# Robust Network Compressive Sensing

Yi-Chao Chen, Lili Qiu, Yin Zhang
The University of Texas at Austin
{yichao,lili,yzhang}@cs.utexas.edu

Guangtao Xue, Zhenxian Hu
Shanghai Jiao Tong University
{gt_xue,huzhenxian}@sjtu.edu.cn

## ABSTRACT

Networks are constantly generating an enormous amount of rich diverse information. Such information creates exciting opportunities for network analytics. However, a major challenge to enable effective network analytics is the presence of *missing data*, *measurement errors*, and *anomalies*. Despite significant work in network analytics, fundamental issues remain: (i) the existing works do not explicitly account for anomalies or measurement noise, and incur serious performance degradation under significant noise or anomalies, and (ii) they assume network matrices have low-rank structure, which may not hold in reality.

To address these issues, in this paper we develop *LENS decomposition*, a novel technique to accurately decompose a network matrix into a low-rank matrix, a sparse anomaly matrix, an error matrix, and a small noise matrix. LENS has the following nice properties: (i) it is general: it can effectively support matrices with or without anomalies, and having low-rank or not, (ii) its parameters are self tuned so that it can adapt to different types of data, (iii) it is accurate by incorporating domain knowledge, such as temporal locality, spatial locality, and initial estimate (*e.g.*, obtained from models), (iv) it is versatile and can support many applications including missing value interpolation, prediction, and anomaly detection. We apply LENS to a wide range of network matrices from 3G, WiFi, mesh, sensor networks, and the Internet. Our results show that LENS significantly out-performs state-of-the-art compressive sensing schemes.

## Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations – Network monitoring

## General Terms

Algorithms, Measurement, Performance

## Keywords

Compressive Sensing; Traffic Matrix; Anomaly Detection


*MobiCom'14,* September 7-11, 2014, Maui, Hawaii, USA.

http://dx.doi.org/10.1145/2639108.2639129.

## 1. INTRODUCTION

**Motivation:** Wireless networks are constantly generating an enormous amount of rich and diverse information. Such information creates exciting opportunities for network analytics. Network analytics can provide deep insights into the complex interactions among network entities, and has a wide range of applications in wireless networks across all protocol layers. Example applications include spectrum sensing, channel estimation, channel feedback compression, multi-access channel design, data aggregation, network coding, wireless video coding, anomaly detection, and localization.

**Challenges:** A major challenge to enable effective network analytics is the presence of *missing data*, *measurement errors*, and *anomalies*. Failures in measurement systems and network losses can lead to missing data. On the other hand, many network tasks require complete data to operate properly. For example, traffic engineering requires knowing the complete traffic matrix between all source and destination pairs in order to properly provision traffic and avoid congestion. Wireless rate adaptation needs SNR information across all OFDM subcarriers in order to compute effective SNR [16]. Channel selection requires knowledge of received signal strength (RSS) information across all channels in order to select the best channel. Moreover, anomalies and erroneous data are common in real-world network data. It is challenging to distinguish genuine network structure and behavior of interest from anomalies and measurement imperfections in a robust and accurate fashion.

There are numerous approaches to interpolate missing data. They all exploit certain structure in the data. Many studies assume that network matrices (*e.g.*, traffic matrices, delay matrices, RSS matrices) exhibit low-rank structure (*i.e.*, the matrix can be well approximated by the product of two factor matrices with few columns). Accurate extraction of such sparse or low-rank structure is crucial for a variety of network analysis tasks. For example, traffic matrix estimation method *tomo-gravity* [47, 48, 49]) achieves high accuracy by exploiting the fact that traffic matrices can be well approximated using the gravity model, which is a rank-1 matrix. The low-rank nature of Internet delay matrix is essential to the effectiveness of network coordinate systems [34, 30]. The presence of sparse or low-rank structure is also a key assumption behind compressive sensing [8, 14, 7, 37, 38, 50]. For example, Zhang *et al.* develop a novel *spatio-temporal compressive sensing* framework that leverages the spatio-temporal characteristics of real-world traffic matrices and their sparse or low-rank structure.

The immense complexity and heterogeneity of these datasets imply that many assumptions and operational conditions required by existing compressive sensing techniques may not hold [50]. In particular, our analysis show that many real network matrices are not low rank. Violation of low rank assumption significantly reduces the effectiveness of existing compressive sensing approaches.

Another key limitation of existing work is that they do not explicitly account for anomalies or measurement noise. Real-world network datasets often contain a plethora of (i) errors (*e.g.*, missing or erroneous measurements), (ii) noise (introduced by either Mother Nature or measurement procedure itself), and (iii) anomalies (caused by network events such as flash crowds, failures, attacks, *etc.*). The performance of existing approaches seriously degrades under significant noise or anomalies. It remains a fundamental challenge to develop new techniques that can accurately distinguish genuine structure and behavior of interest from errors, noise, and anomalies common in real-world network data.

**Our approach:** In this paper, we first develop ***LENS decomposition***, a novel technique to accurately decompose network data represented in the form of a matrix into a low-rank matrix, a sparse anomaly matrix, an error term, and a small noise matrix. This decomposition naturally reflects the inherent structures of real-world data and is more general than existing compressive sensing techniques by removing the low-rank assumption and explicitly supporting anomalies. We further generalize the problem to incorporate domain knowledge, such as temporal stability, spatial locality, and/or initial estimation (*e.g.*, obtained from a model). We formulate this problem as a convex optimization problem, and develop an Alternating Direction Method (ADM) to efficiently solve it.

Our approach has several nice properties: (i) it supports a wide range of matrices: with or without anomalies, and with or without low-rank structure, (ii) its parameters are either exactly computed or self tuned, (iii) it can incorporate domain knowledge, and (iv) it supports various network applications, such as missing value interpolation, prediction, and anomaly detection.

We evaluate LENS using a wide range of network matrices, including traffic matrices from 3G, WiFi, and the Internet, channel state information (CSI) matrices, RSSI matrices in WiFi and sensor networks, and expected transmission time (ETT) traces from UCSB Meshnet. Our results show that it significantly out-performs state-of-the-art compressive sensing methods including Sparsity Regularized Matrix Factorization (SRMF) under anomalies [50].

# 2. ANALYSIS OF NETWORK MATRICES

| Network | Date | Duration | Resolution | Size |
|---|---|---|---|---|
| 3G traffic | Nov. 2010 | 1 day | 10 min. | $472 \times 144$ |
| WiFi traffic | Jan. 2013 | 1 day | 10 min. | $50 \times 118$ |
| Abilene traffic [2] | Apr. 2003 | 1 week | 10 min. | $121 \times 1008$ |
| GÉANT traffic [42] | Apr. 2005 | 1 week | 15 min. | $529 \times 672$ |
| 1 channel CSI | Feb. 2009 | 15 min. | 1 frame | $90 \times 9000$ |
| multi. channels CSI | Feb. 2014 | 15 min. | 1 frame | $270 \times 5000$ |
| Cister RSSI [35] | Nov. 2010 | 4 hours | 1 frame | $16 \times 10000$ |
| CU RSSI [6] | Aug. 2007 | 500 frames | 1 frame | $895 \times 500$ |
| UMich RSS [43] | April 2006 | 30 min. | 1 frame | $182 \times 3127$ |
| UCSB Meshnet [41] | April. 2006 | 3 days | 1 min. | $425 \times 1527$ |

**Table 1: Datasets under study.**

## 2.1 Datasets

Table 1 lists the network matrices used for our study. We obtained 3G traces from 472 base stations in a downtown area of a major city in China during Oct. 2010. We generate traffic matrices by computing the aggregated traffic every 10 minutes. $M(i,t)$ represents the total traffic volume to and from base station $i$ during time interval $t$, where $t$ is a 10-minute interval.

