A Skewness-aware Matrix Factorization Approach for Mesh-structured Cloud Services

Yongquan Fu, Dongsheng Li, Pere Barlet-Ros, Chun Huang, Zhen Huang, Siqi Shen, Huayou Su

Online cloud services need to fulfill clients' requests scalably and fast, otherwise, users' experience and providers' revenue could be severely degraded. State-of-the-art cloud services are increasingly deployed as a distributed service mesh. Service to service communication is frequent in the mesh. Unfortunately, problematic events may occur between any pair of nodes in the mesh, therefore, it is vital to maximize the network visibility for efficient network troubleshooting and application optimization.

A state-of-the-art approach is to model pairwise RTTs based on a latent factor model represented as a low-rank matrix factorization. A latent factor corresponds to a rank-1 component in the factorization model, and is shared by all node pairs. However, different node pairs usually experience a skewed set of hidden factors like divergent routing paths, dynamic path conditions, or transient network congestions, which cannot be captured by existing latent factor models.

In this paper, we propose a skewness-aware matrix factorization method named SMF. We decompose the matrix factorization into the basic units of rank-one latent factors, and progressively combine rank-one factors for different node pairs. We present a unifying framework to automatically and adaptively select the rank-one factors for each node pair, which not only preserves the low rankness of the matrix model, but also adapts to skewed network latency distributions.

Over real-world RTT data sets, SMF significantly improves the relative error by a factor of 0.2x to 10x, converges fast and stably, and compactly captures fine-grained local and global network latency structures.

I. Introduction

Large-scale cloud services are typically organized as a mesh of micro-services that are deployed over hundreds to thousands of nodes, as illustrated in Figure 1. Service-to-service communication is frequent on the mesh structured service topology. For example, a Web request may traverse thousands of servers to search and aggregate results. A request's service level agreement (SLA) is determined based on the response from the slowest server. As problematic locations are essentially unpredictable a priori, we need to track RTTs by all nodes and for all nodes [25].

Timely response is vital for ensuring the Quality of Experience (QoE) [49], otherwise, the increased delay significantly affects users' experience and providers' revenue [13], [7]. Unfortunately, high latency issues may arise between any node pairs of the service mesh, due to changing routing paths, degraded path conditions, or transient network congestions.

Yongquan Fu, Dongsheng Li, Zhen Huang, Siqi Shen, Huayou Su are with the Science and Technology Laboratory of Parallel and Distributed Processing; College of Computer, National University of Defense Technology, Changsha, Hunan province, China.

Chun Huang is with College of Computer, National University of Defense Technology, Changsha, Hunan province, China.

Pere Barlet-Ros is with UPC BarcelonaTech.

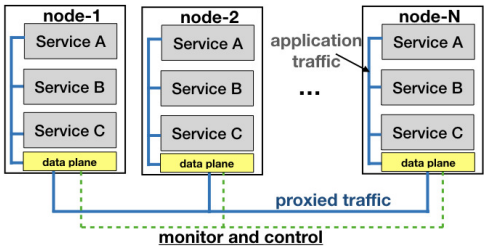


Fig. 1. An illustration of a service-mesh scheme [3], [4]. A cloud service consists of a collection of loosely-coupled micro-services hosted on a set of distributed servers. The service-to-service communication is managed by a dedicated service-mesh proxy layer for autonomous traffic control and optimization. The monitoring functionality is located at the proxy layer to collect pairwise RTT status for the service mesh layer.

As a result, in order to correctly correlate the network issue with the problematic service, it is vital to tell if a service is affected by a network issue by monitoring pairwise RTTs of the service mesh.

Meanwhile, optimizing the performance of the service mesh needs pairwise RTTs. For example, we may choose the detour routing scheme [44], [6], [35] that selects a relay on the direct routing path between a set of hosts and forwards packets via the relay. For example, Akamai SureRoute [40] and Amazon CloudFront content distribution network (CDN) [5] relay packets for end hosts.

A straightforward approach is to directly collect all-pair network latency, which requires a quadratic number of probing packets with respect to the system size, which does not scale well. Therefore, in order to provide enough network visibility for troubleshooting, it is of paramount importance to predict missing measurement results. Besides the scaling limitations, the measurement may also disturb normal application traffic, as node resources are usually shared among multiple tenants.

To reduce the probing cost, researchers have proposed network-latency prediction methods that embed the service mesh into a low-dimensional vector space and estimate the pairwise RTTs based on the vector distance of the corresponding node pair. The vector-space representation succinctly captures the pairwise network latency matrix, and needs only $O(N)$ probes to predict the full pairwise matrix for N hosts in edge data centers where PingMesh becomes ineffective. Further, the vector representation can act as the input for various applications [45], [35], [28].

The vector space is usually represented as a low-rank matrix factorization model [37], [33], [34], which represents each node as a vector of variables. These variables are "latent" in the sense that they are not directly observed, but should be inferred via a mathematical model from observed data. A

matrix factorization model is well known to be equivalently represented as the sum of a set of rank-one matrices [24]. Given a rank-one matrix represented as the product of two vectors $F_k G_k^T$ for $k \leq r$, where a constant r denotes the number of rank-one matrices, this rank-one matrix serves as a "latent factor", and the vector F_k and G_k serve as the soft memberships of each row and each column of the RTT matrix to this latent factor [15], [47]. As the same set of rank-one matrices are shared by all nodes, the matrix factorization model implicitly assumes that all node pairs are affected by the same set of latent factors. However, our analyses in Section III-C show that a real-world network latency matrix is heterogeneous and highly skewed, different node pairs are typically correlated with heterogeneous factors. Consequently, the matrix factorization should be revisited to account for skewed factors.

In this paper, we focus on improving the skewness awareness for the matrix factorization model. We propose a skewness-aware matrix factorization method we call SMF, which decomposes the matrix factorization framework into the basic units of rank-one matrices, and selectively combines the rank-one entries for each node pair to account for skewed network latency distributions.

The first challenge is how to smartly combine the rank-one matrices. Existing methods assume positive correlations between the rank-one matrices and the approximation results. We relax this assumption and consider generalized scenarios where each rank-one matrix is either positively correlated, negatively correlated or irrelevant. Then, we model these relations within a unifying combination framework, by extending the well-known orthogonal matching pursuit algorithm [41]. In each iteration, we add a new rank-one matrix into the prediction, where each entry is either added, bypassed, or decreased to account for the skewness of the network latency distribution.

As most node pairs are unobserved, we need to automate the selection decision for each entry. To that end, we treat the selection decision as the rating score and the service mesh as both clients and goods, and establish the selection decision problem as a collaborative filtering task that predicts missing rating scores between a set of clients and a set of goods. Inspired by this analogy, we propose an adaptive prediction method to map the collaborating filtering scores to discrete combination choice by extending the well-known maximum margin matrix factorization method [42], [48].

Finally, extensive experiments using real-world data sets confirm that our approach reduces the relative errors by a factor of 0.2x to 10x compared to state-of-the-art methods. SMF converges fast and stably. Applying SMF to inform the low-latency detour routing [44], [45] achieves close to optimal performance. Further, we have evaluated the parameter regions where SMF obtains good approximations and verified its computational efficiency.

In summary, we make three primary contributions in this paper:

- We quantify the skewness of the RTT metric with respect to a set of local and global metrics, which motivates novel insights to develop skewness-adaptive matrix models.
- We develop a skewness-aware matrix factorization framework SMF that automatically and adaptively selects rank-one latent factors for different node pairs.
- We perform extensive experiments on real-world data sets to confirm that that SMF finds a good balance between the low rankness and the adaptation to skewed RTT distributions.

The rest of the paper is organized as follows. Section II summarizes the related literature on the service mesh monitoring. Next, Section III presents the background and states the requirements of adapting to skewed latent factors. Next, Section IV introduces the basic ideas of the skewness-ware matrix factorization. Section V presents the detailed algorithms and analysis of the performance and the parameter choices. Section VI reports the extensive simulation experiments compared with state-of-the-art methods. Finally, we conclude in Section VII.

II. Related Work

Extensive studies have been made to enable network latency measurement for large-scale distributed systems and data center networks. We only introduce representative studies that are most related to us.

Mesh Network Monitoring: iPlane [36] predicts end to end network latency based on an Internet topology model. iPlane issues active probes from wide-area vantage points to routable network addresses. PingMesh [25] collects all-pair round-trip time (RTT) measurement system in several scales for geo-distributed data centers, which accumulates 24 TBs of probe results each day. [54] approximates network latency in an OpenFlow network environment based on control messages to and from the OpenFlow controller. Mobilyzer [39] provides a controlled and isolated library for mobile network measurement experiments. Our work is complementary to these studies by predicting missing measurements.

Network Latency Prediction: In order to reduce the measurement overhead, researchers proposed to predict network latency with network coordinate methods. GNP [38] pioneers this field by modeling the network latency matrix with an Euclidean coordinate system and fitting this coordinate system via a multidimensional scaling technique. Vivaldi [11] combines a low-rank model with a height model that reflects the first-hop delay of traversing the accessing link. IDES [37] and DMFSGD [33] embed nodes using the two-factor matrix factorization. Further, DMFSGD [33] proposes a generalized framework of the low-rank matrix factorization via the SGD optimization technique. Liu et al. [34] decompose the latency metric to a distance component and a network feature component by combining the Vivaldi and the matrix completion theory. Fu et al. [19] stabilizes the matrix factorization process under churns via the relative coordinates.

Zhu et al. [57] propose an adaptive matrix factorization approach by data transformation. The transformed metrics become more symmetric than the raw metrics, however, re-transformed metrics become less stable when the observation is incomplete, since the estimator's inaccuracy will be amplified exponentially with respect to the transformation base.

A second approach is to estimate the quantiles [59] instead of the average RTT metric based on the quantile regression framework. Generally, these methods assume that each pair of nodes shares the same set of latent factors, which may not hold when node pairs experience diverse hidden factors due to divergent routing paths, dynamic path conditions, or transient network congestions. Our work addresses this challenge via a skewness-aware matrix factorization model that selectively combines the latent factors.

Matrix Completion: Our study is related with the matrix completion theory [8], which considers the problem of recovering an incomplete matrix via a subset of observed entries. For a rank-r $m \times n$ matrix ($r \ll (m, n)$) that meets an incoherent[1] condition, a unique rank-r matrix can be recovered with a high probability. When the rank is unknown, a generalized matrix completion problem seeks to best fit a partially observed matrix with a minimum rank. Unfortunately, minimizing the matrix rank exactly is NP-hard [8]. OR1MP [47] iteratively computes a rank-one matrix of the top-left and -right singular vectors of the approximation residual. However, it is generally impossible to exactly recover the SVD result for a partially observed matrix. Further, our study shows that, not all rank-one matrices are useful to reduce the approximation residual, moreover, different node pairs are correlated with heterogeneous latent factors.

Traffic Matrix Interpolation: Real-world traffic matrices are usually incomplete. Consequently, interpolating missing entries becomes important. Note that traffic matrix interpolation is a related, but different problem, with different properties. Xie et al. [52], [51], [50] exploit hidden spatial and temporal structures with three-dimensional low-rank tensors, which effectively reduces the estimation error. Zhang et al. [56] interpolate incomplete traffic matrices with structure regularized low-rank matrix factorization and local interpolation procedures. LENS [10] models the traffic matrices as the sum of multiple matrices that are positively correlated with the traffic matrix. Our work is a complement to these studies by proposing a new model that keeps the low-rank interpretation and adapts well to skewed latent factors.

III. Problem Statement

In this section, we first present the measurement environment for the mesh-structured cloud services, then introduce the matrix factorization results, and discuss the open questions.

A. Measurement Architecture

A service mesh consists of a set of nodes located in mega data center networks or edge data-center networks. Each node hosts a set of networked micro-services, as discussed in the introduction. Service to service communication is frequent, while the latency between sending service requests and obtaining responses should meet network SLAs.

