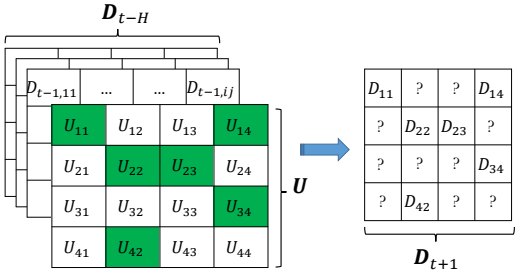

Fig. 6. The entries with the highest values of u are selected and communication is imposed at time-step $t+1$ for the associated host-pairs thus revealing those entries at $\mathbf{D}_{t+1}$.

stochastic algorithm would not be able to show whether the variations in accuracy are due to the algorithm or a different selection of entries. This is reinforced by Figure 5(right) since with the same set of missing entries, the SVD algorithm approximates $\mathbf{D}$ with the same error. The metric *nError* here is produced by $\frac{||\mathbf{D}-\hat{\mathbf{D}}||_F}{||\mathbf{D}||_F}$. However, it is evidenced by Figure 5(left) that we can approximate $\mathbf{D}$ with arbitrary accuracy by selecting different entries of matrix $\mathbf{D}$ to be set as `missing`. This finding suggests that if we are able to identify which measurements should be known at the following time-step then we can approximate $\mathbf{D}_{t+1}$ with more accuracy by imposing communication on pairs (i,j).

For this task, we construct an uncertainty matrix $\mathbf{U} \in \mathbb{R}^{n\times n}$. Each host pair (i,j) becomes associated with an uncertainty value u_{ij} which denotes the inherent uncertainty in predicting what the latency $d_{t,ij}$ would be for a particular pair. The uncertainty value u is computed using the TS component which can provide us with an estimation of uncertainty, given the time-series of measurements from particular host-pairs. The uncertainty value is computed using the residuals associated with the predictions of previous measurements made by the TS component. An estimation residual is computed as $e_t = (d_{t,ij} - \hat{d}_{t,ij})^2$, where $\hat{d}_{t,ij}$ is the forecast produced by the TS component, thus, the uncertainty value is:

$$u_{ij} = \sqrt{\sum_{k=t-L}^{t-1} (e_{k,ij} - \overline{e}_{ij})^2}, \tag{12}$$

where, in (12), $\overline{e}_{ij}$ is the average of the residuals associated with a particular host pair. This expression is equivalent to the standard deviation σ_e of the residuals ($L \to \infty$). As more variation in the residuals corresponds to uncertain forecasts from the TS component we can impose communication at time-step $t+1$ for all of the host-pairs with the highest associated uncertainty. This is depicted in Figure 6. The highlighted entries correspond to the ones with the highest values of u. At the following time-step $t+1$ the partially filled matrix $\mathbf{D}_{t+1}$ will only have missing entries for host-pairs which were *not* selected by using $\mathbf{U}$.

The selected entries are chosen based on a budget $B \in \mathbb{N}$ by the network operator. The budget can be set appropriately so

Algorithm 1: Selecting uncertain entries based on B

Input : B is the assigned budget
$\mathcal{U}$ set of uncertainty values from $\mathbf{U}$
Output: $\mathcal{R}$ most uncertain pairs (i,j)
while $B > 0$ ***and*** $\mathcal{U} = \emptyset$ **do**
  $\mathcal{R} = \mathcal{R} \cup \{u\}, u = \max_{a\in\mathcal{U}}(U)$;
  $\mathcal{U} = \mathcal{U} \setminus \{u\}, u = \max_{a\in\mathcal{U}}(U)$;
  $B \leftarrow B - 1$;
end

that the network is not overwhelmed by probing messages between host-pairs wishing to identify their end-to-end latencies. Algorithm 1 describes how this selection is made. Based on B we select the pair with the highest associated uncertainty u that is included in set $\mathcal{U} = \{u_{11}, u_{12}, \ldots, u_{ij}, \ldots, u_{nn}\}$. The corresponding pair is added to $\mathcal{R}$ and subsequently removed from $\mathcal{U}$. This sequence is executed iteratively until the budget is exhausted $B = 0$ or not sufficient pairs remain in $\mathcal{U}$.