We also got WiFi traffic from a large university in China, and generated traffic matrices based on the traffic collected at the 50 most loaded access points (APs) on Jan. 4, 2013. $M(i,t)$ denotes the total traffic to/from AP $i$ during the $t$-th 10-minute time interval.

In addition to wireless traffic matrices, we also use traffic matrices from Abilene (Internet2) [2] and GÉANT [42], which are standard traffic matrices used in several previous studies (*e.g.*, [22, 24, 46, 50]) and useful for comparison. Abilene traces report total end-to-end traffic between all pairs in a 11-node network every 10 minutes. GÉANT traces report total traffic between all pairs in a 23-node network every 15 minutes.

For diversity, in addition to traffic matrices, we also obtain signal strength and expected transmission time (ETT). We use the following SNR and RSS matrices: (a) our 1 channel CSI traces, (b) our multi-channel CSI traces, (c) Cister RSSI traces, (d) CU RSSI traces, and (e) UMich RSS. We collected (a) by having a moving desktop transmit back-to-back to another desktop and letting the receiving desktop record the SNR across all OFDM subcarriers over 15 minutes using the Intel Wi-Fi Link 5300 (iwl5300) adapter. The modified driver [20] reports the channel matrices for 30 subcarrier groups in a 20MHz channel, which is about one group for every two subcarriers according to the standard [3]. The sender sends 1000-byte frames using MCS 0 at a transmission power of 15 dBm. Since MCS0 has 1 stream and the receiver has 3 antennas, the NIC reports CSI as a $90 \times 1$ matrix for each frame. We collect (b) on channels 36, 40, 44, 48, 149, 153, 157, 161, and 165 at 5GHz. The transmitter starts from channel 36, and sends 10 packets before switching to the next channel. The receiver synchronizes with the transmitter and also cycle through the 9 channels, which yields $270 \times 1$ matrix for each frame. In addition, we use Cister RSSI traces [35], CU RSSI traces [6], and UMich RSS traces [43], all of which are publicly available at CRAWDAD [12]. $M(f,i)$ in Cister traces denotes RSSI on IEEE 802.15.4 channel $f$ in the $i$-th frame, $M(l,i)$ in CU traces denotes RSSI of the $i$-th frame at location $l$, and $M(s,i)$ in UMich-RSS trace denotes the RSS measurement received by the $s$-th sensor pair in the $i$-th packet. We also use ETT traces from UCSB Meshnet [41], which contains the ETT of every links in a 20-node mesh network. We generate the ETT matrix $M(l,t)$, where $M(l,t)$ denotes the ETT of the link $l$ during the $t$-th 10-second window.

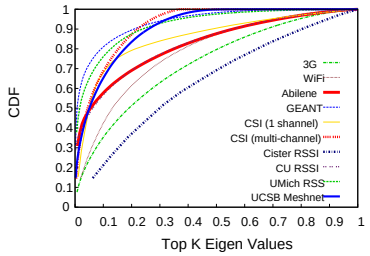


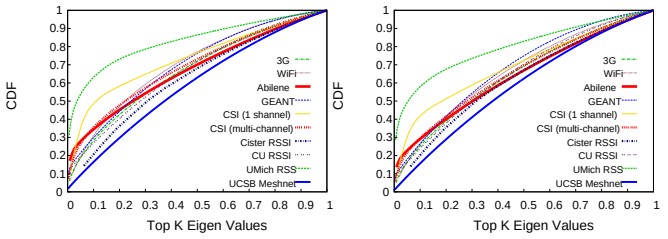

**Figure 1: CDF of energy that are contained in the top K singular values of the original matrices.**

## 2.2 Analysis

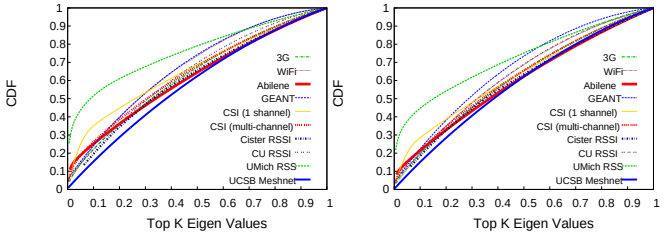
**Rank analysis:** For each network matrix, we mean center each row (*i.e.*, subtract from each row its mean value). We then apply singular value decomposition (SVD) to examine if the mean-centered matrix has a good low-rank approximation. The metric we use is the fraction of total variance captured by the top $K$ singular values, *i.e.*, $\left(\sum_{i=1}^{K} s_i^2\right) / \left(\sum_i s_i^2\right)$, where $s_i$ is the $i$-th largest singular value and $\left(\sum_i s_i^2\right)$ gives the total variance of the mean-centered coordinate matrix. Note that $1 - \left(\sum_{i=1}^{K} s_i^2\right) / \left(\sum_i s_i^2\right)$ is the relative approximation error of the best rank-$K$ approximation with respect to the squared Frobenius norm.

Figure 1 plots the fraction of total variance captured by the top $K$ singular values for different traces. As it shows, from low to high, UCSB Meshnet, GÉANT, multi-channel CSI, UMich RSS, RON, 1-channel CSI, CU RSSI, Abilene, WiFi, 3G, and Cister RSSI matrices take 7.5%, 20.8%, 22.0%, 23.9%, 47.2%, 48.9%,

55.8%, 57.0%, 58.0%, 68.1%, and 81.0% singular values to capture 90% variance, respectively. Therefore, only UCSB Meshnet, GÉANT, multi-channel CSI, and UMich RSS are close to low rank.