As the network issues may arise in unknown locations, it is vital to maximize the network visibility between nodes for troubleshooting and proactive network optimization. To meet these needs, it is necessary to monitor network latency between any node pairs [33], [25]. Therefore, a global view of the network conditions is needed, as the service mesh is managed under a single entity.

We organize the measurement system as two components: a data plane that consists of service-mesh nodes and a control plane on a logically centralized server, inspired by the software defined networks [36], [25], [54].

We assume that, the service mesh should have synchronized their clocks, as otherwise we could not correlate the network problems in different locations. The synchronization protocols such as Network Time Protocol (NTP) [2] or the IEEE 1588 Precise Time Protocol (PTP) [1] can provide millisecond-level precision for geo-distributed nodes.

(i) At the control plane, the logically centralized controller schedules the RTT measurement process on the data plane. To that end, the controller randomly samples a small list of nodes as *probing targets*. The choices of probing targets are randomized for different nodes for load balancing. The number of probing targets depend on the measurement capability of the data plane. For a scale of hundreds of nodes, our experiments show that tens of probing targets is sufficient to obtain a good accuracy.

Further, the controller handles the system dynamics, since an offline node is useless and should be detected and filtered by the controller. Accordingly, the controller keeps the online status of the data plane as volatile states in the main memory. Each online node periodically sends a heart-beating message to the centralized controller to notify its online status. After the controller successfully obtains a heart-beating message from a node, the controller piggybacks a list of sampled online nodes. The frequency of the heart-beat messages is platform-dependent. Stable platforms with dedicated nodes could choose a long period, while edge platforms should choose a relatively short period to reflect system churns.

(ii) At the data plane, each service-mesh node performs a number of measurements towards other nodes in the same service mesh. It downloads the list of probing targets from the controller, and measures the round-trip time (RTT) towards these probing targets in a periodical approach. After collecting the RTT samples in an interval, each node uploads the RTT results to the persistent storage that is accessed by the controller.

The data plane could use any kinds of measurement methods. For example, at the network or transport level, the data plane may choose ICMP or TCP protocol based measurement methods; while at the application level, the data plane could use RPC or HTTP protocol based methods. Generally, the RTT value amounts to the absolute difference between the time of sending a request message to the probing target and that of receiving the response message from this probing target.

The unit of a measurement interval determines the granularity of the monitoring process. Increasing the sampling interval towards a probing target yields a coarser measurement granularity. Generally, the time interval should be large enough to obtain the response from the slowest probing target in the

[1] A well-known assumption of recovering a low-rank r matrix is *incoherence* that spread the singular vectors out, which keeps the matrix not to be closely aligned with the coordinate axes.

system, otherwise, a data-plane node may not collect the RTT samples from some of its probing targets.

B. Challenges for RTT Matrix Completion

For a set of N nodes, the pairwise RTTs between N nodes in an interval can be represented as a N-by-N matrix $\mathbf{D}$. The state-of-the-art approaches predict pairwise RTT values based on the matrix factorization approach, which factorizes a matrix $\mathbf{D} \in R^{N\times N}$ as a product of two low-dimensional factor matrices $\mathbf{F} \in R^{N\times r}$ and $\mathbf{G} \in R^{N\times r}$, i.e., $\mathbf{D} \approx \mathbf{F}\mathbf{G}^T$, where $r \ll N$, and ^{T} denotes the transpose of a matrix.

Generally, a matrix factorization model is equivalent to a sum of a set of rank-one matrices:

$$\hat{\mathbf{D}} = FG^T = \sum_k F_k G_k^T \tag{1}$$

where F_k, G_k denote the k-th ($k \le r$) column vector of the matrix F and G, respectively. The objective function seeks to minimize the approximation residual between the observed entries and the sum of the rank-one matrices:

$$\min_{F,G} \left\| \mathbf{D} - \sum_{k=1}^{r} F_{*k} G_{*k}^T \right\| \tag{2}$$

As the matrix factorization approximates the target matrix $\mathbf{D}$ via the sum of each rank-one matrix $F_k G_k^T$, each of which is assumed to be positively correlated with the approximation results. Consequently, the matrix factorization assumes that all node pairs experience the same set of latent factors.

Unfortunately, this assumption may not hold for the RTT metric, which additively consists of many unobservable factors, e.g., the propagation latency, the queueing latency, the transmission latency. The additive character implies that, for two node pairs with a number of different routing links, their RTTs are likely to be affected by independent latency components, or hidden factors as they are invisible from the end. Consequently, different node pairs are likely to experience a diverse set of latent factors that are unobservable from the end to end measurement, as the RTT metric only reveals the sum of all factors, not individual factors.

C. Empirical Distributions

Next, we empirically analyze the RTT characteristics of real-world data sets and motivate the prediction requirements.

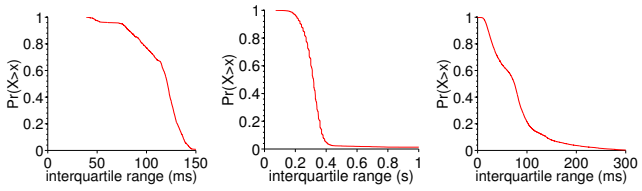
1) Data Set

Real-world network connections are heterogeneous, therefore, an ideal network latency prediction algorithm should be resilient to different network connections. We choose three kinds of publicly available data sets different in terms of scale and distributions, which provide an ideal benchmark to study the generalized performance of different prediction algorithms:

- **Seattle**: The Seattle platform is an open peer-to-peer cloud computing platform that includes donated personal devices like personal computers, laptops and mobile phones [9] This data set was collected in summer 2014 for three hours between 99 nodes. Each interval aggregates pairwise RTT samples within 15.7 seconds, which indicates short-term dynamics between Seattle nodes.

TABLE I
BASIC STATISTICS OF DATA SETS.

Trace	Interval	Mean	STD	Min	Max
Seattle	15.7s	0.37s	0.90s	0.01s	90.50s
PlanetLab	14.7-hr	147.38ms	103.65ms	0.064ms	7892.8ms
RIPE	2-hr	118.47ms	106.02ms	0.08ms	10,425.75ms



(a) PlanetLab (b) Seattle (c) RIPE

Fig. 2. The CCDFs of interquartile ranges of each row vector in the dataset.

- **PlanetLab**: The PlanetLab platform has been widely used in many previous network prediction studies [36], [33], [19]. This data set was collected in 2013 for a 9-day period between 490 distributed nodes. Each interval aggregates pairwise RTT measurements within 14.7 hours, which represents long-term RTT trends between PlanetLab nodes.
- **RIPE**: The RIPE Atlas measurement platform consists of tiny networked devices that issue measurements to a small number of addresses, most of which are chosen by the platform owner. There are two kinds of devices, i.e., Probes and Anchors, depending on their measurement capability. The Anchor is more powerful than the Probe. As we need dense RTT matrices to train the matrix factorization methods, we choose RIPE Atlas nodes that are powerful enough to probe each other. This data set[2] was collected in August 19, 2018 between 250 RIPE Atlas nodes.

Table I summarizes several basic statistics about the data sets. We can see that the data sets span wide ranges.

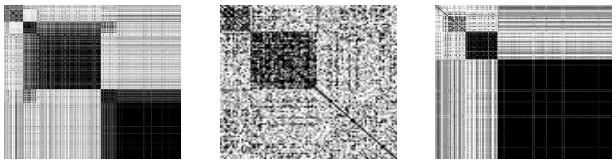
2) Distributions

First, we compare the RTT distributions for each node. We calculate the interquartile ranges for each RTT vector, i.e., the difference between the 25th and the 75th percentiles of the samples in the vector. In Figure 2, we see that the interquartile ranges span wide intervals, thus different node pairs experience divergent hidden factors.

Next, we evaluate the correlation between pairs of nodes. We compute the linear correlation coefficient [43] for any pair of nodes i and j, defined as $c(i,j) = \frac{\langle \vec{D_i} \circ \vec{D_j} \rangle - \langle \vec{D_i} \rangle \langle \vec{D_j} \rangle}{\sigma(\vec{D_i})\sigma(\vec{D_j})}$, where D denotes the pairwise RTT matrix, $\vec{D_j}$ denotes the i-th row vector, $\circ$ denotes the hadamard operator $\left(\vec{D_i} \circ \vec{D_j}\right)_k = D_{ik}D_{jk}$ for $i,j,k \in [1,N]$, $\langle\cdot\rangle$ denotes the average of the vector, and $\sigma\left(\vec{D_i}\right) = \sqrt{\left\langle \vec{D_i} \circ \vec{D_i} \right\rangle - \left\langle \vec{D_i} \right\rangle^2}$.

Figure 3 plots the heat map of the linear correlation coefficients of each node pair. Darker pixels correspond to larger correlation coefficients. We can see that two to three groups

[2]https://data-store.ripe.net/datasets/atlas-daily-dumps/



(a) PlanetLab (b) Seattle (c) RIPE

Fig. 3. Heat maps of the pairwise correlation coefficients. We choose the first interval for illustration purpose. Varying the intervals yields the same conclusions. We randomly select a number of nodes from the data set as the landmarks, calculate the vector of correlation coefficients from this node to the landmarks. Then we compute the K-means clustering with these feature vectors based on the Lloyd method [53]. Finally, we reorganize the correlation coefficient matrix by putting nodes in the same cluster at adjacent positions, and plot the heat map of the matrix where darker pixels correspond to larger correlation values. During the experiments, we set the number of landmarks to 16, the number of clusters to three. The same conclusions hold as we vary the parameters.

TABLE II
FREQUENTLY USED NOTATIONS IN THIS PAPER.

Notations	Meaning
N	Number of nodes
$\mathbf{D}$, $\hat{\mathbf{D}}$	RTT matrix and the estimated RTT matrix
$\mathbf{X}$	rank-1 matrix
β, $\hat{\beta}$	Sign matrix and the estimated sign matrix
$\mathbf{E}$	Approximation residual
F, G	rank-1 matrix model
$(\vec{u}, \vec{v}, \vec{\theta}, b)$	Sign matrix model
r	Approximation rank
S_i	Probing targets of node i
n_p	Number of probing targets
r_s	Dimension of the estimated sign matrix
λ	Regularized parameter

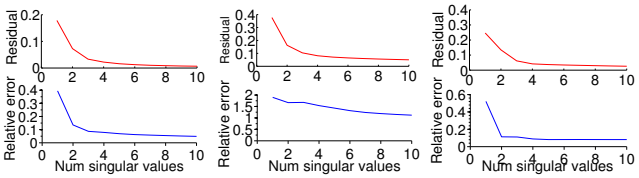
of nodes are separable from the rest of the plot, where intra-group correlation coefficients are relatively larger than node pairs from different groups. As a result, the pairwise RTT distribution is highly skewed, and intra-group nodes are more likely to experience similar latent factors than inter-group node pairs. Accordingly, we need a powerful representative model to account for skewed latent factor distributions in the RTT matrix.

IV. SKEWNESS-AWARE MATRIX FACTORIZATION

Having analyzed the RTT characteristics of three data sets, the RTTs of different node pairs are likely to be affected by divergent hidden factors, which should be accounted by the prediction algorithm.

Next, we first present naive approaches to account for divergent latent factors and discuss its limitations, then present a new matrix factorization model that keeps the interpretation of the low-rank representation and adapts to the skewness of the RTT distributions.

Table II summarizes frequently used notations used in this paper.



(a) PlanetLab (b) Seattle (c) RIPE

Fig. 4. Energy contained in the top-k singular values VS. the relative error of the rank-k approximation using the SVD method. For a squared and complete matrix $\mathbf{D}$, SVD calculates a Frobenius-norm optimal approximation with respect to $\mathbf{D}$ [24]. The SVD represents $\mathbf{D}$ as $\mathbf{D} = USV^T$, where V^T denotes the transpose of V, $U \in R^{N\times N}$ is an orthogonal matrix, i.e., $UU^T = U^TU = I$, $V \in R^{N\times N}$ is also an orthogonal matrix, i.e., $VV^T = V^TV = I$, S is a squared diagonal matrix consisting of a vector of descending-ordered real numbers $(\sigma_1, \cdots, \sigma_N)$, where $\sigma_i \geq 0$ for $i \leq N$.