VI. Performance Evaluation

In this section, we evaluate TSMF under different scenarios using a variety of metrics. Our aim is to answer the following questions:

1) How accurate is TSMF compared to other baseline and recent approaches?
2) How well does TSMF perform in web-service recommendation and forecasting tasks?
3) Can TSMF achieve better overall accuracy by selecting hosts that should communicate at next time-step $t+1$?
4) How sensitive is TSMF to different associated parameters?

A. Latency Prediction Accuracy

We evaluate the accuracy of TSMF in predicting missing entries from incomplete matrices from two well-known data sets used in the literature [9], [16], abbreviated as *Seattle* and *PlanetLab*. *Seattle* contains 688 matrices, each one with 99×99 entries, which correspond to measurements between different devices (spanning from cellular devices to desktop computers) taken at different time intervals. *PlanetLab* contains 18 matrices, with 490×490 entries, taken from devices

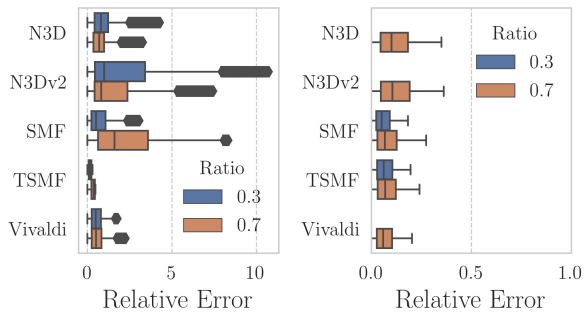

Fig. 7. Relative Error across different models for (left) *Seattle* (right) *PlanetLab* datasets and ratios of missing entries.

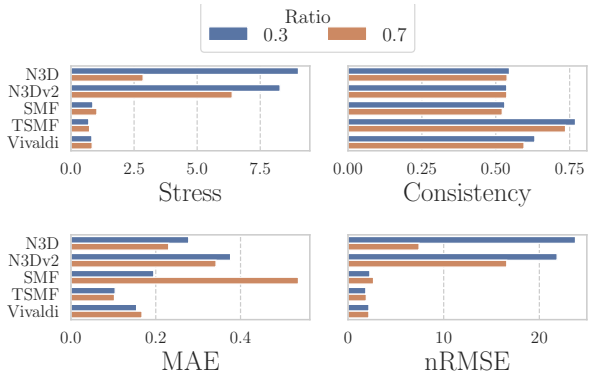

Fig. 8. Alternative latency prediction accuracy metrics measured for different models at different ratios (*Seattle* dataset).

in a peer-to-peer network. The accuracy is measured using *relative error* which is computed by $rel = |\frac{d_{ij}-\hat{d}_{ij}}{d_{ij}}|$ (where *less* is better). We sample 100 random matrices from the *Seattle* dataset and *20* random matrices from the *PlanetLab* dataset to compute the various evaluation metrics described below.

We compare TSMF with 4 other methods: (1) **N3D** (2) **N3Dv2** are algorithms described in [9], N3D follows a hybrid approach using MF and Euclidean Embedding. To the best of our knowledge, N3Dv2 is the only available method taking the temporal dynamics of measurements under consideration. Hence, we are mainly interested in comparing TSMF's accuracy with N3Dv2. In addition, we assess (3) **SMF**, which is an MF approach using SGD (to be used as a baseline) and (4) **Vivaldi** [3], which is a well known Euclidean Embedding method. **Note:** Details on parameters used and reproducibility of experiments will be made available in a github repository. We omit this for now to not violate the double-blind constraint.