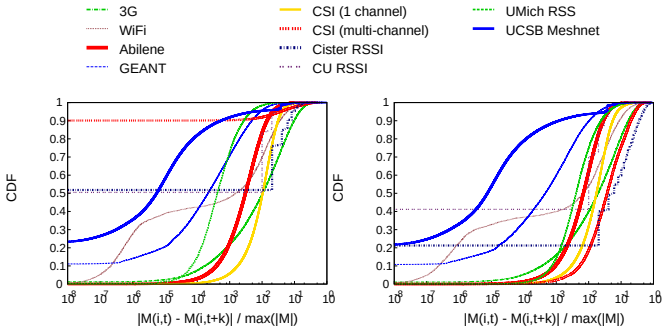

(a) 5% anomalies with s=0.5 (b) 10% anomalies with s=0.5




(c) 5% anomalies with s=1 (d) 10% anomalies with s=1

**Figure 2: CDF of energy that are contained in the top K singular values under anomalies in traffic matrices.**

Next we inject anomalies to see how it affects the results. We inject anomalies to a portion of the entries in the original matrices. Following the standard anomaly injection method used in existing work [22, 18, 31], we first use exponential weighted moving average (EWMA) to predict the future entries based on their past values (*i.e.*, $y = \alpha x + (1-\alpha)y$, where $\alpha = 0.8$ in our evaluation) and use the maximum difference between the actual and predicted value as the anomaly size to be injected. We vary the fraction of entries to inject anomalies from 5% to 10%, and also scale the anomaly size by $s$, which is 0.5 or 1. As shown in Figure 2, when we inject more anomalies or larger anomalies, more singular values are required in order to capture the variance of the matrices. This trend is consistent across all traces. For example, as shown in Figure 2(a)-(b), when we inject 5% and 10% anomalies with s=0.5, it takes 60.9% and 67.6% singular values to capture 90% variance in UMich Meshnet, 71.0% and 75.8% in WiFi trace, and 75.7% and 80.0% in 1-channel CSI trace. As shown in Figure 2(c)-(d), when we inject 5% and 10% anomalies with s=1, the corresponding numbers are 72.5% and 76.9% in UMich Meshnet, 72.0% and 78.3% in WiFi trace, and 81.7% and 84.1% in 1-channel CSI trace. The other matrices exhibit the same trend.




(a) k=1 (b) k=10

**Figure 3: CDF of normalized difference between i-th and i+k-th time slot.**

**Temporal stability:** Figure 3 plots the CDF of normalized temporal variation (*i.e.*, $\frac{x(i)-x(i-t)}{max(x(i))}$) across different traces. As it shows, different traces exhibit varying degrees of temporal stability. For example, 3G and Cister RSSI have high variation: the two adjacent entries differ by 6.1%-9.8% in 90% cases, and 10 time-slot apart entries differ by 16.7%-36.2% in 90%. In comparison, UMich Meshnet and GÉANT have low variation, where the adjacent entries differ by 0.3%-0.5% in 90% cases and 10 time-slot apart entries differ by 1.0%-2.4% in 90%. The other traces are in between.

**Summary:** The major findings from the above analysis include: (i) Not all real network matrices are low rank. (ii) Adding anomalies further increases the rank. (iii) Temporal stability varies substantially across different traces. These findings motivate us to develop a general compressive sensing framework to support diverse matrices that may not be low rank, exhibit different degrees of temporal stability, and may even contain anomalies.

# 3. LENS DECOMPOSITION

In this section, we first present LENS decomposition framework. Next we develop an alternating direction method for solving the decomposition problem. Then we describe how to set the parameters.

## 3.1 LENS Decomposition Framework

**Overview:** There are many factors that contribute to real network matrices, including measurement errors, anomalies and inherent noise. To capture this insight, we decompose the original matrix into a Low-rank component, an Error component, a Noise component, and a Sparse anomaly component (hence the acronym LENS decomposition). This is motivated by the following observations:

- Low-rank component: Network matrices often exhibit significant redundancy. A concrete form of redundancy is that the network matrix of interest can be well approximated by low-rank matrices. For example, TM estimation makes use of the gravity model [47], which is essentially a rank-1 approximation to matrices. [36] uses low-rank matrices for localization.

- Sparse component: Anomalies are common in large network dataset. Anomalies may arise from a number of factors. For example, traffic anomalies may be caused by problems ranging from security threats (*e.g.*, Distributed Denial of Service (DDoS) attacks and network worms) to unusual traffic events (*e.g.*, flash crowds), to vendor implementation bugs, and to network misconfigurations. Such anomalies are typically not known a priori and are sparse [11, 25].
  Note that there can be systematic effects that are only sparse after some transformation (*e.g.*, wavelet transform). For example, a major level shift may result in persistent changes in the original data (and is thus not sparse). But after wavelet transform (or simple temporal differencing), it becomes sparse.

- Error and artifacts: The measurement and data collection procedure may also introduce artifacts. For example, a SNMP traffic counter may wrap around, resulting in negative measurements. One can always try his best to apply domain knowledge to filter out obvious errors and artifacts (*e.g.*, missing data or negative traffic measurements). However, in general it is difficult to filter out all such artifacts. The advantage of considering both anomalies and errors jointly is that the parts that cannot be filtered can get absorbed by the sparse components.

- Noise. Noise is universal, making clean mathematical models "approximate" in practice. For example, real-world network

matrices are typically only approximately low-rank as opposed to exactly low-rank.

Therefore a natural approach is to consider the original dataset as a mixture of all these effects. It is useful if one can decompose the original matrix into individual components, each component capturing one major effect.

**Basic formulation:** The basic LENS decomposition decomposes an original $m \times n$ data matrix $D$ into a low-rank matrix $X$, a sparse anomaly matrix $Y$, a noise matrix $Z$, and an error matrix $W$. This is achieved by solving the following *convex* optimization problem:

$$\begin{aligned} \text{minimize:} \quad & \alpha\|X\|_* + \beta\|Y\|_1 + \frac{1}{2\sigma}\|Z\|_F^2, \\ \text{subject to:} \quad & X + Y + Z + W = D, \\ & E. * W = W. \end{aligned} \tag{1}$$

where $X$ is a low-rank component, $Y$ is a sparse anomaly component, $Z$ is a dense noise term, and $E$ is a binary error indicator matrix such that $E[i,j] = 1$ if and only if entry $D[i,j]$ is erroneous or missing, and $W$ is an arbitrary error component with $W[i,j] \neq 0$ only when $E[i,j] = 1$ (thus $E. * W = W$, where $.*$ is an element-wise multiplication operator). Since $W$ fully captures the erroneous or missing values, we can set $D[i,j] = 0$ whenever $E[i,j] = 1$ without loss of generality. The constraint enforces $D$ to be the sum of $X$, $Y$, and $Z$ when $D$ is neither missing nor has errors (since $E[i,j]. * W[i,j] = 0$ in this case), while imposing no constraint when $D$ is missing or has error (since $E[i,j]. * W[i,j] = W[i,j]$ allows $W[i,j]$ to take an arbitrary value to satisfy $X + Y + Z + W = D$).

The optimization objective has the following three components:

- $\|X\|_*$ is the nuclear norm [38, 37] of matrix $X$, which penalizes against high rank of $X$ and can be computed as the total sum of $X$'s singular values.
- $\|Y\|_1$ is the $\ell_1$-norm of $Y$, which penalizes against lack of sparsity in $Y$ and can be computed as $\|Y\|_1 = \sum_{i,j} |Y[i,j]|$.
- $\|Z\|_F^2$ is the squared Frobenius norm of matrix $Z$, which penalizes against large entries in the noise matrix $Z$ and can be computed as $\|Z\|_F^2 = \sum_{i,j} Z[i,j]^2$.

The weights $\alpha$, $\beta$ and $\frac{1}{2\sigma}$ balance the conflicting goals to simultaneously minimize $\|X\|_*$, $\|Y\|_1$ and $\|Z\|_F^2$. We describe how to choose these weights in Section 3.3.