A. Strawman Approaches

1) Increasing Rank

A straightforward approach is to increase the rank of the matrix factorization model. In mathematics, the rank of a matrix is the maximal number of columns or rows that are linearly independent with each other, which is neatly characterized by the number of positive singular values of the SVD of this matrix [24]. The SVD can be equivalently represented as the sum of k rank-1 matrices $\hat{D} = \sum_{i=1}^{k} \delta_i u_i v_i^T$, where δ_i is the i-th singular value, u_i is the i-th left singular vector, and v_i is the i-th right singular vector. Each rank-1 matrix represents a latent factor that introduces a degree of freedom to approximate the matrix.

We compute the **relative error** $\frac{|\mathbf{D}_{ij} - \hat{\mathbf{D}}^r_{ij}|}{\mathbf{D}_{ij}}$ for each node pair between the rank-k approximation $\hat{\mathbf{D}}^r$ and the RTT matrix $\mathbf{D}$. Further, we compute residual fraction of the total variance captured by top-k singular values as $1 - \frac{\sum_i^k \delta_i^2}{\sum_i^N \delta_i^2}$, where δ_i represents the i-th largest singular value.

Figure 4 shows the residual variance and the relative error with increasing number of top singular values. First, 90% of its variance can be captured with two to four top singular values, therefore, the RTT matrix is approximately low-rank. Second, the relative error is still high. A longer tail of singular values implies a higher relative error, since the residual of the SVD approximation is correlated with the remaining set of singular values, represented as $D - \hat{D}_k = \sum_{i=1}^{N} \delta_i u_i v_i^T - \sum_{i=1}^{k} \delta_i u_i v_i^T = \sum_{i=k+1}^{N} \delta_i u_i v_i^T$.

2) Incorporating Weights

A second approach is to regularize the latent factor model for different node pairs. For example, let $\hat{D}'[i,j] = \hat{D}[i,j] \cdot w[i,j]$, where w denotes the weight matrix. The estimation of each node pair is scaled independently, which introduces N^2 degrees of freedom to the prediction $\hat{D}$. Accordingly, the weighted model has enough degrees of freedom to recover any matrix exactly. For example, [58] proposes to assign an N-by-N real-valued weighted matrix to the low-rank matrix, and estimates the weight matrix in a matrix completion framework. As [58] uses a single weight matrix to regularize every latent factor, it assumes that each latent factor is of equal importance for each node pair, which may not hold due to complex routing decisions and varying latency components.

Instead of assigning real-valued weights to low-rank models, we propose to decompose the matrix factorization framework into the basic units of rank-1 matrices, and selectively combine the rank-1 entries for each node pair, inspired by the well-known orthogonal matching pursuit algorithm [41].

B. Ideal Skewness-aware Model

Next, we present an ideal model to account for skewed latent factors assuming that we obtain the complete RTT matrix. In the next subsection, we relax this assumption and present a practical approach.

Recall that matrix factorization is equivalent to the sum of rank-1 matrices, while each rank-1 matrix serves as a latent factor. To make the matrix factorization be aware of skewed latent factors, we should incorporate a rank-1 matrix entry for a node pair only if this node pair is correlated with this latent factor.

1) Correlation Model

Let k denote the index of the current rank-1 matrix ($1 \leq k \leq r$). Let $\hat{\mathbf{D}}_k$ be the approximation result of combining the first k rank-1 matrices, and $\mathbf{E}_k = \mathbf{D} - \hat{\mathbf{D}}_{k-1}$ the current residual, where $\hat{\mathbf{D}}_0$ represents an empty matrix.

We define three kinds of correlation types to combine the latent factor for each node pair (i, j) as follows:

(i) Positive correlation: If the approximation residual $\mathbf{E}_k[i,j]$ will be smaller by adding the rank-1 entry $\mathbf{X}_k[i,j]$ to the current approximation $\hat{\mathbf{D}}_{k-1}[i,j]$, i.e., $\mathbf{D} - \left(\hat{\mathbf{D}}_{k-1}[i,j] + \mathbf{X}_k[i,j]\right) < \mathbf{E}_k[i,j]$, then we set the rank-1 entry $\mathbf{X}_k[i,j]$ to be **positively correlated** with the current approximation residual.

Accordingly, we update the approximation for the node pair i, j as: $\hat{\mathbf{D}}_k[i,j] = \hat{\mathbf{D}}_{k-1}[i,j] + \mathbf{X}_k[i,j]$.

(ii) Negative correlation: If the approximation residual $\mathbf{E}_k[i,j]$ will be smaller if we subtract the rank-1 entry $\mathbf{X}_k[i,j]$ from the current approximation $\hat{\mathbf{D}}_{k-1}[i,j]$, i.e., $\mathbf{D} - \left(\hat{\mathbf{D}}_{k-1}[i,j] - \mathbf{X}_k[i,j]\right) < \mathbf{E}_k[i,j]$, then we set the rank-1 entry $\mathbf{X}_k[i,j]$ to be **negatively correlated** with the approximation residual.

To reduce the approximation error, we update the approximation as: $\hat{\mathbf{D}}_k[i,j] = \hat{\mathbf{D}}_{k-1}[i,j] - \mathbf{X}_k[i,j]$.

(iii) Irrelevance: Finally, if neither positive nor negative correlations hold, we should skip this rank-1 component $\mathbf{X}_k[i,j]$ for node pair i, j, since otherwise, the approximation residual will increase. In other words, this rank-1 entry $\mathbf{X}_k[i,j]$ is *irrelevant* with the current approximation residual.

Accordingly, the approximation residual either decreases monotonically (positive and negative correlations) or keeps the current status (irrelevance). Thus we not only keep the interpretation of the matrix factorization, but also account for the skewed latent factors.

2) Algebraic Representation Model

To unify the correlation model with the estimation purpose, we represent the correlation model in a compact format. As the "positive correlation, negative correlation, irrelevance" choices are equivalent to three discrete choices (+1, -1, 0), respectively, we summarize them in a pairwise matrix (denoted as a **sign matrix**) $\beta \in \{+1, -1, 0\}^{N \times N}$.

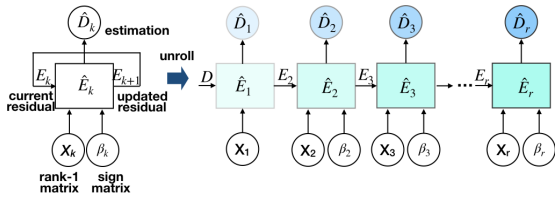


Fig. 5. The diagram of the ideal correlation model.

Consequently, we transform the choice of the correlation model from a combination optimization problem to a linear algebraic problem. Now we can directly represent the k-th approximation residual $\hat{E}_k = \beta_k \circ \mathbf{X}_k$. The overall estimation amounts to $\hat{D} = \sum_{k=1}^{r} \hat{E}_k = \sum_{k=1}^{r} \beta_k \circ X_k$.

3) Workflow

The ideal model can be represented as a recurrent framework that predicts the residuals layer by layer, as illustrated in Figure 5. Each recurrence aims to find a skewness-aware rank-1 matrix to minimize the current residuals. The model sequentially finds a rank-1 matrix with respect to the residual and a sign matrix according to the correlation types. Then, it combines the rank-1 matrix and the sign matrix to approximate the residual. Finally, the overall estimation is updated and the refreshed residual is forwarded to the next recurrence.

Lemma 1 shows that the ideal model either decreases the approximation residual monotonically or keeps the current status in each layer. The proof is put in the appendix.

Lemma 1. *Assume that we obtain the perfect sign matrix, for all* $k \geq 1$, $\|\mathbf{E}_{k+1}\| \leq \|\mathbf{E}_k\|$

Unfortunately, manually deciding the selection choices is only possible for observed RTTs, but impossible for unobserved node pairs. To address this challenge, SMF predicts the selection choices to make the ideal model practical.

C. Our Work

1) Overview

SMF implements the ideal model with a layerwise learning framework. SMF unrolls the recurrent framework of the ideal model to a chained sequence of layers, and trains the model layer by layer similar to the stacking architecture of the deep neural network [27].

Each layer k takes the partially-observed residual as the input, and computes a new rank-1 matrix by maximizing the correlation with the current residual, then combines each entry of this rank-1 matrix into the current approximation based on the sign matrix, both of which are learnt from partial observations. Next, SMF calculates the approximation residual at this layer, and finally refreshes the residual and forwards that to the next layer until reaching the final layer.

SMF improves the matrix factorization in several aspects:

- **Explainability**: SMF progressively finds a rank-1 matrix to best explain the residual in each layer. Further, higher layers' rank-1 matrices are trained based on the residuals with respect to lower layers' rank-1 matrices. Thus there exists no ambiguity for the rank-1 matrices.

- **What-if Analysis**: SMF optimizes a separate rank-1 matrix for each layer, and sequentially combines them with the sign matrices to produce the residual approximation. Thus SMF enables the operator to test the choice of rank based on the layerwise RTT approximation from one to r.
- **Robustness**: SMF formulates the model with respect to the residual in each layer, which is known to be robust to the gradient-vanishing problem, as proved by the ResNet method [26].
- **Modularity**: SMF divides each layer to two conditionally independent optimization problems and combines layer-wise estimation into a modular framework.

2) SMF Components

(i) **Rank-1 Matrix**

As each rank-1 matrix amounts to a product of two vectors [24], [47], we represent the rank-1 matrix $\mathbf{X}_k$ as $\mathbf{X}_k = F_k(G_k)^T$, where F_k, G_k denote two vectors of length N. Moreover, to keep the latent factor interpretable, we enforce the rank-1 matrix $\mathbf{X}_k$ to be nonnegative by approximating the absolute-value of the residual.

When the residual matrix is completely observed, the optimal vectors can be calculated based on the SVD of E_k [24], [47]: $\mathbf{X}_k = \delta_1^k u_1^k (v_1^k)^T$, where δ_1^k, u_1^k and v_1^k denote the largest singular value, the leftmost and the rightmost singular vectors of E_k. While when the observation is incomplete, we formulate a rank-1 matrix factorization problem to estimate the vectors F_k and G_k.

(ii) **Sign Matrix**

We utilize the analogy between the sign matrix estimation and the collaborative filtering problem that recovers missing discrete rating scores (e.g., one to five stars) between a set of clients and a set of goods, and predict the sign matrix β_k with a Maximum Margin Matrix Factorization (MMMF) [42], [18].

We represent pairwise signs with a low-dimensional coordinate model. We assign each node a r_s-dimensioned coordinate $(\vec{u}_i, \vec{v}_i)$ and a threshold vector $\vec{\theta}_i$ that serves as boundaries to obtain the discrete signs. We make the coordinate model tolerate skewed distributions by incorporating a bias parameter to each node similar to [48], which captures the local factors in each machine.

Let each node i keep a bias variable b_i, the coordinate distance Y_{ij} from node i to node j is calculated as the sum of the vector dot-product result plus the bias scalars:

$$Y_{ij} = \vec{u}_i \vec{v}_j + b_i + b_j \tag{3}$$

For a pair of nodes i and j, we represent the sign from node i to node j by mapping the coordinate distance $\vec{u}_i \vec{v}_j$ to "-1", 0, "+1" using node i's threshold values $\vec{\theta}_i$:

- If $\vec{u}_i \vec{v}_j \leq \vec{\theta}_i(1)$, the estimated sign $\widehat{\beta}_{ij}$ = -1;
- If $\vec{\theta}_i(1) \leq \vec{u}_i \vec{v}_j \leq \vec{\theta}_i(2)$, $\widehat{\beta}_{ij}$ = 0;
- Otherwise, $\widehat{\beta}_{ij}$ = +1.