The results of this experiment are shown in Figure 7(left) for *Seattle* and Figure 7(right) for *PlanetLab*. For each model, two boxplots are shown that correspond to different ratios for missing entries. The blue boxplots correspond to 30% of the total entries in the dataset missing whereas the orange boxplots correspond to 70% of the entries missing. For *PlanetLab*, we have removed the blue boxplots for all models apart from TSMF and SMF as the rest of the methods performed poorly, possibly due to overfitting. As the number of available entries is at 0.7×490^2 (when the missing ratio is at 30%), some methods could overfit due to the large number of *training* examples and not be able to generalize to predict missing entries.

However, all models performed similarly when the missing ratio for *PlanetLab* is at 70%. This is because we observed little temporal variability in the *PlanetLab* dataset as was also witnessed by [9]. On the other hand, we observe TSMF's accuracy far surpassing all other methods under comparison for the *Seattle* dataset, in which the temporal variability was much higher. We also witness that N3Dv2 performed worse at the tails of the distribution than its simple predecessor, N3D, which does not incorporate any temporal information.

In our comparative assessment for latency prediction accuracy, we include certain additional metrics to report on different aspects that have to be considered when estimating end-to-end latencies. We report our findings on 4 additional metrics: (1) *Stress*, (2) *Consistency*, (3) *Median Absolute Error* (MAE) and (4) *normalized Root Mean Squared Error* (nRMSE). *Stress* is computed over all entries by $\sqrt{\frac{\sum_{ij}(d_{ij}-\hat{d}_{ij})^2}{\sum_{i\neq j} d_{ij}^2}}$; it characterises the overall fitness of the model (less is better) and has also been reported in prior work [8]. *Consistency* is an important aspect introduced in [5]; it outputs a statistic lying between $[0, 1]$, (closer to 1 is better) reporting on whether the implication $\hat{d}_{i,j} > \hat{d}_{u,v} \iff d_{i,j} > d_{u,v}$ holds. The implication, denotes that a predicted measurement $\hat{d}_{i,j}$ should be greater (or less) than another predicted measurement, $\hat{d}_{u,v}$ if and only if the true measurement, $d_{i,j}$ is greater (or less) than the other true measurement, $d_{u,v}$. This is an important evaluation metric to consider when comparing different end-to-end latencies between random pairs of hosts; especially in cases where we wish to produce a *sorted list* of low-latency hosts. In addition, *MAE* is an indication of the prediction error (in terms of seconds) for 50% of all estimated entries. Lastly, we report on *nRMSE* which is a complimentary metric to *Relative Error* as it allows for model comparison that is not susceptible to large errors when measurements (d_{ij}) are small.

All metrics were computed using the *Seattle* dataset over 100 random matrices with $\{30\%, 70\%\}$ entries missing. The results are shown in Figure 8 and, at a first glance, we notice that TSMF outperforms all other approaches for all metrics used. Specifically, for *Stress*, we see a large difference across both ratios used between SMF, TSMF, Vivaldi and N3D, N3Dv2. TSMF, has the lowest *Stress* reported at 0.70 and 0.73 for ratios 30% and 70%, respectively. In addition, TSMF reports the highest overall *Consistency* at 77% and 74% for ratios 30% and 70%, respectively. This shows that more than 70% of pair comparisons are accurate. The rest of the models are close to 50% indicating the superiority of TSMF in this task.

Moreover, the *MAE* for TSMF is reported at 0.10 (seconds), $3\times$ less than the MAE reported by N3D, N3Dv2 methods and

Model	p@10	p@100	NDCG@10	NDCG@100
NMF	0.012646	0.089488	0.147761	0.215652
SVD	0.007602	0.059561	0.133791	0.172975
SMF	0.021637	0.165234	0.190211	0.310334
TSMF	**0.086330**	**0.293816**	**0.325301**	**0.463622**
TSMF-NMF	0.044956	0.228070	0.247146	0.383865
TSMF-SVD	0.042032	0.227675	0.230410	0.368401

TABLE I
WEB-SERVICE RECOMMENDATION ACCURACY VALUES VARY WITHIN $[0, 1]$. LARGER VALUES SIGNIFYING BETTER ACCURACY; HIGHLIGHTED VALUES ARE THE LARGEST PER-COLUMN.