**Generalized formulation:** Below we generalize both the constraints and the optimization objective of the basic formulation in Eq. (1) to accommodate rich requirements in the analysis of real-world network matrices.

First, the matrix of interest may not always be directly observable, but its linear transform can be observed though subject to missing data, measurement errors, and anomalies. For example, end-to-end traffic matrices $X$ are often not directly observed, and what can be observed are link load $D$. $X$ and $D$ follow $AX = D$, where $A$ is a binary routing matrix: $A(i,j) = 1$ if link $i$ is used to route traffic for the $j$-th end-to-end flow, and $A(i,j) = 0$ otherwise. We generalize the constraints in Eq. (1) to cope with such measurement requirements:

$$AX + BY + CZ + W = D \tag{2}$$

Here $A$ may capture tomographic constraints that linearly relate direct and indirect measurements (*e.g.*, $A$ is a routing matrix in the traffic matrices). $B$ may represent an over-complete anomaly profile matrix. If we do not know which matrix entries may have anomalies, we can simply set $B$ to the identity matrix $I$. It is also possible to set $B = A$ if we are interested in capturing anomalies in $X$. Without prior knowledge, we set $C$ to be the identity matrix.

Prior research on network inference and compressive sensing highlights the importance of incorporating domain knowledge about the structure of the underlying data. To capture domain knowledge, we introduce one or more penalty terms into the optimization objective: $\sum_{k=1}^{K} \|P_k X Q_k^T - R_k\|_F^2$, where $K$ is the number of penalty terms. We also introduce a weight $\gamma$ to capture our confidence in such knowledge.

Examples of domain knowledge include temporal stability constraints, spatial locality constraints, and initial estimation of $X$ (*e.g.*, [47] derives initial traffic matrices using the gravity model [21]). Temporal and spatial locality are common in network data [32, 44, 15]. Such domain knowledge is especially helpfulwhen there are many missing entries, making the problem severely under-constrained.

Consider a few simple cases. First, when $k = 1$, $P_1$ is an identity matrix $I$, $R_1$ is a zero vector, we can set $Q_1 = Toeplitz(0, 1, -1)$, which denotes the Toeplitz matrix with central diagonal given by ones, the first upper diagonal given by negative one, *i.e.*,

$$Q = \begin{bmatrix} 1 & -1 & 0 & 0 & \dots \\ 0 & 1 & -1 & 0 & \ddots \\ 0 & 0 & 1 & -1 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}. \tag{3}$$

$Q^T$ denotes the transpose of matrix $Q$. $P_1 X Q_1^T$ captures the differences between two temporally adjacent elements in $X$. Minimizing $\|P_1 X Q_1^T - R_1\|_F^2 = \|P_1 X Q_1^T\|_F^2$ reflects the goal of making $X$ temporally stable. For simplicity, this is what we use for our evaluation. In general, one can use similar constraints to capture other temporal locality patterns during different periods (*e.g.*, seasonal patterns or diurnal patterns).

Next we consider the spatial locality, which is represented by $P$. If $k = 1$, $R_1 = 0$, $Q_1$ is an identity matrix $I$, we can set $P_1$ to reflect the spatial locality. For example, if two adjacent elements in the matrix have similar values, we can set $P = Toeplitz(0, 1, -1)$. Similarly, if different parts of the matrix have different spatial locality patterns, we can use different $P$'s to capture these spatial locality patterns. For simplicity, our evaluation considers only temporal stability, which is well known to exist in different networks. We plan to incorporate spatial locality in the future.

Finally, if we have good initial estimate of $X_{init}$ (*e.g.*, [47] uses the gravity model to derive the initial TM), we can leverage this domain knowledge by minimizing $\|X - X_{init}\|$ (*i.e.*, $R_1 = X_{init}$). This term can be further combined with spatial and/or temporal locality to produce richer constraints.

Putting everything together, the general formulation is:

$$\begin{aligned} \text{minimize:} \quad & \alpha\|X\|_* + \beta\|Y\|_1 + \frac{1}{2\sigma}\|Z\|_F^2 + \\ & \frac{\gamma}{2\sigma}\sum_{k=1}^{K} \|P_k X Q_k^T - R_k\|_F^2, \\ \text{subject to:} \quad & AX + BY + CZ + W = D \\ & E. * W = W. \end{aligned} \tag{4}$$

Note that our formulation is more general than recent research on compressive sensing (*e.g.*, [50, 9, 38, 37]), which do not consider anomalies, have simpler constraints (*e.g.*, there is no $A$, $B$, or $C$), and have less general objectives.

## 3.2 Optimization Algorithm

The generality of the formulation in Eq. (4) makes it challenging to optimize. We are not aware of any existing work on compressive sensing that can cope with such a general formulation. Below we first reformulate Eq. (4) to make it easier to solve. We then consider the augmented Lagrangian function of the reformulated problem and develop an Alternating Direction Method to solve it.

**Reformulation for optimization:** Note that $X$ and $Y$ appear in multiple locations in the objective function and constraints in the optimization problem 4. This coupling makes optimization difficult. To reduce coupling, we introduce a set of auxiliary variables $X_0, X_1, \cdots, X_K$ and $Y_0$ and reformulate the problem as follows:

$$\begin{aligned} \text{minimize:} \quad & \alpha\|X\|_* + \beta\|Y\|_1 + \frac{1}{2\sigma}\|Z\|_F^2 \\ & + \frac{\gamma}{2\sigma}\sum_{k=1}^{K}\|P_k X_k Q_k^T - R_k\|_F^2, \\ \text{subject to:} \quad & AX_0 + BY_0 + CZ + W = D, \\ & E. * W = W, \\ & X_k - X = 0 \quad (\forall k = 0, \cdots, K), \\ & Y_0 - Y = 0. \end{aligned} \tag{5}$$

where $Y_0$ and $X_k (0 \le k \le K)$ are auxiliary variables. Note that formulations Eq. (5) and Eq. (4) are equivalent.

**Alternating Direction Method for solving (5):** We apply an Alternating Direction Method (ADM) [4] to solve the convex optimization problem in (5). Specifically, we consider the augmented Lagrangian function:

$$\begin{aligned} & \mathcal{L}(X, \{X_k\}, Y, Y_0, Z, W, M, \{M_k\}, N, \mu) \\ \triangleq \quad & \alpha\|X\|_* + \beta\|Y\|_1 + \frac{1}{2\sigma}\|Z\|_F^2 \\ & + \frac{\gamma}{2\sigma}\sum_{k=1}^{K}\|P_k X_k Q_k^T - R_k\|_F^2 \\ & + \langle M, D - AX_0 - BY_0 - CZ - W\rangle && (6) \\ & + \sum_{k=0}^{K}\langle M_k, X_k - X\rangle && (7) \\ & + \langle N, Y_0 - Y\rangle && (8) \\ & + \frac{\mu}{2}\cdot\|D - AX_0 - BY_0 - CZ - W\|_F^2 && (9) \\ & + \frac{\mu}{2}\cdot\sum_{k=0}^{K}\|X_k - X\|_F^2 && (10) \\ & + \frac{\mu}{2}\cdot\|Y_0 - Y\|_F^2 && (11) \end{aligned}$$