3) Example

We provide a simple example to illustrate the SMF algorithm. Note that our purpose here is not to characterize the performance of the SMF.

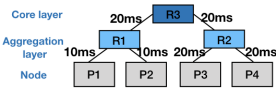


Fig. 6. Four nodes organized in a hierarchical topology.

Given four nodes as shown in Figure 6 with an RTT matrix:

$$D = \begin{bmatrix} 0 & 20 & 80 & 80 \\ 20 & 0 & 80 & 80 \\ 80 & 80 & 0 & 40 \\ 80 & 80 & 40 & 0 \end{bmatrix}$$

For example, we build a sampled RTT matrix to model the partial choices of probing targets as follows:

$$D' = \begin{bmatrix} 0 & 20 & 0 & 80 \\ 20 & 0 & 80 & 0 \\ 0 & 80 & 0 & 40 \\ 80 & 0 & 40 & 0 \end{bmatrix}$$

We compute a rank-1 matrix with respect to the illustrative matrix D', which gives $F_1 = (-6.6791, -6.6791, -7.1096, -7.1096)$, $G_1 = (-6.6791, -6.6791, -7.1096, -7.1096)$. The first two nodes and the last two nodes are identical with each other, respectively. Therefore, the rank-1 components captures the global proximity index. Further, F_1 and G_1 are identical, as the RTT matrix is symmetric.

Multiplying F_1 and G_1 yields the rank-1 matrix:

$$X_1 = \begin{bmatrix} 0 & 44.6102 & 47.4854 & 47.4854 \\ 44.6102 & 0 & 47.4854 & 47.4854 \\ 47.4854 & 47.4854 & 0 & 50.5459 \\ 47.4854 & 47.4854 & 50.5459 & 0 \end{bmatrix}$$

Next, we calculate the residual E_1 with respect to the RTT matrix D as follows:

$$E_1 = \begin{bmatrix} 0 & -24.6102 & 32.5146 & 32.5146 \\ -24.6102 & 0 & 32.5146 & 32.5146 \\ 32.5146 & 32.5146 & 0 & -10.5459 \\ 32.5146 & 32.5146 & -10.5459 & 0 \end{bmatrix}$$

Some RTTs are overestimated, while the others are overestimated, therefore, it is vital to selectively refine the residual.

Given the partially observed residual matrix E_1':

$$E_1' = \begin{bmatrix} 0 & -24.6102 & 0 & 32.5146 \\ -24.6102 & 0 & 32.5146 & 0 \\ 0 & 32.5146 & 0 & -10.5459 \\ 32.5146 & 0 & -10.5459 & 0 \end{bmatrix}$$

, we find a new rank-1 matrix similar to step one, which gives: $F_2 = (-4.7934, -4.7934, -4.3030, -4.3030)$, $G_2 = (-4.7934, -4.7934, -4.3030, -4.3030)$.

The new rank-1 matrix amounts to:

$$X_2 = \begin{bmatrix} 0 & 22.9771 & 20.6263 & 20.6263 \\ 22.9771 & 0 & 20.6263 & 20.6263 \\ 20.6263 & 20.6263 & 0 & 18.5161 \\ 20.6263 & 20.6263 & 18.5161 & 0 \end{bmatrix}$$

In order to automate the combination choices, we extract the sign matrix with respect to the partially observed residual matrix E_1':

$$\beta = \begin{bmatrix} 0 & -1 & 0 & 1 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix}$$

based on Algorithm 2, and predict the sign matrix via the MMMF framework, which gives:

$$U: \begin{bmatrix} 0.6074 & -0.3540 \\ 0.8816 & -0.1533 \\ -1.0691 & -0.2674 \\ -0.5347 & 0.5162 \end{bmatrix}, V: \begin{bmatrix} -0.8211 & -0.1208 \\ -0.6933 & 0.1867 \\ 0.8124 & -0.9372 \\ 0.3635 & 0.0986 \end{bmatrix}, \theta: \begin{bmatrix} 0.3265 & 0.3686 \\ 0.2530 & 0.2870 \\ 0.3905 & 0.5373 \\ 0.0705 & 0.0771 \end{bmatrix}$$

and the bias vector (0.1988 ,0.1186 ,-0.0130 ,0.6004).

The coordinate distance matrix can be derived by Eq (3) as:

$$\hat{Y} = \begin{bmatrix} 0 & -0.1698 & 1.0110 & 0.9851 \\ -0.3880 & 0 & 0.9655 & 1.0243 \\ 1.0959 & 0.7969 & 0 & 0.1724 \\ 1.1759 & 1.1861 & -0.3308 & 0 \end{bmatrix}$$

Fig. 7. The diagram of the SMF method.

Next, we automatically infer the sign from the coordinate distance matrix via the threshold:

$$\hat{\beta} = \begin{bmatrix} 0 & -1 & 1 & 1 \\ -1 & 0 & 1 & 1 \\ 1 & 1 & 0 & -1 \\ 1 & 1 & -1 & 0 \end{bmatrix}$$

which accurately recovers the sign matrix.

Based on the estimated rank-1 matrix and the sign matrix, we obtain the selective combination $X_2 \circ \beta$ as:

$$X_2 \circ \beta = \begin{bmatrix} 0 & -22.9771 & 20.6263 & 20.6263 \\ -22.9771 & 0 & 20.6263 & 20.6263 \\ 20.6263 & 20.6263 & 0 & -18.5161 \\ 20.6263 & 20.6263 & -18.5161 & 0 \end{bmatrix}$$

Plugging this rank-1 approximation to the rank-1 approximation, yields

$$\hat{D}_2 = \begin{bmatrix} 0 & 21.6331 & 68.1117 & 68.1117 \\ 21.6331 & 0 & 68.1117 & 68.1117 \\ 68.1117 & 68.1117 & 0 & 32.0298 \\ 68.1117 & 68.1117 & 32.0298 & 0 \end{bmatrix}$$

The estimation matrix $\hat{D}_2$ is much more accurate than the rank-1 estimation X_1. Thus the selective combination effectively penalizes the approximation residuals.

V. Optimization Methods

Having discussed SMF's model, we next present optimization details of the skewness-aware matrix factorization.

As stated in Section III-A, the monitoring framework consists of a logically centralized controller in the control plane and distributed service-mesh nodes in the data plane. The controller predicts missing measurements in each interval: It aggregates the measurements of an interval as a partially observed matrix, and runs SMF to estimate missing matrix entries.

A. Optimization Workflow

We summarize the workflow of SMF in Figure 7:

(i) For the first layer, let $\mathbf{E}_1$ be the partially observed RTT matrix $\mathbf{D}$, which consists of aggregated RTT samples of an interval collected from the data plane (S1).

(ii) We compute a rank-1 matrix $\mathbf{X}_k$ (k is initialized as one for the first layer) to maximize the correlation with the residual $\mathbf{E}_k$ ($\mathbf{E}_1$ is initialized to the partially observed matrix $\mathbf{D}$ for the first layer) (S2).

(iii) Next, we compute a sign matrix β_k to determine the combination choices of the current rank-1 matrix $\mathbf{X}_k$ to minimize the approximation error of the residual $\mathbf{E}_k$ (S3).

(iv) We approximate the residual as the dot product of these two matrices (S4): $\hat{\mathbf{E}}_k = \mathbf{X}_k \circ \beta_k$, and update the accumulated RTT prediction as: $\hat{\mathbf{D}}_k = \hat{\mathbf{D}}_{k-1} + \hat{\mathbf{E}}_k = \sum_{l=1}^{k} \mathbf{X}_l \circ \beta_l$, where $\circ$ represents the element-wise product operator. Next, we update the residual for each observed entry (i, j): $\mathbf{E}_{k+1}[i,j] = \mathbf{D}[i,j] - \hat{\mathbf{D}}_k[i,j]$.

(v) Let $k \leftarrow k+1$. If $k < r$, then we move to the next layer until reaching the final layer; otherwise, we stop and output the approximation as: $\hat{\mathbf{D}} = \sum_{l=1}^{r} \mathbf{X}_l \circ \beta_l$.

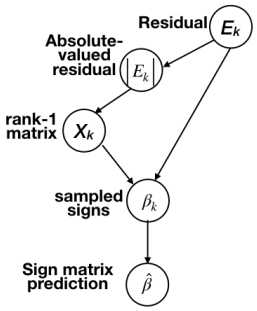


Fig. 8. Graphical model of key variables in the optimization workflow.

B. Loss Function

Based on the optimization workflow, we see that the rank-1 matrix is optimized with respect to the residual from the last layer, while the sign matrix is optimized with respect to the observed sign samples, which is calculated based on the correlation type between the residual and the approximated rank-1 matrix. Thus, each layer optimizes two conditionally independent optimization problem, as clearly shown in Figure 8.

Therefore, we separate the optimization problem to decomposed subproblems and design modular optimization workflow, due to the conditional independence between the rank-1 matrix and the sign matrix.

1) Rank-1 Matrix

The input to the rank-1 matrix completion is the partially observed residual. We compute the residual based on prediction $\hat{\mathbf{D}}_{k-1}$ from the first to the $k-1$-th layer:

$$\mathbf{E}_k[i,j] = \mathbf{D}[i,j] - \hat{\mathbf{D}}_{k-1}[i,j] \tag{4}$$

for $(i,j) \in$ Observed samples and $k \geq 2$.

To find the rank-1 matrix for the k-th layer, we minimize the difference between the rank-1 matrix and the absolute-valued residual $\mathbf{E}_k$ as:

$$\min \left\| F_k (G_k)^T - |\mathbf{E}_k| \right\|_F^2 \tag{5}$$

We formulate a regularized objective function for Eq (5) to avoid the overfitting issue [60], [21]:

$$\min J_k = \left\| F_k (G_k)^T - |\mathbf{E}_k| \right\|_F^2 + \lambda \left\| F_k^T F_k - G_k^T G_k \right\|_F^2 \tag{6}$$

where λ denotes a regularized factor. The regularization $\left\| F_k^T F_k - G_k^T G_k \right\|_F^2$ addresses the scaling ambiguity as $F_k (G_k)^T = (F_k R)(G_k R)^T$ holds for any orthogonal matrix R [60]: Suppose J_k is defined in Eq (6), any stationary point (F_k, G_k) of J_k with $\nabla J(F_k, G_k) = 0$ implies that $F_k^T F_k = G_k^T G$.

2) Sign Matrix

We extract the signs of observed node pairs as the input to the sign prediction problem and predict signs of for unobserved node pairs. The sign matrix is vital to keep the combined prediction monotonically decreasing.

We next present a simple algorithm to set the sign matrix β that is consistent with the selective combination rules (presented in Algorithm 2 as shown in the Appendix B):

First, if adding the new component reduces the absolute-valued residual, then $\mathbf{X}_{ij}$ is positively correlated, so we set the corresponding sign to "+1"; second, if subtracting the new component reduces the absolute-valued residual, then $\mathbf{X}_{ij}$ is negatively correlated, so we set the sign to "-1"; otherwise, we set the sign to "0".