around $5\times$ less than SMF. Lastly, for *nRMSE*, there is notable difference between SMF, TSMF, Vivaldi and N3D/N3Dv2 methods. This reinforces our findings exhibited in Figure 7 suggesting better overall accuracy when using TSMF. We also note that Vivaldi performs similarly well but its application is limited and cannot be utilized in additional tasks, such as *Web Service recommendation* and *Forecasting*. [1]

B. Recommendation & Forecasting Prediction Accuracy

In this experiment, we evaluate TSMF in the context of Web-Service recommendation, where the task is to identify the top-k services (with the lowest latency) a particular host should communicate with to improve QoS. We used a dataset made available by [15], which includes 64 matrices with measurements from 4500 web-services by 142 users. For each user, these services are sorted in an ascending order using the actual measurements and then each method is evaluated based on its produced predicted measurements. We use the fundamental metrics *Precision@k* (p@k) and *Normalized Discounted Cumulative Gain@k* (NDCG@k) generally used in Recommender Systems and also used by prior work [17]. These metrics assess the accuracy of the methods in producing top-k ranked lists of web services.

We compare TSMF against alternative MF approaches: *Nonnegative Matrix Factorization* (NMF) and *Singular Value Decomposition* (SVD) and Simple MF (SMF). We also include these methods as alternatives to the MF component in TSMF, namely TSMF-NMF, TSMF-SVD. The results are shown in Table I, in which TSMF performed best, as shown by the bold entries which denote the highest score for each column (metric). Specifically, *p@k*, shows the overlap ratio between the true top-k list of web-services and the predicted top-k list. So for $k = 10$ we see that all methods do not perform particularly well, with TSMF obtaining the highest ratio at 0.08. However, when computing the average latency difference between web-services in the true top-10 list and the predicted list obtained using TSMF, we note that latency differences were at 0.37 (seconds). Hence, although the predicted top-10 list does not include the same web-services (per their ID), it includes similar (in terms of latency) web-services. As the number of web-services k increase, so does *p@k*, with TSMF having the highest ratio at 0.29 for *p@*100.

[1] Similar results are obtained from *PlanetLab*; not showed here due to space limitations.

	TSMF	SES	TSMF-SVD	TSMF-NMF
RMSE	0.57	0.56	0.54	0.53

TABLE II
FORECASTING ACCURACY (RMSE)

In general, it is important to produce a predicted top-k list, where the initial entries match the true entries more than entries listed at the end of the list. Because there should be more weight given for getting the first few entries right as it is those web-services that the user will connect to. The metric that we have included in this experiment, *NDCG@k*, captures this phenomenon. The metric penalizes a method which includes low-latency web-services at the end of the list and *gain* is accumulated from the top with entries at the end of the list having a *discounted gain*. For both *NDCG@*10 and *NDCG@*100, TSMF performs best, indicating that the produced top-k lists include the lowest-latency web-services at the start of the list. Overall, the results show the applicability of our proposed method in the context of web-service recommendation.

We also evaluate TSMF's capability in forecasting future measurements based on prior and current measurements. Being able to forecast future end-to-end latencies between hosts is important as a particular host can pre-configure communication with other hosts. In this experiment, we used the *Seattle* dataset and compared the accuracy of TSMF, against SES, TSMF-NMF, and TSMF-SVD. We evaluate their accuracy based on Root Mean Squared Error (RMSE) measured using all entries in $\mathbf{D}_{t+1}$[2]. Each model produces estimates for future measurements based on the last $L = 5$ matrices. The results of this experiments are shown in Table II, which evidences that all approaches perform similarly to this task. On average, the forecasts of TSMF are expected to be off by 0.57 (seconds). All other methods have similar results as "TS"=SES. The only difference, is the MF components used which also provide forecasts by using the latest estimation. However, no notable differences were observed.