where $M$, $\{M_k\}$, $N$ are the Lagrangian multipliers [1] for the equality constraints in Eq. (5), and $\langle U, V\rangle \triangleq \sum_{i,j}(U[i,j]\cdot V[i,j])$ for two matrices $U$ and $V$ (of the same size). Essentially, the augmented Lagrangian function includes the original objective, three Lagrange multiplier terms (6)–(8), and three penalty terms converted from the equality constraints (9)–(11). Lagrange multipliers are commonly used to convert an optimization problem with equality constraints into an unconstrained one. Specifically, for any optimal solution that minimizes the (augmented) Lagrangian function, the partial derivatives with respect to the Lagrange multipliers must be 0. Hence the original equality constraints will be satisfied. The penalty terms enforce the constraints to be satisfied. The benefit of including Lagrange multiplier terms in addition to the penalty terms is that $\mu$ no longer needs to iteratively increase to $\infty$ to solve the original constrained problem, thereby avoiding ill-conditioning [4]. Note that we do not include terms corresponding to constraint $E. * W = W$ in the augmented Lagrangian function, because it is straightforward to enforce this constraint during each iteration of the Alternating Direction Method without the need for introducing additional Lagrange multipliers.

The ADM algorithm progresses in an iterative fashion. During each iteration, we alternate among the optimization of the augmented Lagrangian function by varying each one of $X$, $\{X_k\}$, $Y$, $Y_0$, $Z$, $W$, $M$, $\{M_k\}$, $N$ while fixing the other variables. Introducing auxiliary variables $\{X_k\}$ and $Y_0$ makes it possible to obtain a close-form solution for each optimization step. Following ADM, we increase $\mu$ by a constant factor $\rho \ge 1$ during each iteration. When involving only two components, ADM is guaranteed to converge quickly. In our general formulation, convergence is no longer guaranteed, though empirically we observe quick convergence in all our experiments (*e.g.*, as shown in Section 4). We plan to apply techniques in [13] to ensure guaranteed convergence in future work. We further improve efficiency by replacing exact optimization with approximate optimization during each iteration. Appendix A gives a detailed description on the steps during each iteration.

**Improving efficiency through approximate SVD:** The most time-consuming operation during each iteration of the Alternating Direction Method is performing the singular value decomposition. In our implementation, we add an additional constraint on the rank of matrix $X$: $rank(X) \le r$, where $r$ is a user-specified parameter that represents an estimated upper bound on the true rank of $X$. We then explicitly maintain the SVD of $X$ and update it approximately during each iteration through the help of rank-revealing QR factorization of matrices that have only $r$ columns (which are much smaller than the original matrices used in SVD). We omit the details of approximate SVD in the interest of space.

## 3.3 Setting Parameters

**Setting $\alpha$, $\beta$ and $\sigma$:** A major advantage of our LENS decomposition is that a good choice of the parameters $\alpha$ and $\beta$ can be analytically determined without requiring any manual tuning. Specifically, let $\sigma_D$ be the standard deviation of measurement noise in data matrix $D$ (excluding the effect of low-rank, sparse, and error terms). For now, we assume that $\sigma_D$ is known, and we will describe how to determine $\sigma_D$ later in this section. Moreover, we first ignore the domain knowledge term and will adaptively set $\gamma$ for the domain knowledge term based on the given $\alpha$ and $\beta$.

Let density $\eta(D) = 1 - \frac{\sum_{i,j} E[i,j]}{m \times n}$ be the fraction of entries in $D$ that are neither missing nor erroneous, where the size of $D$ is $m \times n$ and the size of $Y$ is $m_Y \times n_Y$. $E[i,j]$ can be estimated based on domain knowledge. For example, we set $E[i,j] = 1$ if the corresponding entry takes a value outside its normal range (*e.g.*, a negative traffic counter) or measurement software reports an error on the entry. Moreover, our evaluation shows that LENS is robust against estimation error in $\eta(D)$.

We propose to set:

$$\alpha = (\sqrt{m_X} + \sqrt{n_X}) \cdot \sqrt{\eta(D)} \tag{12}$$

$$\beta = \sqrt{2 \cdot \log(m_Y \cdot n_Y)} \tag{13}$$

$$\sigma = \theta \cdot \sigma_D \tag{14}$$

where $(m_X, n_X)$ is the size of $X$, $(m_Y, n_Y)$ is the size of $Y$. $\theta$ is a user-specified control parameter that limits the contamination of the dense measurement noise $\sigma_D$ when computing $X$ and $Y$. In all our experiments, we set $\theta = 10$, though it is also possible to choose $\theta$ adaptively, just like how we choose $\gamma$ as described later in this section.

Below we provide some intuition behind the above choices of $\alpha$ and $\beta$ using the basic formulation in Eq. (1). The basic strategy is to consider all variables except one are fixed. Our evaluation shows that these choices work well in practice.

**Intuition behind the choice of $\alpha$:** Consider the special case when all variables except that $X$ are already given and stay fixed. Then we just need to solve:

$$\min_{X} \quad \alpha \cdot \|X\|_* + \frac{1}{2\sigma} \cdot \|D - X - Y - W\|_F^2 \qquad (15)$$

since $Z = D - X - Y - W$. We can prove the optimal $X$ in Eq. (15) can be obtained by performing soft-thresholding (a.k.a., shrinkage) on the singular values of $D - Y - W$. That is,

$$\begin{aligned} X_{\text{opt}} &= \mathsf{SVSoftThresh}(D - Y - W,\ \alpha \cdot \sigma) \\ &\triangleq U * \mathsf{SoftThresh}(S,\ \alpha \cdot \sigma) * V^T, \end{aligned} \qquad (16)$$

where $[U, S, V] = \mathsf{svd}(D - Y - W)$ is the singular value decomposition of $(D - Y - W)$ (thus $D - Y - W = USV^T$), and $\mathsf{SoftThresh}(S,\ \alpha\sigma) = \mathsf{sign}(S). * \max\{0,\ \mathsf{abs}(S) - \alpha\sigma\}$ ($\mathsf{sign}(S) = S./\mathrm{abs}(S)$). Intuitively, soft-thresholding eliminates the contamination on the singular values of $X$ due to the dense measurement noise $\sigma_D$.

From asymptotic random matrix theory [33], for a random matrix with entries drawn $i.i.d.$ from a Gaussian distribution with probability $\eta(D)$, its norm (*i.e.*, the largest singular value) is bounded by $O((\sqrt{m} + \sqrt{n}) \cdot \sqrt{\eta(D)} \cdot \sigma_D)$ with a high probability. So a good heuristic is to set the soft threshold to:

$$\alpha \cdot \sigma = (\sqrt{m} + \sqrt{n}) \cdot \sqrt{\eta(D)} \cdot \sigma_D \cdot \theta,$$

where $\theta$ is a control parameter that captures the desired separation from the influence of dense measurement noise $\sigma_D$. Therefore, we simply set $\alpha = (\sqrt{m} + \sqrt{n}) \cdot \sqrt{\eta(D)}$ and $\sigma = \theta \cdot \sigma_D$.