Generally, let the ground-truth sign $\beta[i,j]$ of each observed node pair (i,j) be shifted to the interval $\{1,2,3\}$ for ease of performing the maximum-margin matrix factorization (MMMF). We formulate a separable soft-margin classification error $L\left(Y,\vec{\theta}\right)$ for each pair of observed node pair $(i,\ j)$ [42], [14], [18]:

$$\begin{aligned} &L\left(\vec{u}_i,\vec{v}_j,b_i,b_j,\vec{\theta}_i\right)=\sum_{c=1}^{\beta_{ij}-1}h\left(\vec{u}_i\cdot\vec{v}_j+b_i+b_j-\vec{\theta}_i\left(c\right)\right)+\\ &\sum_{c=\beta_{ij}}^{2}h\left(\vec{\theta}_i\left(c\right)-\left(\vec{u}_i\cdot\vec{v}_j+b_i+b_j\right)\right)\\ &=\sum_{c=1}^{2}h\left(T_{ij}^{c}\left[\beta_{ij}\right]\cdot\left(\theta_{ic}-\left(\vec{u}_i\cdot\vec{v}_j+b_i+b_j\right)\right)\right) \end{aligned} \tag{7}$$

where $\theta_{ic}=\vec{\theta}_i\left(c\right)$, $T_{ij}^{c}\left[\beta_{ij}\right]=\left\{\begin{smallmatrix}+1 & c\geq\beta_{ij}\\ -1 & c<\beta_{ij}\end{smallmatrix}\right.$ serves as an indicator function for the sign value β_{ij}, and $h\left(z\right)=\left\{\begin{smallmatrix}\frac{1}{2}-z & z<0\\ 0 & z>1\\ \frac{1}{2}\left(1-z\right)^2 & otheriwse\end{smallmatrix}\right.$ represent a smoothed hinge loss with derivative: $h'\left(z\right)=\left\{\begin{smallmatrix}-1 & z<0\\ 0 & z>1\\ z-1 & otheriwse\end{smallmatrix}\right.$ Thus, we can optimize the loss function with gradient based optimization techniques.

Further, We present a regularized loss function with respect to Eq (7) for each node i [42], [14], [18] for robustness against outliers and missing items:

$$g_k(x_i)=\sum_{\in S_i}L\left(\vec{u}_i,\vec{v}_j,b_i,b_j,\vec{\theta}_i\right)+\lambda\left(\left\|\vec{u_i}\right\|_F^2+\left\|\vec{v_i}\right\|_F^2+\left\|b_i\right\|_F^2\right) \tag{8}$$

where $x_i=\left[\vec{u}_i;\vec{v}_i;b_i;\vec{\theta}_i\right]$, λ denotes the regularization constant.

C. Algorithm

The key building block of each layer is the optimization of the rank-1 matrix and that of the sign matrix, both of which belong to the non-convex matrix factorization problem.

1) Rank-1 Matrix Optimization

First, we seek a new rank-1 matrix $\mathbf{X}_k$, which needs to be maximally correlated with the approximation residual. We compute the rank-1 matrix with the stochastic gradient descendent method that onverges fast to the vicinity of the optimum regions and helps escape the local minimum for distributed optimization [22], [31], [46], [29], [23], [21].

We optimize Eq (6) with the stochastic gradient descendent method. Let $F_k(i)$ and $G_k(i)$ denote the row vectors of node i. We optimize each node i's row vectors in T rounds, whereas in each round t, we compute the partial gradients of the row vectors with respect to probed targets of node i, and then adjust the row vectors towards the approximated negative gradient direction scaled by a learning-rate parameter η_1, which is automatically chosen using the line search method [33]. We summarize the optimization steps in Algorithm 3 (Appendix B).

Algorithm 1: Optimization procedures of the SMF algorithm.

1 `SMF` ($\mathbf{D}$, r)
input : Partially observed RTT matrix $\mathbf{D}$, rank r
output: Estimated RTT matrix $\hat{\mathbf{D}}$
2 Compute a rank-1 matrix $\mathbf{X}_1=F_1(G_1)^T$ that is maximally correlated with the partially observed RTT matrix $\mathbf{D}$ by calling `RankOne` ($\mathbf{D}$) in Algorithm 3 (Appendix B);
3 Set the RTT prediction as $\hat{\mathbf{D}}_1=\mathbf{X}_1$;
4 **for** *Integer* $k\in[2,r]$ **do**
5 Compute the partially observed residual $\mathbf{E}_k=\mathbf{D}-\hat{\mathbf{D}}_{k-1}$;
6 Find a rank-1 matrix $\mathbf{X}_k=F_k(G_k)^T$ with respect to $\mathbf{E}_k$ by calling `RankOne` ($|E_k|$) in Algorithm 3 (Appendix B);
7 Calculate the partially observed sign matrix $\beta_{\mathbf{k}}$ by calling `SignRule` ($\mathbf{E}_k[i,j]$, $\mathbf{X}_{ij}$) in Algorithm 2 (Appendix B);
8 Find a sign matrix $\hat{\beta}_k$ by calling `SignMatrix` ($\beta_{\mathbf{k}}$) in Algorithm 4 (Appendix B);
9 Update the RTT prediction $\hat{\mathbf{D}}_k=\hat{\mathbf{D}}_{k-1}+\mathbf{X}_k\circ\hat{\beta}_k$;
10 Set $k\leftarrow k+1$;
11 **return** $\hat{\mathbf{D}}=\sum_{k=1}^{r}F_k(G_k)^T\circ\hat{\beta}_k$;

2) Sign Matrix Optimization

Second, we seek a sign matrix that determines the correlation types for each node pair between the residual and the current rank-1 matrix. We compute the sign matrix based on the MMMF framework [42] that adaptively penalizes the difference between the estimated signs and the ground-truth ones.

We optimize Eq (8) with the nonlinear conjugate-gradient optimization [42], [18]. We form a vector x_i as the concatenation of the coordinate components of i, i.e., $x_i(s)=\left[\vec{u_i};\vec{v_i};\vec{\theta}_i;b_i\right]$, and adjust each node's vector x_i in T rounds, whereas in each round s ($s\geq1$), we compute the conjugate direction of the vector x_i with respect to probed targets, and move the vector x_i a small distance in the direction of the conjugate direction scaled by a learning-rate parameter η_2, which is automatically computed to meet the Strong wolfe line search conditions [12]. The optimization steps are summarized in Algorithm 4 (Appendix B).

3) Putting It All

Algorithm 1 summarizes the overall workflow. First, we find a rank-1 matrix that is maximally correlated with the partially-observed RTT matrix $\mathbf{D}$ (In *line two*). Next, we set the current approximation based on the rank-1 matrix $\mathbf{X}_1$ (in *line three*). Then, we perform layerwise training (from *lines four to ten*), by finding the rank-1 matrix and the sign matrix, and forward the residual to the next loop cycle.

D. Analysis

1) Convergence

SMF generalizes the truncated SVD that recursively finds an approximately optimal rank-1 matrix to the residual. The SVD sums up the rank-1 matrix to form the low-rank prediction, while SMF flexibly combines the rank-1 matrix to be aware of the skewness of node pairs. As a result, the truncated SVD is a special case for SMF.

SMF and ResNet [26] both optimize the residuals in a layerwise approach. ResNet [26] forces each layer to learn the residual feature between the output of this layer and that of the last layer, which efficiently addresses the gradient vanishing problems for deep neural networks. While SMF not only forces each layer to approximate the residual, but also regularizes the structure of the layer to be aware of skewed latent factors. Accordingly, SMF adapts well to the matrix-factorization context.

Recall that the rank-1 matrix factorization and the sign matrix estimation are two conditionally independent optimization problems. As a result, the problem of proving SMF's convergence is transformed to prove the convergence of the rank-1 matrix factorization and that of the sign matrix in each layer.

(i) **Rank-1 Matrix**

A point x is a stationary point iff its gradient $\nabla J(x)$ =0. Geometrically, x is a local minimum iff x is a stationary point and there exists a neighborhood area $NA(x)$ of the vector x such that $J(z) \geq J(x)$ for any $z \in NA(x)$ [32]. We restate the strict saddle property for twice differentiable functions such as the rank-1 matrix factorization.

Definition 1 ([20], [21])**.** *A twice differentiable function J is (ψ, Ω, ω)-strict saddle, if for any point x at least one condition holds: (i) $\|\nabla J(x)\| \geq \psi$; (ii) $\lambda_{min}\left(\nabla^2 J(x)\right) \leq -\Omega$; (iii) x is ω-close to the set of local minima.*

The strict saddle definition implies that for any point x, either it has a large gradient, or it has a negative directional curvature, or it is close to a local minima. Lemma 2 shows that, the loss function J_k defined in Eq (6) satisfies the strict saddle condition.

Lemma 2. *Let σ_1^*, σ_r^* denote the largest and the r-th singular values of the residual $\mathbf{E}_k$. The loss function J_k is $\left(\epsilon, \Omega(\sigma_r^*), O\left(\frac{\epsilon}{\sigma_r^*}\right)\right)$-strict saddle.*

The proof of Lemma 2 directly derives from Theorem 4 in [21]. Accordingly, the rank-1 matrix factorization satisfies that: All local minima are also globally optimal; Any saddle point has at least one strictly negative eigenvalue in its Hessian matrix, thus local search methods efficiently find points towards the local minima [20], [21].

Further, the optimization process of the rank-1 matrix with a random initialization converges to or a local minimizer almost surely, as described in Lemma C and proved in Appendix C.

Lemma 3. *If $\lambda \geq 0$, $\eta < \frac{1}{L_{smooth}}$, where L_{smooth} denotes the smoothness of function J, i.e. $\|\nabla J(x_1) - \nabla J(x_2)\| \leq L\|x_1 - x_2\|$, Algorithm 3 with a random initialization converges to a local minimizer almost surely.*

(ii) **Sign Matrix**

The nonlinear conjugate-gradient optimization converges to zero gradients, as proved in Theorem 3.5 by Liu et al. [12] under the following conditions:

Lemma 4. *The gradient of the nonlinear conjugate-gradient method converges to zero, i.e., $\lim_{s\to\infty} \nabla g_k(s) = 0$ given the conditions:*

- *$g(x)$ is bounded below on the level set Ł $= \{x|g(x) \leq g(x_1)\}$, where x_1 denotes the starting point. And in some neighborhood of Ł, g is continuously differentiable, and the gradient is Lipschitz continuous, i.e., for a constant $L > 0$, such that $\|g(x) - g(y)\| \leq L\|x - y\|$.*
- *The level set Ł $= \{x|g(x) \leq g(x_1)\}$ is bounded.*
- *Polak-Ribière scalar $\gamma_s \geq 0$.*
- *Strong wolfe line search conditions: A positive step-length η_2 computed by a line search satisfies that $g(x_s + \eta_2 \Lambda x_s) \leq g(x_s) + \rho\eta_2 \Delta {x_s}^T \Lambda x_s$ and $\Delta(x_s + \eta_2 \Lambda x_s)^T \Lambda x_s \geq \sigma \Delta {x_s}^T \Lambda x_s$, for $0 < \rho < \sigma < 1$.*
- *Descendent condition: $\Delta {x_s}^T \Lambda x_s < 0$.*
- *Property (*): There exists constants $b > 0$ and $\lambda > 0$ for all k, if $|\gamma_s| \leq b$, and $\left\|x_(s) - x(s-1)\right\| \leq \lambda$, then $|\gamma_s| \leq \frac{1}{2b}$.*

(iii) **Discussions**

Further, significant efforts [20], [22], [29], [21], [31], [23], [16], [30], [46] have proved that first-order local search optimization algorithms have nice convergence guarantee for general non-convex optimization problems: No spurious local minimum arise in the optimization landscape; further, simple local search methods escape saddle points efficiently and find the exact low-rank matrix from arbitrary starting points with high probability in polynomial time.

2) Robustness

In practice, some entries of the sign matrix may be incorrectly predicted. Specifically, there exists a small probability that "+1" is flipped to "-1", "-1" to "+1", and "0" to "+1" or "-1". Note that mapping "+1" or "-1" to "0" does not degrade the current approximation residual. We next bound the failure probability that the sign of any server pair is always incorrect in Theorem 1.

Theorem 1. *Let r be the approximation rank and ϵ the expected fraction of incorrectly predicted signs. For any server pair, the expected number of correctly predicted signs amounts to $r(1-\epsilon)$, the variance is $\epsilon r(1-\epsilon)$, and for $t > 0$, the number of correct estimations is not within $t\sqrt{r(1-\epsilon)}$ from its expectation is at most $\frac{\epsilon}{t^2}$.*

The proof is put in the appendix. For example, if $\epsilon = 0.2$ and r = 8, the expected number of correct predictions amounts to $\lfloor 8 \times 0.8 \rfloor = 6$, with the variance $0.2 \cdot 8(1 - 0.2) = 1.28$. Further, note that when a sign is mapped to zero from "+1" or "-1", then the current approximation will be skipped and the residual will not change accordingly.