C. Accuracy Improvement using Pre-selected Entries

As described in Section V, we can leverage the uncertainty values of the TS component to identify which hosts *should* communicate during the following time-steps. By doing so, we depart from a passive monitoring scenario and leverage active probing to improve overall accuracy in estimating the rest of the missing entries. To examine this premise, we conduct an experiment using the *Seattle* dataset with a fixed ratio of 30% missing entries. We select 50 random matrices with missing entries selected at (a) random and (b) optimal selection using uncertainty values. The TSMF is then used to predict the remaining missing entries and the relative error associated with each missing entry is computed. In Figure 9, we plot the boxplots describing the distribution of relative error for both approaches. Overall, we observe lower error by using our proposed approach. When performing a paired t-test using the

[2] We use 20 randomly selected matrices.

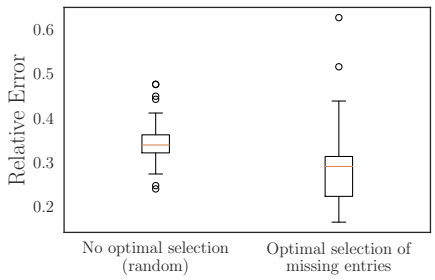

Fig. 9. Optimal selection of missing entries using the uncertainty values associated with measurement entries.

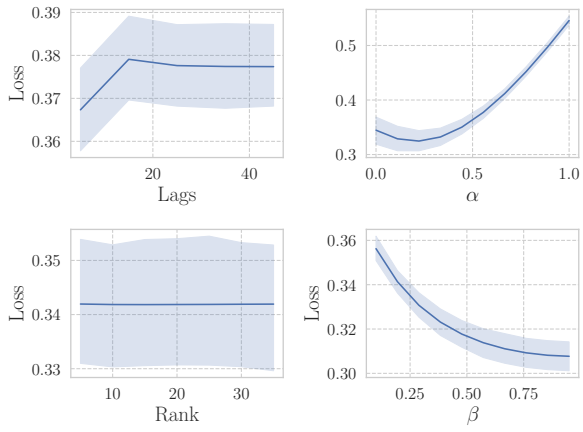

Fig. 10. TSMF sensitivity analysis over different hyper-parameters.

obtained relative error values, we reject the null hypothesis of the mean relative error being identical with $p = 0.00002$. This is because TSMF has the ability to flag the most uncertain entries in matrix $\mathbf{D}_t$. By flagging these entries, it is causing them to be revealed at the next time step $\mathbf{D}_{t+1}$ with the remaining entries being much easier to estimate.

D. TSMF Sensitivity Analysis

TSMF has a number of hyper-parameters that should be tuned (usually using past known matrices) before being used to make future estimations. In this section, we evaluate the impact of each of those hyper-parameters to assess their importance and identify well-performing parameter values. We use the *Seattle* dataset and select 10 random matrices with 30% missing entries and evaluate the *loss* $= \frac{\|\mathbf{D}-\bar{\mathbf{D}}\|_F}{\|\mathbf{D}\|_F}$ for each tested parameter value. Specifically we vary: (a) time *lags*, $L \in [5, 50)$, (b) parameter $\alpha \in [0, 1)$, (c) rank $r \in [5, 40)$, (d) parameter $\beta \in [0.1, 0.95)$. The results of the sensitivity analysis are shown in Figure 10. For parameter L (time lags), loss increases and then stabilizes, after incorporating more than 15 past observations. This is because more distant observations are less informative and negatively influence estimations. Parameter α influences the balance between predictions from the MF and TS components. An increased value for α means that more weight is given to the predictions made by the MF components and less to the TS component. As seen in this experiment, the TS component produces better estimations than the MF component for this particular dataset. Specifically, the lowest error is obtained at $\alpha = 0.2$, at which point the error starts increasing for $\alpha > 0.2$. It does not imply that the MF component alone is not accurate. Instead, it demonstrates the superiority of using an ensemble (hybrid) approach combining both MF and TS components. Relying only on the MF component at $\alpha = 1$ can often lead to less-accurate estimates as the temporal dynamics of measurements are not exploited, as evidenced in the comparative assessment.