**Intuition behind the choice of $\beta$:** Now suppose $X$ is given and we need to solve:

$$\min_{Y} \quad \beta\dot{\|}Y\|_1 + \frac{1}{2\sigma} \cdot \|D - X - Y - W\|_F^2 \qquad (17)$$

We can prove that the optimal $Y$ for (17) can be obtained by performing soft-thresholding (a.k.a., shrinkage) on the entries of $D - X - W$. Specifically, we have:

$$Y_{\text{opt}} = \mathsf{SoftThresh}(D - X - W,\ \beta \cdot \sigma), \qquad (18)$$

where soft-thresholding eliminates the contamination on entries of $Y$ due to the dense measurement noise.

In the context of standard compressive sensing setting:

$$\min_{y} \quad \beta * \sigma_d \cdot \|y\|_1 + \frac{1}{2} \cdot \|y - d\|_2^2,$$

where $y$ is a vector of length $n_y$, and $\sigma_d$ is the standard deviation of the vector of observables $d$. As justified in Basis Pursuit De-Noising (BPDN) [10], a penalty term $\beta = \sqrt{2 \cdot \log(n_y)}$ should be used, where $n_y$ is the number of elements in $Y$. Similarly, in our context, a good heuristic is to set the soft threshold to:

$$\beta \cdot \sigma = \sqrt{2 \cdot \log(m_Y \cdot n_Y)} \cdot \sigma_D \cdot \theta.$$

So we simply set $\beta = \sqrt{2 \cdot \log(m_Y \cdot n_Y)}$ and $\sigma = \theta \cdot \sigma_D$.

**Estimating $\sigma_D$:** When $\sigma_D$ is not known in advance, we simply estimate it from entries of $D - AX_0 - BY_0 - W$ during each iteration in the Alternating Direction Method (see Appendix A). Specifically, let $J = D - AX_0 - BY_0 - W$, we estimate $\sigma_D$ as the standard deviation of $\{J[i, j] \mid E[i, j] = 0\}$. It is also possible to use a more robust estimator (*e.g.*, the mean absolute value), which gives similar performance in our experiments.

**Searching for $\gamma$:** $\gamma$ reflects the importance of domain knowledge terms. It is challenging to find an appropriate $\gamma$, since its value depends on how valuable are the domain knowledge versus the information from the measurement data. Therefore instead of using a fixed value, we automatically learn $\gamma$ without user feedback or ground-truth of the real missing entries as follows. Given the incomplete data matrix $D$, we further drop additional entries of $D$ and apply our algorithm under several $\gamma$ values, and quantify the error of fitting the entries that were present in $D$ but dropped intentionally during the search (so we know their true values). We adopt the value of $\gamma$ that gives the lowest fitting error on these entries as the final $\gamma$ and apply it to our final matrix interpolation, which only has the real missing elements.

**Supporting the general formulation:** In the general formulation in Eq. (4), we first ensure that matrices $A$, $B$, $C$ are properly scaled such that each column of $A$, $B$, $C$ has unit length (i.e. the square sum of all elements in a column is equal to 1). We also automatically scale $P_k$, $Q_k$, $R_k$ such that each row of $P_k$ and $Q_k$ has unit length. We then use the same choice of $\alpha$, $\beta$, $\sigma$, and $\gamma$ as in the basic formulation.

# 4. EVALUATION

## 4.1 Evaluation Methodology

**Performance metrics:** We quantify the performance in terms of estimation error of the missing entries and anomaly detection accuracy. We drop data from existing network matrices and compare our estimation with the ground truth. We use Normalized Mean Absolute Error (NMAE) to quantify the estimation error. NMAE is defined as follow:

$$NMAE = \frac{\sum_{i,j:M(i,j)=0} |X(i,j) - \hat{X}(i,j)|}{\sum_{i,j:M(i,j)=0} |X(i,j)|}, \qquad (19)$$

where $X$ and $\hat{X}$ are the original and estimated matrices, respectively. We only measure errors on the missing entries. For each setting, we conduct 10 random runs, which drop random set of data, and report an average of these 10 runs.

We quantify the anomaly detection accuracy using *F1-score* [45], which is the harmonic mean of precision and recall: $\textit{F1-score} = \frac{2}{1/precision + 1/recall}$, where precision is the fraction of anomalies found by anomaly detection schemes that are indeed real anomalies we injected and recall is the fraction of real anomalies that are correctly identified by anomaly detection schemes. The higher F1-score, the better. F1-score of 1 is perfect. We report an average of 10 random runs.

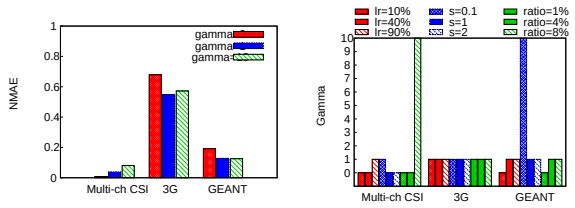
**Anomaly generation:** As mentioned in Section 2.2, we find the maximum difference between the original trace and the EWMA prediction, and then inject the anomaly of this size to the trace. We vary the anomaly size using different scaling factors $s$ and the fraction of anomalies to understand their impacts.

**Different dropping modes:** As in [50], we drop data in the following ways: (i) PureRandLoss: elements in a matrix are dropped independently with a random loss rate; (ii) xxTimeRandLoss: xx% of columns in a matrix are selected and the elements in these selected columns are dropped with a probability $p$ to emulate random losses during certain times (*e.g.*, disk becomes full); (iii) xxElemRandLoss: xx% of rows in a matrix are selected and the elements in these selected rows are dropped with a probability $p$ to emulate

certain nodes lose data (*e.g.*, due to battery drain); (iv) xxElemSyncLoss: the intersection of xx% of rows and p% of columns in a matrix are dropped to emulate a group of nodes experience the same loss events at the same time; (v) RowRandLoss: drop random rows to emulate node failures, and (vi) ColRandLoss: drop random columns for completeness. We use PureRandLoss as the default, and further use other loss models to understand impacts of different loss models. We feed the matrices after dropping as the input to LENS, and use LENS to fill in the missing entries.

**Schemes evaluated:** We compare the following schemes:

- Base: It approximates the original matrix as a rank-2 approximation matrix $X_{base} = \overline{X} + X_{row}1^T + 1X_{col}^T$, where 1 is a column vector consisting of all ones and $X_{row}$ and $X_{col}$ are computed using a regularized least square according to [50].
- SVD Base: As shown in [50], SVD Base, which applies SVD to $X - X_{base}$, out-performs SVD applied directly to $X$. We observe similar results, so below we only include SVD Base.
- SVD Base + KNN: We obtain the result from SVD Base and then apply $K$ nearest neighbors (KNN) to perform local interpolation to leverage the local structure.
- SRMF: Sparsity Regularized Matrix Factorization (SRMF) leverages both low-rank and spatio-temporal characteristics [50].
- SRMF+KNN: It combines SRMF results with local interpolation via KNN [50].
- LENS: We use the output from the LEN decomposition as described in Section 3.