3) Time Complexity

The time complexity of SMF amounts to the sum of time for each layer, which is spent on finding the rank-1 matrix and the sign matrix.

(i) **Rank-1 matrix**: The time complexity of the SGD algorithm is proportional to the gradient computation. For each node i, one gradient calculation takes $O(n_p)$ time, where n_p denotes the set of probing targets. N nodes take $O(Nn_p)$ time to compute the gradient for one round. Algorithm 3 needs T rounds, requiring $O(TNn_p)$ time.

(ii) **Sign Matrix**: The time complexity of the conjugate-gradient method is linearly proportional to the calculation

of the conjugate gradient [18]. Let r_s denote the dimension of the coordinate, it takes $O(r_s n_p)$ time to compute the partial gradient and $O(n_p r_s^2)$ time to derive the conjugate gradient direction for each node i. Consequently, N nodes take $O(N n_p r_s^2)$. Algorithm 4 needs T rounds, requiring $O(TN n_p r_s^2)$.

In summary, training a rank-r SMF model needs $O(rTN n_p(1 + r_s^2))$ time.

E. Parameter Choices

As the optimization problem is non-convex, it is challenging to choose optimal parameters for the SMF method. For an unknown dataset, we propose an offline approach to decide the number of rank-1 components, the dimension of the sign matrix, and the regularization constant. We fix all but one parameters, and incrementally adjust the parameter until the average relative error does not decrease significantly.

From our empirical experiments, a wide range of parameters achieve similar degrees of the estimation accuracy. As the majority of a RTT matrix is typically captured by several singular values from Figure 4. Therefore, we could select a relatively small numbers of probing targets and rank-1 matrices, as well as the dimension of the sign-matrix approximation.

VI. Evaluation

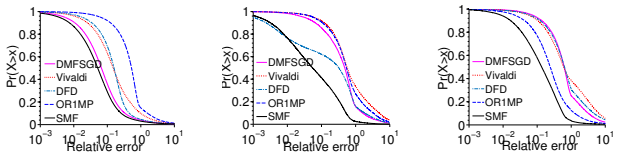
Having presented the optimization algorithms, we next systematically evaluate the performance of our method with some of the state-of-the-art methods on real-world data sets.

A. Experimental Setup

We built a trace-driven controller-agent structured simulator. We reuse the data sets introduced in Section III-C1 for the simulator. The simulator parses the trace for continuous network latency monitoring. During a slice, each agent actively randomly samples a small set of nodes from the data set as probing targets, then submits network latency values to these targets to the controller. The controller collects probed samples from all agents, then initializes parameters for each node in the continuous vector space, and adjusts parameters based on optimization rules until reaching the local minimum. Afterward, the controller predicts network latency values for missing node pairs.

Performance metric: We quantify the performance of unobserved entries using *relative error* that is widely used for network latency prediction studies. It is defined as the ratio between the relative error and the ground-truth value $\frac{|\mathbf{D}_{ij} - \hat{\mathbf{D}}_{ij}|}{\mathbf{D}_{ij}}$ for each unobserved node pair (i, j). For each setting, the simulation runs in ten times. The results reported show the average relative error.

Default parameters: We choose the default parameters for SMF to balance the diminishing returns of the expected relative error and the computational cost, based on the sensitivity analysis in Section VI-D. Specifically, we set the approximation rank r to 8, the number of probing targets n_p to 32, the bias MMMF coordinate dimension r_s to 16, and the regularized parameter λ to one.



(a) PlanetLab (b) Seattle (c) RIPE

Fig. 9. The CCDFs of relative errors for different methods. We set the number of probes per node to 32. The x-axis is in log-scale.

Our parameter choices yield sparse matrices. For example, for a 99-by-99 matrix, setting the number of probing targets to 32 implies that 32% of matrix entries are observed, while for a 490-by-490 matrix, it implies that only 6.5% of matrix entries are observed.

We run experiments on a MacBook-Pro Intel Core i7 with Quad-core and 16 GB memory.

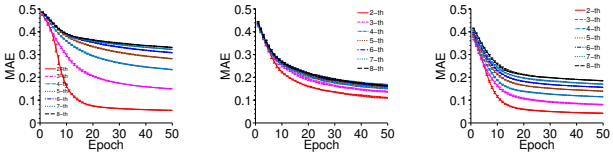
B. Comparison Results

First, we compare the relative error of our method with some of the state-of-the-art methods. As we mentioned in the related work, there are many methods proposed in the literature for network latency prediction or matrix completion. We choose four baseline algorithms covering the recent well-known works: (i) Vivaldi [11]: predicts RTTs in an Euclidean coordinate system and trains the coordinates via a scalable spring-force field simulation. Further, Vivaldi incorporates a height constant to model the access-link portion from end hosts to the edge. (ii) DMFSGD [33]: predicts RTTs in a low-rank matrix factorization model and trains the factorized matrices based on the Stochastic Gradient Descendent (SGD) method. (iii) Distance feature decomposition (denoted as DFD in the plot) [34]: predicts RTTs via the product of a low-rank matrix and a complete weight matrix that are trained by the Vivaldi method [11] and the Penalty Decomposition (PD) method [55], respectively. (iv) OR1MP [47]: recovers an incomplete matrix by incrementally generating a rank-one basis matrix by the SVD method and linearly combining this matrix to the current approximation by computing a weight vector in a closed form.

For Vivaldi, DMFSGD and OR1MP, we directly downloaded authors' implementations, while for DFD, we implemented the algorithm based on [34].

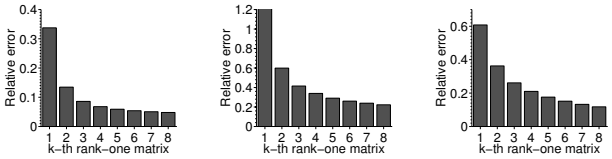
For fair comparison, all methods use the same number of probes, and set the same coordinate dimension and the identical set of probing targets. We set the regularized parameters based on the recommended configuration of each method.

We plot the distributions of the relative errors. We set the number of probing targets to 32 and the coordinate dimension to eight. We compute the CCDF of relative errors for all methods. From Figure 9, SMF consistently outperforms the other methods in three data sets, while DFD, DMFSGD and Vivaldi are less stable than SMF. OR1MP is less accurate than SMF, since OR1MP combines a set of rank-one matrices with a weight vector, which is insufficient to account for skewed latent factors.



(a) PlanetLab (b) Seattle (c) Seattle

Fig. 10. The convergence of the MAE values at each round. We plot the average MAE and the standard deviations.



(a) PlanetLab (b) Seattle (c) Seattle

Fig. 11. The average relative error as a function of the number of combined rank-one matrices.

C. Convergence Analysis

Having shown the effectiveness of our method, we next evaluate the convergence of SMF's performance.

1) Sign-matrix Prediction

First, we evaluate the convergence of the sign-matrix prediction process, as it determines the convergence of the selective combination procedure. We quantify the difference between the estimated sign matrix and the ground-truth result based on the normalized mean absolute error (MAE): $\frac{\sum_{(i,j)\in\text{missing entries}}|\widehat{Y}_{ij}-Y_{ij}|}{\sum_{(i,j)\in\text{missing entries}}Y_{ij}}$, where Y_{ij} represents the ground-truth sign from i to j, and $\widehat{Y}_{ij}$ denotes the estimated sign.

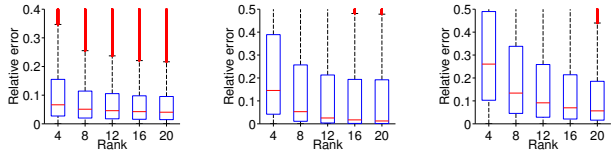
In Figure 10, we plot the average MAE values of the coordinates at each round. We see that the MAE values monotonically decrease towards the local minimum with increasing numbers of rounds. Further, the MAE values generally increase as we consider more rank-1 matrix components, while most incorrect estimations are either mapping "+1" or "-1" to "0", since most incorrect signs are adjacent to the correct values. Accordingly, the estimation accuracy is not affected by the sign "0", as it implies that we skip the current rank-1 component entry, therefore,

2) Approximation Residual

Having shown the convergence of the sign matrix prediction, we next evaluate the relative error of approximation residuals as we combine more rank-one matrices to the approximation. In Figure 11, we see that the RTT matrix can be compactly captured with two to three rank-one matrices. The average relative error decreases significantly as we combine the second and the third rank-one matrices, while combining more matrices marginally decreases the average relative error.

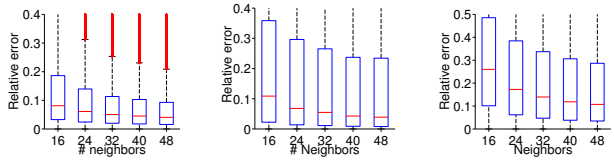
D. Sensitivity Results

Having presented the convergence of the skewness-aware matrix factorization, we next test the performance sensitivity of our method. We fix all but one parameters to default values, and vary the parameter configuration. We report the average relative errors and the standard deviation.



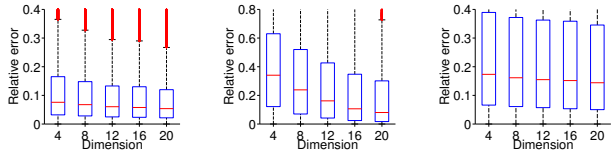
(a) PlanetLab (b) Seattle (c) RIPE

Fig. 12. Sensitivity of the approximation rank as we increase the rank from 4 to 20.



(a) PlanetLab (b) Seattle (c) RIPE

Fig. 13. Sensitivity of the number of probing targets as we increase the number from 16 to 48.



(a) PlanetLab (b) Seattle (c) RIPE

Fig. 14. Sensitivity of the bias-MMMF dimension as we increase the coordinate dimension from 4 to 20.

1) Approximation Rank

First, we compute the distributions of the relative errors as we increase the approximation rank r from four to twenty. In Figure 12, we see that eight to twelve rank-1 components are enough to obtain an accurate estimation. Further, the median, the 15-th and the 75-th percentiles of relative errors are steady on the Planetlab dataset, since the PlanetLab dataset is approximately low-rank. Comparatively, the relative errors on the Seattle and RIPE dataset decrease significantly when we increase the rank from 4 to 8, as these data sets have longer tails of singular values than the PlanetLab dataset.

2) Number of Measurements for each Node

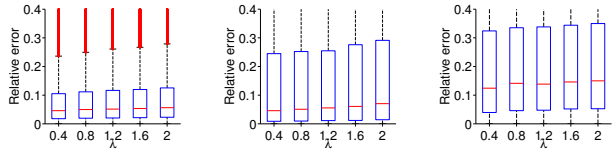
Next, we evaluate the choice of the number of targets. We compute the relative errors for each setting. From Figure 13, we see that the average relative error decreases progressively, and 32 is enough to obtain relatively accurate results.

3) Biased-MMMF Dimension

Next, we vary the choice of the coordinate dimension for predicting the sign matrix and plot the relative-error distribution. In Figure 14, we see that setting the dimension to 12 to 16 is enough to obtain a relatively high degree of estimation. The median relative error decreases marginally on the PlanetLab, but decrease progressively on the Seattle and RIPE datasets, which also holds for the 15-th and 75-th percentiles of the relative errors.

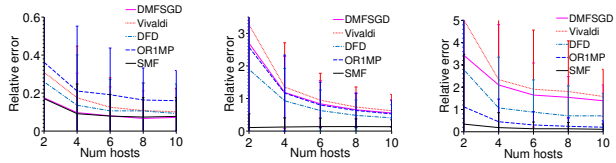
4) Regularization λ

Next, we study the performance sensitivity with respect to the regularized parameter λ that controls the extent of the regularization. We vary the regularized parameter from 0.4 to



(a) PlanetLab (b) Seattle (c) RIPE

Fig. 15. Sensitivity of the regularization parameter λ as we increase the regularized parameter λ from 0.4 to 2.