In Figure 10 (lower-left), we observe that an increased rank r offers no significant additional benefit in this case. The missing measurements can be estimated accurately with a relatively small rank for the MF component. In addition, as we fix the value of $\alpha = 0.5$, the MF component does not produce the end predictions by itself, thus, the relative importance of *rank* might be decreased. Suggesting that this experiment might have failed to detect the actual importance of *rank* to the MF component. However, when testing with $\alpha = 1$, (giving full weight to the MF component) there was no observable difference to the sensitivity results. Lastly, we examine the impact of parameter β which is associated with the TS component. As β is increased more weight is given to the most recent observations and less weight to more distant ones. By considering more distant observations, the expected loss is increased, reinforcing our finding for parameter L which demonstrated the same thing. However, as β is increased, the loss steadily decreases indicating that, in this case, more recent observations carry more information and are more useful for making predictions. (Note: SGD learning rate γ in (5), (4) is set to 0.1 as suggested in [1] for robust convergence.)

VII. Related Work

Over the past decades, research in NCS has been strong, mainly due to the rise of different types of overlay networks. The challenge in such networks is being able to predict end-to-end latencies in diverse paths for optimal peer selection. Initial attempts at this were made by methods such as [3], [12]. The main approach was to assign Euclidean coordinates for each host and then use a learning procedure to adapt these coordinates so that the distance between a pair of hosts approximates their end-to-end latency. Such approaches shown to suffer from inaccuracies in modeling end-to-end latencies due to large number of TIVs [6].

On the contrary, more recent work [2], [7]–[9], [11], [13], [16], [17], including the one presented in this paper, uses Matrix Factorization which does not suffer from this shortcoming. MF methods can be categorised to centralized [9], [16], decentralized [8] or landmark approaches [11]. In centralized approaches, it is assumed that all known measurements are available centrally where the algorithm is executed and the predicted measurements are subsequently distributed to the hosts. Although this approach might have some overhead in

communication it generally leads to more stable/accurate solutions as all information necessary is available centrally. In a decentralized setting, each host leverages local measurements and estimations are performed locally. Specifically, every host i, adapts its assigned vectors $\mathbf{x}_i, \mathbf{y}^i$, using measurements that were obtained passively by communicating with K other hosts. All K hosts, also transmit their vectors $\mathbf{x}_k, \mathbf{y}^k, \forall k \in K$ to host i. Host i is then able to perform estimations *locally* by using the available information. However, this might prove problematic in scenarios where a host i wishes to estimate latencies for all other $N-1$ hosts in a network with total participating nodes set as N. As it only has information on K nodes, with $K \ll N$ this becomes impossible. Lastly, a landmark-based approach sets random hosts as *landmarks* with which the participating hosts exchange messages. The latency between a pair of hosts is then determined by the distance of the two hosts to the landmark nodes. We note that this approach could lead to significant inaccuracies as the paths followed from host to landmark and from host to host could be drastically different. For TSMF, we assume a centralized architecture, where all information is available centrally. Once TSMF is *trained*, its components can be transmitted to the participating hosts such that estimations for all other hosts can be performed locally.

The temporal nature of measurements has not been exploited fully by prior work. In [9], [16], this property was used to construct a much bigger matrix by stacking the matrices obtained at previous time-steps. In our initial experimentation and during our comparisons with the approaches discussed at [9], we find this approach to be extremely time-consuming and not being able to fully exploit the temporal property. In addition, in [13], the dynamic nature of measurements was addressed (not exploited), by assuming that each measurement d_{ij} is inherently uncertain and assigned an error term to account for this fact. To our knowledge TSMF, is the first model proposed that exploits the temporal property of measurements by fusing together time-series forecasting and MF models.

In addition, we notice a transition in the context to which NCSs are applied. From peer-to-peer networks in Vivaldi [3] to mobile networks [9], to Fog/Cloud latency estimation [7] and lastly for Web-Service recommendation [17]. TSMF, does not make any assumptions as to the context that it can be applied. Hence, we provide various experimental results using data sets from a variety of contexts.