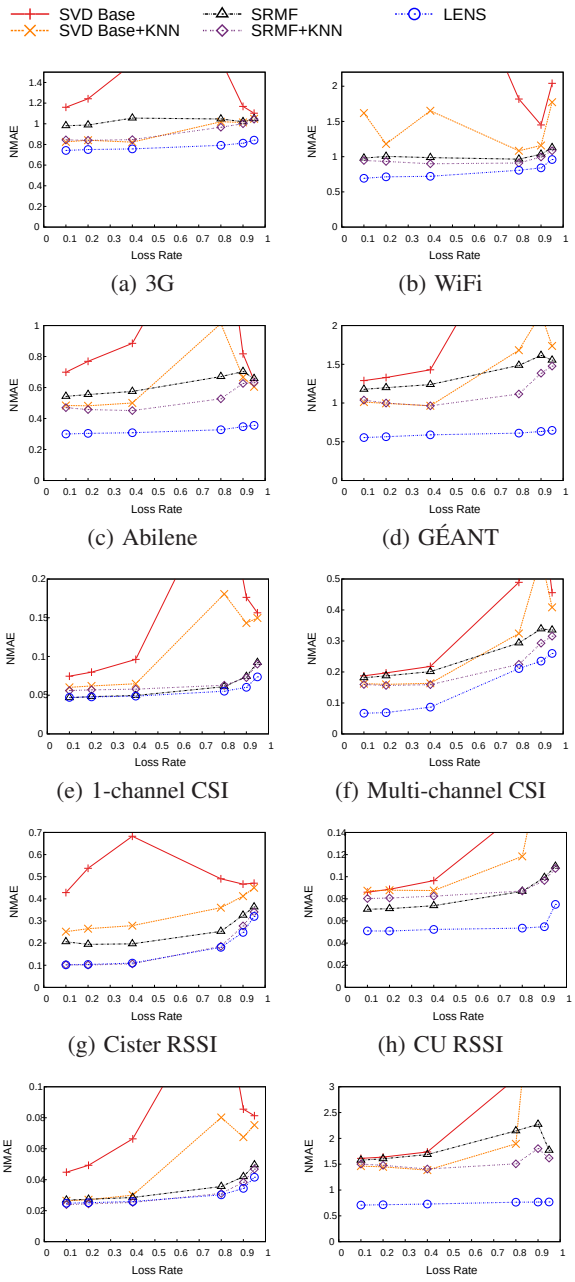

(a) NMAE of different $\gamma$ (b) $\gamma$ under different conditions

**Figure 4: (a) NMAE of different $\gamma$: learned $\gamma$ is 0 for multi-channel CSI, 1 for 3G, and 10 for GÉANT under no anomalies. (b) The learned $\gamma$ under loss rates = 10%, 40%, 90%; anomaly size $s$ = 0.1, 1, 2; ratio of anomalies = 1%, 4%, 8%.**

## 4.2 Performance Results

**Self learned $\gamma$:** LENS supports many types of domain knowledge as described in Sec. 3.1. Our evaluation only considered temporal stability for simplicity and $\gamma$ reflects its importance. To illustrate the benefit of self learning, Figure 4 (a) shows the performance under different $\gamma$ values and different traces. Figure 4 (b) shows the best $\gamma$ under different traces, loss rates, anomaly sizes, and ratio of anomalies. There does not exist a single $\gamma$ that works well for all traces or conditions. Self tuning allows us to automatically select the best $\gamma$ for these traces and achieves low NMAE in all cases.

**Varying dropping rates:** We first compare different schemes in terms of interpolation accuracy measured using NMAE. Figure 5 shows the interpolation error as we randomly inject anomalies to 5% elements with $s = 1$. For clarity of the graphs, we cap the y-axis so that we can focus on the most interesting parts of the graphs. We observe that LENS consistently out-performs the other schemes. In terms of NMAE, LENS < SRMF + KNN < SRMF < SVD Base + KNN < SVD Base. LENS reduces NMAE by 35.5% over SRMF, 27.8% over SRMF+KNN, 59.8% over SVD

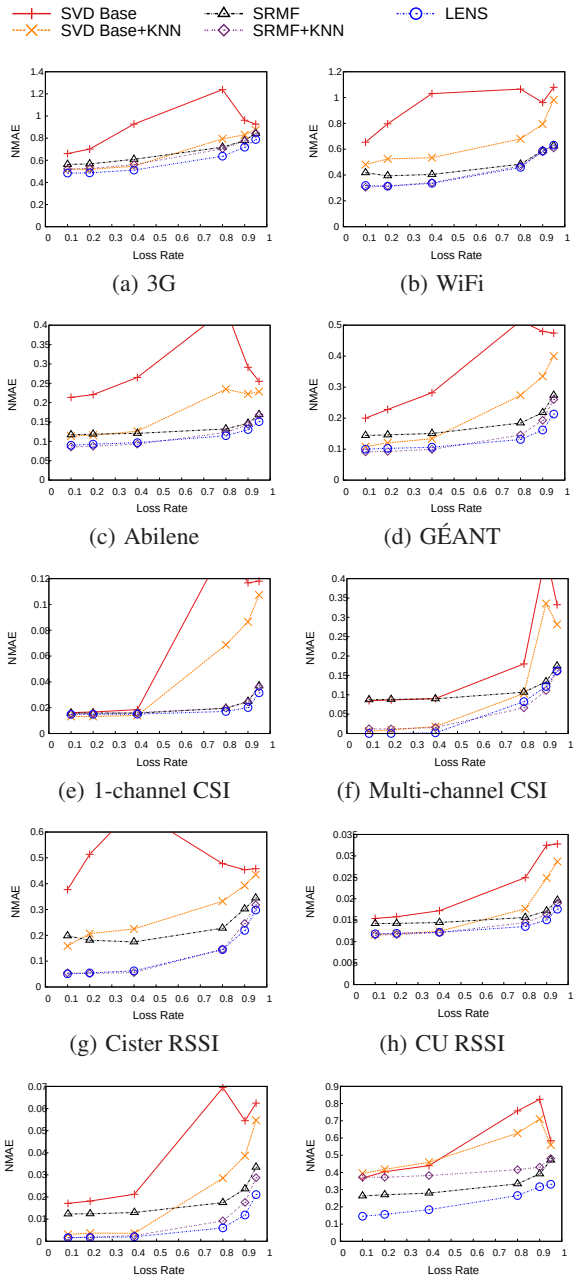



(a) 3G (b) WiFi
(c) Abilene (d) GÉANT
(e) 1-channel CSI (f) Multi-channel CSI
(g) Cister RSSI (h) CU RSSI
(i) UMich RSS (j) UCSB Meshnet