(a) PlanetLab (b) Seattle. (c) RIPE.

Fig. 16. The average relative error and the standard deviation of the found relay as a function of the number of targeting detouring hosts. For a set of nodes $(T_1, \ldots, T_L)$ of size ϖ that need detour routing, we define the optimal relay as the one that minimizes the average network latency: $\frac{1}{\varpi}\sum_{j=1}^{\varpi}\left(d_{T_jR_O} + d_{R_OT_j}\right)$ We quantify the detouring performance for each node pair by comparing the optimal relay selected from the ground-truth network latency matrix and that from the estimated all-pair network latency matrix: $\frac{\left|\sum_{j=1}^{\varpi}\left(d_{T_jR_f}+d_{R_fT_j}\right)-\sum_{j=1}^{\varpi}\left(d_{T_jR_O}+d_{R_OT_j}\right)\right|}{\sum_{j=1}^{\varpi}\left(d_{T_jR_O}+d_{R_OT_j}\right)}$, where R_f and R_O denote the estimated relay and the optimal relay respectively.

2 and compute the relative error distribution for each setting. From Figure 15, the relative error keeps steady as we change the regularized parameter, thus the prediction is less sensitive to the regularization than other parameters.

Summary: From the sensitivity analysis, a wide range of parameters yield similar accuracy. Therefore, we should choose modest parameters in order to trade off well between the accuracy and the computation complexity.

E. Use Case: Predicting Latency-optimal Detouring Servers

Having evaluated the performance of SMF, we next evaluate the performance gains for selecting detouring routing nodes that act as proxies to forward packets for end hosts, such as Internet telephony [28]. Given a group of clients that need detour routing via a relay, we choose a server from the overlay with the minimal average RTT value towards this group of clients.

Figure 16 plots the variations of the average relative errors of the found relay as we vary the number of relayed hosts. We can see that on the PlanetLab dataset, our method and DMFSGD obtain the most accurate relays, since the PlanetLab dataset can be well approximated via a low-rank model, while on the Seattle and RIPE datasets, our method significantly outperforms the other methods.

VII. Conclusion

Monitoring pairwise RTT status in a service mesh is vital for network troubleshooting and performance management. However, an all-pair probing approach faces severe scaling limitations. Existing matrix factorization based methods overcome the scaling limitations, but could not truthfully capture the skewed distributions of latent factors in the RTT distributions.

We address this challenge by proposing a skewness-aware matrix factorization method named SMF to adaptively select latent factors for different node pairs. We extend the orthogonal matching pursuit algorithm to the matrix factorization context, by incrementally combining a new rank-one matrix weighted by their correlations with the current approximation residual. We propose an automatic combination method based on the logical connection between the selection policies and the recommendation system. Extensive experiments over real-world data sets show that SMF significantly improves the relative error by a factor of 0.2x to 10x, converges fast and stably, and captures fine-grained local and global network latency structures with two to three rank-one matrices.

Although this paper focuses on a software-defined measurement architecture, we can decompose the representation model to decentralized components. First, a rank-1 matrix can be decomposed to separable models with respect to each node. Second, the sign matrix model can be decomposed to separate coordinates, as already proven in previous studies in [17], [18]. After we decompose the representation model, we may optimize tuples a decentralized procedure, which is left as future work.

References

[1] Linux PTP Project. http://nwtime.org/projects/linuxptp/, 2016.
[2] NTP. http://www.ntp.org, 2016.
[3] Conduit service mesh. https://conduit.io, 2018.
[4] Istio service mesh. https://istio.io, 2018.
[5] Amazon. Amazon Elastic Compute Cloud. aws.amazon.com/ec2, October 2010.
[6] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. of SOSP*, pages 31–41, 2001.
[7] L. A. Barroso, M. Marty, D. Patterson, and P. Ranganathan. Attack of the Killer Microseconds. *Commun. ACM*, 60(4):48–54, 2017.
[8] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Commun. ACM*, 55(6):111–119, 2012.
[9] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson. Seattle: a Platform for Educational Cloud Computing. In *Proc. of SIGCSE*, pages 111–115, 2009.
[10] Y. Chen, L. Qiu, Y. Zhang, G. Xue, and Z. Hu. Robust Network Compressive Sensing. In *Proc. of MobiCom*, pages 545–556, 2014.
[11] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: a Decentralized Network Coordinate System. In *Proc. of ACM SIGCOMM*, pages 15–26, 2004.
[12] Y. Dai, J. Han, G. Liu, D. Sun, H. Yin, and Y. Yuan. Convergence properties of nonlinear conjugate gradient methods. *SIAM Journal on Optimization*, 10(2):345–358, 2000.
[13] J. Dean and L. A. Barroso. The Tail at Scale. *Commun. ACM*, 56(2):74–80, Feb. 2013.
[14] D. DeCoste. Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proc. of ICML*, pages 249–256, 2006.
[15] C. H. Q. Ding, T. Li, and M. I. Jordan. Convex and Semi-Nonnegative Matrix Factorizations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(1):45–55, 2010.
[16] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai. Gradient descent finds global minima of deep neural networks. *CoRR*, abs/1811.03804, 2018.
[17] Y. Fu and Y. Wang. iRank: Supporting Proximity Ranking for Peer-to-Peer Applications. In *Proc. of IEEE ICPADS '09*, pages 836–841.
[18] Y. Fu, Y. Wang, and E. Biersack. A General Scalable and Accurate Decentralized Level Monitoring Method for Large-scale Dynamic Service Provision in Hybrid Clouds. *Future Generation Comp. Syst.*, 29(5):1235–1253, 2013.
[19] Y. Fu and X. Xu. Self-Stabilized Distributed Network Distance Prediction. *IEEE/ACM Trans, Netw.*, 25(1):451–464, Feb 2017.

[20] R. Ge, F. Huang, C. Jin, and Y. Yuan. Escaping from saddle points - online stochastic gradient for tensor decomposition. In *Proc. of COLT*, pages 797–842, 2015.

[21] R. Ge, C. Jin, and Y. Zheng. No spurious local minima in nonconvex low rank problems: A unified geometric analysis. In *Proc. of ICML*, pages 1233–1242, 2017.

[22] R. Ge, J. D. Lee, and T. Ma. Matrix completion has no spurious local minimum. In *Proc. of NIPS*, pages 2973–2981, 2016.

[23] R. Ge and T. Ma. On the optimization landscape of tensor decompositions. In *Proc. of NIPS*, pages 3656–3666, 2017.

[24] G. H. Golub and C. F. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[25] C. Guo, L. Yuan, D. Xiang, Y. Dang, R. Huang, D. Maltz, Z. Liu, V. Wang, B. Pang, H. Chen, Z.-W. Lin, and V. Kurien. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. In *Proc. of SIGCOMM*, pages 139–152, 2015.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, pages 770–778, 2016.

[27] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[28] J. Jiang, R. Das, G. Ananthanarayanan, P. A. Chou, V. N. Padmanabhan, V. Sekar, E. Dominique, M. Goliszewski, D. Kukoleca, R. Vafin, and H. Zhang. VIA: Improving Internet Telephony Call Quality Using Predictive Relay Selection. In *Proc. of SIGCOMM*, pages 286–299, 2016.

[29] C. Jin, S. M. Kakade, and P. Netrapalli. Provable efficient online matrix completion via non-convex stochastic gradient descent. In *Proc. of NIPS*, pages 4520–4528, 2016.

[30] C. Jin, P. Netrapalli, R. Ge, S. M. Kakade, and M. I. Jordan. Stochastic gradient descent escapes saddle points efficiently. *CoRR*, abs/1902.04811, 2019.

[31] J. D. Lee, I. Panageas, G. Piliouras, M. Simchowitz, M. I. Jordan, and B. Recht. First-order methods almost always avoid saddle points. *CoRR*, abs/1710.07406, 2017.

[32] X. Li, J. D. Haupt, J. Lu, Z. Wang, R. Arora, H. Liu, and T. Zhao. Symmetry. saddle points, and global optimization landscape of nonconvex matrix factorization. In *Proc. of ITA*, pages 1–9, 2018.

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

[34] B. Liu, D. Niu, Z. Li, and H. V. Zhao. Network latency prediction for personal devices: Distance-feature decomposition from 3d sampling. In *Proc. of INFOCOM*, pages 307–315, 2015.

[35] C. Lumezanu, R. Baden, D. Levin, N. Spring, and B. Bhattacharjee. Symbiotic Relationships in Internet Routing Overlays. In *Proc. of NSDI*, NSDI'09, pages 467–480, 2009.

[36] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. E. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An Information Plane for Distributed Services. In *Proc. of USENIX OSDI*, pages 367–380, 2006.

[37] 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, 2006.

[38] T. S. E. Ng and H. Zhang. Predicting Internet Network Distance with Coordinates-Based Approaches. In *Proc. of INFOCOM*, pages 170–179, 2002.

[39] A. Nikravesh, H. Yao, S. Xu, D. R. Choffnes, and Z. M. Mao. Mobilyzer: An Open Platform for Controllable Mobile Network Measurements. In *Proc. of MobiSys*, pages 389–404, 2015.

[40] E. Nygren, R. K. Sitaraman, and J. Sun. The Akamai Network: a Platform for High-performance Internet Applications. *SIGOPS Oper. Syst. Rev.*, 44:2–19, August 2010.

[41] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *Proc. of ACSSC*, pages 40–44 vol.1, Nov 1993.

[42] J. D. M. Rennie and N. Srebro. Fast Maximum Margin Matrix Factorization for Collaborative Prediction. In *Proc. of ICML*, pages 713–719, 2005.

[43] J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42(1):59–66, 1988.

[44] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: A Case for Informed Internet Routing and Transport. *IEEE Micro*, 19:50–59, 1999.

[45] A.-J. Su, D. R. Choffnes, A. Kuzmanovic, and F. E. Bustamante. Drafting behind Akamai (travelocity-based detouring). In *Proc. of SIGCOMM*, 2006.

[46] R. Sun and Z. Luo. Guaranteed matrix completion via non-convex factorization. *IEEE Trans. Information Theory*, 62(11):6535–6579, 2016.

[47] Z. Wang, M. Lai, Z. Lu, W. Fan, H. Davulcu, and J. Ye. Orthogonal Rank-One Matrix Pursuit for Low Rank Matrix Completion. *SIAM J. Scientific Computing*, 37(1), 2015.

[48] M. Weimer, A. Karatzoglou, and A. Smola. Improving Maximum Margin Matrix Factorization. *Mach. Learn.*, 72:263–276, September 2008.

[49] F. M. F. Wong, C. Joe-Wong, S. Ha, Z. Liu, and M. Chiang. Improving User QoE for Residential Broadband: Adaptive Traffic Management at the Network Edge. In *Proc. of IWQoS*, pages 105–114, 2015.

[50] K. Xie, C. Peng, X. Wang, G. Xie, J. Wen, J. Cao, D. Zhang, and Z. Qin. Accurate Recovery of Internet Traffic Data Under Variable Rate Measurements. *IEEE/ACM Trans. Netw.*, 26(3):1137–1150, 2018.

[51] K. Xie, L. Wang, X. Wang, G. Xie, J. Wen, G. Zhang, J. Cao, and D. Zhang. Accurate Recovery of Internet Traffic Data: A Sequential Tensor Completion Approach. *IEEE/ACM Trans. Netw.*, 26(2):793–806, 2018.

[52] K. Xie, L. Wang, X. Wang, G. Xie, G. Zhang, D. Xie, and J. Wen. Sequential and Adaptive Sampling for Matrix Completion in Network Monitoring Systems. In *Proc. of INFOCOM*, pages 2443–2451, 2015.

[53] R. Xu and D. Wunsch. Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.