VIII. Conclusions

We introduce TSMF, a hybrid approach based on Time Series forecasting and Matrix Factorization, capable of accurately estimating and forecasting end-to-end latencies in large networks. Unlike current MF approaches, TSMF exploits the temporal dynamics of latency measurements structured in matrices to achieve accurate and robust estimations. As it leverages an ensemble approach, TSMF outperforms several other approaches based on MF, Euclidean Embedding and approaches that also exploit temporal properties of measurements. In addition, we evaluate and show that TSMF acts as a web-service recommender for users wishing to communicate with the fastest responding node. We have also demonstrated that TSMF forecasts future end-to-end latencies, something that is not possible with the standard approaches witnessed in prior work. Finally, using TSMF, we are capable of selecting pairs of hosts that should communicate in the future such that TSMF's accuracy is improved when making future estimations.

Acknowledgments

This work has been supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) grant EP/N033957/1.

References

[1] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[2] Y. Chen, X. Wang, C. Shi, E. K. Lua, X. Fu, B. Deng, and X. Li. Phoenix: A Weight-Based Network Coordinate System Using Matrix Factorization. *IEEE Transactions on Network and Service Management*, 8(4):334–347, Dec. 2011.

[3] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a decentralized network coordinate system. In *SIGCOMM '04*, SIGCOMM '04, pages 15–26, Portland, Oregon, USA, Aug. 2004. Association for Computing Machinery.

[4] C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International journal of forecasting*, 20(1):5–10, 2004.

[5] H. Huang, H. Yin, G. Min, D. Wu, Y. Wu, T. Zuo, and K. Li. Network distance prediction for enabling service-oriented applications over large-scale networks. *IEEE Communications Magazine*, 2015.

[6] S. Lee, Z.-L. Zhang, S. Sahu, and D. Saha. On Suitability of Euclidean Embedding for Host-Based Network Coordinate Systems. *IEEE/ACM Transactions on Networking*, 18(1):27–40, Feb. 2010.

[7] J. Li, T. Zhang, J. Jin, Y. Yang, D. Yuan, and L. Gao. Latency estimation for fog-based internet of things. In *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–6, Nov. 2017.

[8] Y. Liao, W. Du, P. Geurts, and G. Leduc. DMFSGD: A Decentralized Matrix Factorization Algorithm for Network Distance Prediction. *IEEE/ACM Transactions on Networking*, 21(5):1511–1524, Oct. 2013.

[9] B. Liu, D. Niu, Z. Li, and H. V. Zhao. Network latency prediction for personal devices: Distance-feature decomposition from 3D sampling. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 307–315, Apr. 2015.

[10] S. Makridakis, S. C. Wheelwright, and R. J. Hyndman. *Forecasting methods and applications*. John wiley & sons, 2008.

[11] Y. Mao, L. K. Saul, and J. M. Smith. IDES: An Internet Distance Estimation Service for Large Networks. *IEEE Journal on Selected Areas in Communications*, 24(12):2273–2284, Dec. 2006.

[12] T. Ng and Hui Zhang. Predicting Internet network distance with coordinates-based approaches (GNP). In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 170–179, New York, NY, USA, 2002. IEEE.

[13] R. Tripathi and K. Rajawat. Dynamic Network Latency Prediction with Adaptive Matrix Completion. In *2018 International Conference on Signal Processing and Communications (SPCOM)*, pages 407–411, July 2018.

[14] H. Zheng, E. K. Lua, M. Pias, and T. G. Griffin. Internet routing policies and round-trip-times. In *International Workshop on Passive and Active Network Measurement*, pages 236–250. Springer, 2005.

[15] Z. Zheng, Y. Zhang, and M. R. Lyu. Investigating qos of real-world web services. *IEEE transactions on services computing*, 7(1):32–39, 2012.

[16] R. Zhu, B. Liu, D. Niu, Z. Li, and H. V. Zhao. Network Latency Estimation for Personal Devices: A Matrix Completion Approach. *IEEE/ACM Transactions on Networking*, 25(2):724–737, Apr. 2017.

[17] R. Zhu, D. Niu, and Z. Li. Robust web service recommendation via quantile matrix factorization. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, May 2017.