[54] C. Yu, C. Lumezanu, A. B. Sharma, Q. Xu, G. Jiang, and H. V. Madhyastha. Software-Defined Latency Monitoring in Data Center Networks. In *Proc. of PAM*, pages 360–372, 2015.

[55] Y. Zhang and Z. Lu. Penalty Decomposition Methods for Rank Minimization. In *Proc. of NIPS*, pages 46–54, 2011.

[56] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu. Spatio-temporal Compressive Sensing and Internet Traffic Matrices. In *Proc. of SIGCOMM*, pages 267–278, 2009.

[57] J. Zhu, P. He, Z. Zheng, and M. R. Lyu. Online QoS Prediction for Runtime Service Adaptation via Adaptive Matrix Factorization. *IEEE Trans. Parallel Distrib. Syst.*, 28(10):2911–2924, 2017.

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

[59] R. Zhu, D. Niu, and Z. Li. Robust Web Service Recommendation via Quantile Matrix Factorization. In *Proc. of INFOCOM*, pages 1–9, 2017.

[60] Z. Zhu, Q. Li, G. Tang, and M. B. Wakin. Global optimality in low-rank matrix optimization. *IEEE Trans. Signal Processing*, 66(13):3614–3628, 2018.

APPENDIX

Note: The material in this Appendix is included for reviewer information only. It will be included in the supplementary material in the final version.

A. Proof of *Lemma 1*

Lemma 1: Assume that we obtain the perfect sign matrix, for all $k \geq 1$, $\|\mathbf{E}_{k+1}\| \leq \|\mathbf{E}_k\|$

Proof. For all $k \geq 1$

$$\begin{aligned} \|E_{k+1}\| &= \|E_k + \beta_{k+1}. * X_{k+1}\| \\ &= \left\|\min_{\beta_{k+1}, X_{k+1}} (E_k, E_k + \beta_{k+1}. * X_{k+1})\right\| \leq \|E_k\| \end{aligned}$$

holds, which completes the proof. □

Therefore, it is vital to control the estimation accuracy of the sign matrix.

B. Algorithm Pseudocodes

(i) SignRule($\mathbf{E}_k\,[i,j]$, $\mathbf{X}_{ij}$), Algorithm 2:
(ii) RankOne($|E_k|$), Algorithm 3:
(iii) SignMatrix($\beta_{\mathbf{k}}$), Algorithm 4:

Algorithm 2: The rule of calculating the sign for an observed node pair (i, j).

```
SignRule(E_k [i,j], X_ij)
  input : E_k [i,j]: Current approximation residual, X_ij:
          rank-one esimtation for pair (i,j)
  output: Sign β_k [i,j].
if |E_k [i,j] + X_ij]| < |E_k [i,j]| then
    Set β_k [i,j] = "+1";
else if |E_k [i,j] − X_ij]| < |E_k [i,j]| then
    Set β_k [i,j] = "-1" ;
else
    Set β_k [i,j] = "0";
return β_k [i,j] ;
```

Algorithm 3: SGD based rank-one matrix optimization.

```
RankOne(|E_k|)
  input : Partially observed residual |E_k|, learning rate η_1,
          optimization rounds T
  output: F_k, G_k
for t = 1,2,··· ,T do
    for i ∈ 1,2,··· ,N do
        Compute the partial gradients of ∂J/∂F_k(i) and ∂J/∂G_k(i) of
        node i as:
            ∂J/∂F_k(i) = λF_k(i)(F_k^T F_k − G_k^T G_k) −
            Σ_{j∈S_i} (|E_ij| − F_k(i)G_k(j)) G_k(j)
            ∂J/∂G_k(i) = −λG_k(i)(F_k^T F_k − G_k^T G_k) −
            Σ_{j∈S_i} (|E_ji| − F_k(j)G_k(i)) F_k(j)        (9)
        Update the coordinate as:
            F_k(i) = F_k(i) − η_1 ∂J/∂F_k(i)
            G_k(i) = G_k(i) − η_1 ∂J/∂G_k(i)                (10)
return F_k, G_k;
```

C. Convergence Analysis of the Rank-1 Matrix

For a twice differentiable function such as the rank-1 matrix factorization, a vector x is a local minimum iff it is a stationary point and satisfies that the Hessian matrix $\nabla^2 J(x)$ is positive definite. A vector x is a strict saddle point iff it is a stationary point and satisfies that the minimum eigenvalue of the Hessian matrix $\nabla^2 J(x)$ is less than zero, i.e., $\lambda_{min}\left(\nabla^2 J(x)\right) < 0$ [21]. Further, if $\lambda_{min}\left(\nabla^2 J(x)\right) = 0$, then x is either a local minima or a degenerate saddle point.

Lemma: If $\lambda \geq 0$, $\eta < \frac{1}{L_{smooth}}$, where L_{smooth} denotes the smoothness of function J, i.e. $\|\nabla J(x_1) - \nabla J(x_2)\| \leq L\|x_1 - x_2\|$, Algorithm 3 with a random initialization converges to a local minimizer almost surely.

Proof. Lemma 5 states that the gradient descendent algorithm efficiently finds points towards the local minima, given that the step size η is less than $\frac{1}{L_{smooth}}$, where L_{smooth} denotes the smoothness of the objective function J.

Lemma 5 ([31]). *For any twice continuous differentiable function that satisfies the strict saddle property, if $\eta < \frac{1}{L_{smooth}}$, where L_{smooth} denotes the smoothness of function, i.e. $\|\nabla J(x_1) - \nabla J(x_2)\| \leq L\|x_1 - x_2\|$, the gradient descendent algorithm with a random initialization converges to a local minimizer or $-\infty$ almost surely.*

Algorithm 4: Sign matrix optimization.

```
SignMatrix(β)
  input : Partially observed sign matrix β, learning rate η_2.
  output: u⃗, v⃗, θ⃗ and b
for s = 1,2,··· ,T do
    for i ∈ 1,2,··· ,N do
        Let the gradient of x_i be
            ∇_x J(x_i) = [∂J/∂u_i; ∂J/∂v_i; ∂J/∂θ_i; ∂J/∂b_i]        (11)
        where ∂J/∂u, ∂J/∂v, ∂J/∂θ, ∂J/∂b are the partial derivatives with
        respect to u⃗_i, v⃗_i, θ⃗_i and b_i:
            ∂J/∂u_iz = λu_iz − Σ_{j∈S_i} Σ_{c=1}^{2} T_ij^c[β_ij] · h'(T_ij^c[β_ij] · (θ_ic − θ̂_ij)) v_jz   (12)
            ∂J/∂v_iz = λv_iz − Σ_{j∈S_i} Σ_{c=1}^{2} T_ji^r[β_ji] · h'(T_ji^c[β_ji] · (θ_jc − θ̂_ji)) u_jz   (13)
            ∂J/∂θ_ic = Σ_{j∈S_i} T_ij^c[β_ij] · h'(T_ij^c[β_ij] · (θ_ic − θ̂_ij))                            (14)
            ∂J/∂b_i  = λb_i − Σ_{j∈S_i} Σ_{c=1}^{2} T_ij^c[β_ij] · h'(T_ij^c[β_ij] · (θ_ic − θ̂_ij))         (15)
        Calculate the nonnegative Polak-Ribière scalar γ_i [12]
        by ensuring that the updated Λx_i(s) and Λx_i(s−1)
        must be conjugate:
            γ_i = max(0, Δx_i(s)^T (Δx_i(s) − Δx_i(s−1)) / (Δx_i(s−1)^T Δx_i(s−1)))   (16)
        We set Δx_i(0) = −∇_x J(x_i(0)), and
        Λx_i(0) = Δx_i(0);
        Calculate the conjugate gradient Λx_i(s) as
            Λx_i(s) = Δx_i(s−1) + γ_i Λx_i(s−1)                      (17)
        where Δx_i(s−1) denotes the negative gradient ∇_x J,
        i.e.,
            Δx_i(s) = −∇_x J(x_i(s−1))                               (18)
        and Λx_i(s−1) denotes the conjugate direction in the
        (s−1)-th round;
        Update the vector x_i(s) = x_i(s) + η_2 Λx_i(s);
        Update the coordinate [u⃗_i; v⃗_i; θ⃗_i; b_i] = x_i(s);
return u⃗, v⃗, θ⃗ and b;
```

Further, since $\lambda \geq 0$, the objective function J is non-negative, thus the objective function J converges to a local minimizer almost surely.

□

D. Proof of ***Theorem 1***

Theorem 1: Let r be the approximation rank and ϵ the expected fraction of incorrectly predicted signs. For any server

pair, the expected number of correctly predicted signs amounts to $r(1-\epsilon)$, the variance is $\epsilon r(1-\epsilon)$, and for $t > 0$, the number of correct estimations is not within $t\sqrt{r(1-\epsilon)}$ from its expectation is at most $\frac{\epsilon}{t^2}$.

Proof. Assume that the sign matrix can be correctly predicted for at least $(1-\epsilon)$ (for $\epsilon \in [0,1]$) fractions of entries in each iteration, and that the incorrect signs are spread uniformly at random over the matrix. Then the sign of any server pair is incorrectly predicted with a probability ϵ. For any matrix entry (i,j) and the index k, $\|\mathbf{E}_{k+1}[i,j]\| \leq \|\mathbf{E}_k[i,j]\|$ holds with a probability $(1-\epsilon)$.

Since the sign matrices of different iterations are independent with each other, the always-failure probability is at most ϵ^r, which decreases exponentially to zero with increasing rank r. For example, if $\epsilon = 0.1$ and r = 8, the always-failure probability amounts to $\epsilon^r = 0.1^8 \approx 0$.

Generally, let $Z_k[i,j]$ be an indicator function of whether the estimated sign is correct for the k-th iteration:

$$Z_k[i,j] = \begin{cases} 1 & \hat{\beta}[i,j] == \beta[i,j] \\ 0 & \hat{\beta}[i,j] \neq \beta[i,j] \end{cases}$$

We can see that the probability $Pr(Z_k == 1) = 1-\epsilon$. Let $Z[i,j]$ be the random variable of the number of correct predictions for the server pair (i,j). Then we have

$$Z[i,j] = \sum_{k=1}^{r} Z_k[i,j] \tag{19}$$

The expected number of correct predictions for the server pair (i,j) can be calculated as:

$$\begin{array}{l} E[Z[i,j]] = E\left[\sum_{k=1}^{r} Z_k[i,j]\right] \\ = \sum_{k=1}^{r} E[Z_k[i,j]] \\ = \sum_{k=1}^{r} (1 \times (1-\epsilon)) \\ = r(1-\epsilon) \end{array} \tag{20}$$

Further, the variance of the number of correct predictions can be written as:

$$\begin{array}{l} Var(Z[i,j]) = Var\left[\sum_{k=1}^{r} Z_k[i,j]\right] \\ = \sum_{k=1}^{r} Var[Z_k[i,j]] \\ = \sum_{k=1}^{r} \left(E\left[(Z_k[i,j])^2\right] - (E[Z_k[i,j]])^2\right) \\ = \sum_{k=1}^{r} \left(\left(1^2 \times (1-\epsilon)\right) - (1-\epsilon)^2\right) \\ = \epsilon r(1-\epsilon) \end{array} \tag{21}$$

By the standard Chebyshev's inequality, for $\delta > 0$, we have

$$P(|Z[i,j] - r(1-\epsilon)| > \delta) \leq \frac{\epsilon r(1-\epsilon)}{\delta^2}$$

By taking $\delta = t\sqrt{r(1-\epsilon)}$, where $t > 0$, we have

$$P\left(|Z[i,j] - r(1-\epsilon)| < t\sqrt{r(1-\epsilon)}\right) \geq 1 - \frac{\epsilon}{t^2} \tag{22}$$

which implies that the probability that a particular $Z[i,j]$ not within $t\sqrt{r(1-\epsilon)}$ from its expectation is at most $\frac{\epsilon}{t^2}$. □