**Figure 5: Interpolation performance under varying data loss rates under 5% anomalies and $s = 1$.**

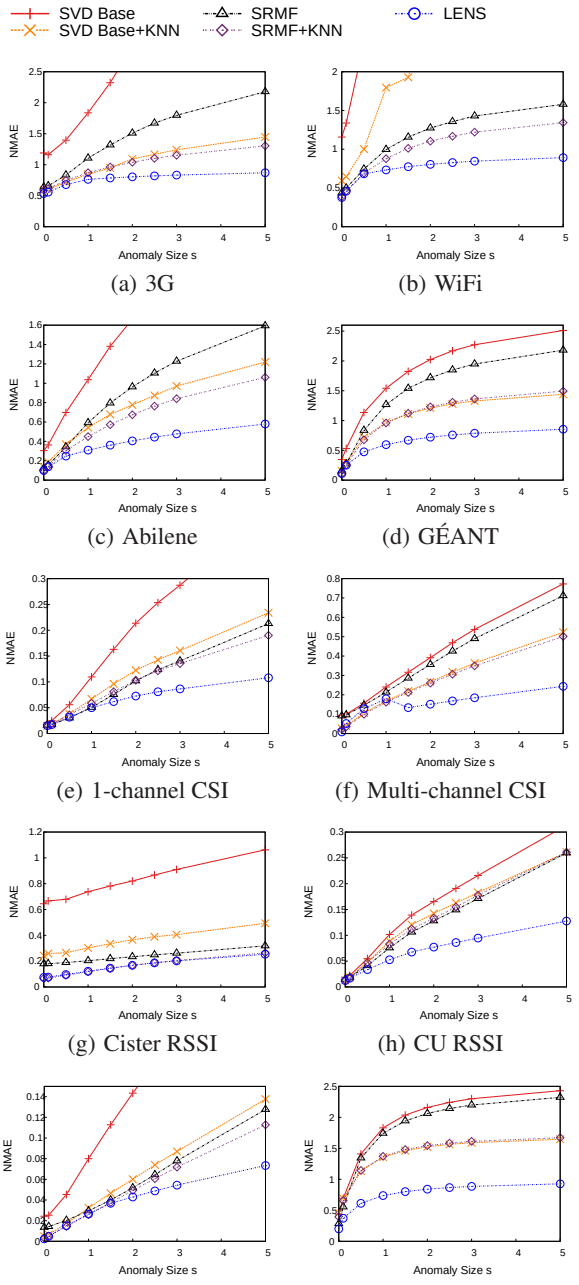
Base, and 44.9% over SVD Base + KNN on average. Moreover, the error is low for high-rank matrices. For example, the highest rank matrices in our datasets are 1-channel CSI, CU RSSI, Abilene, WiFi, 3G, and Cister RSSI matrices. Their corresponding NMAE are 0.05, 0.05, 0.3, 0.69, 0.74, 0.1, respectively. Most of them have low errors except WiFi and 3G. The error does not monotonically increase with the loss rate because an increasing loss rate reduces the number of anomalies, which may help reduce the error.

Figure 6 summarizes the results under varying data loss rates and no anomaly. In most traces, LENS performs comparably to SRMF+KNN, the best known algorithm under no anomaly. In UCSB Meshnet, LENS already out-performs SRMF+KNN even without injecting additional anomalies. This is likely because the trace has more anomalies before our anomaly injection. In UCSB




(a) 3G (b) WiFi (c) Abilene (d) GÉANT (e) 1-channel CSI (f) Multi-channel CSI (g) Cister RSSI (h) CU RSSI (i) UMich RSS (j) UCSB Meshnet

**Figure 6: Interpolation performance under varying data loss rates and no anomaly.**




(a) 3G (b) WiFi (c) Abilene (d) GÉANT (e) 1-channel CSI (f) Multi-channel CSI (g) Cister RSSI (h) CU RSSI (i) UMich RSS (j) UCSB Meshnet

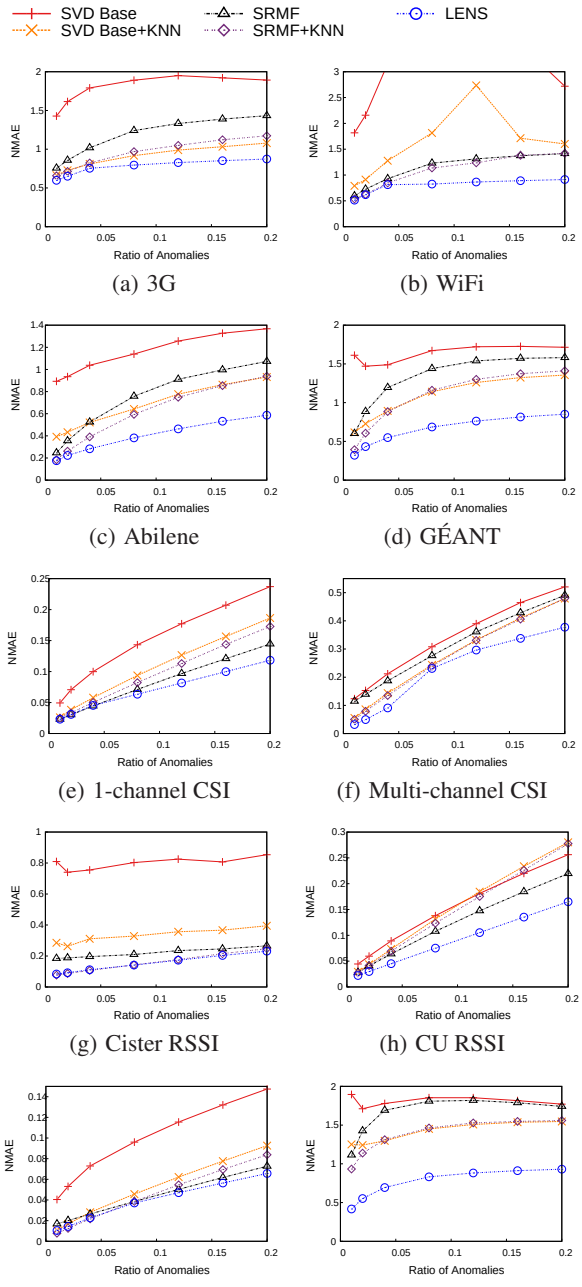
**Figure 7: Impact of anomaly sizes when ratio of anomalies = 5% and loss rate = 50%.**

Meshnet trace, 3.2% of EWMA prediction errors are larger than 5 times standard deviation from mean, whereas the corresponding numbers in other traces are 1.2%-2.4%. 3G trace has the second largest number of EWMA prediction error where we can also see LENS shows 7.7% improvement over SRMF+KNN.

**Varying anomaly sizes:** Figure 7 shows the interpolation performance as we vary the anomaly size $s$. LENS significantly outperforms all the other schemes. Its benefit increases with the anomaly size. For example, when $s = 1$, the NMAE of LENS is 33.7% lower than SRMF, 20.2% lower than SRMF+KNN, 61.8% lower than SVD Base, and 34.8% lower than SVD Base+KNN. The corresponding numbers under $s = 2$ are 44.9%, 31.9%, 69.8%, and 45.8%, respectively. Moreover, as we would expect, NMAE of all schemes tends to increase with the anomaly size in all traces, though the NMAE of LENS increases more slowly than the other schemes, since LENS explicitly separates anomalies before data interpolation. These results highlight the importance of anomaly detection in interpolation.

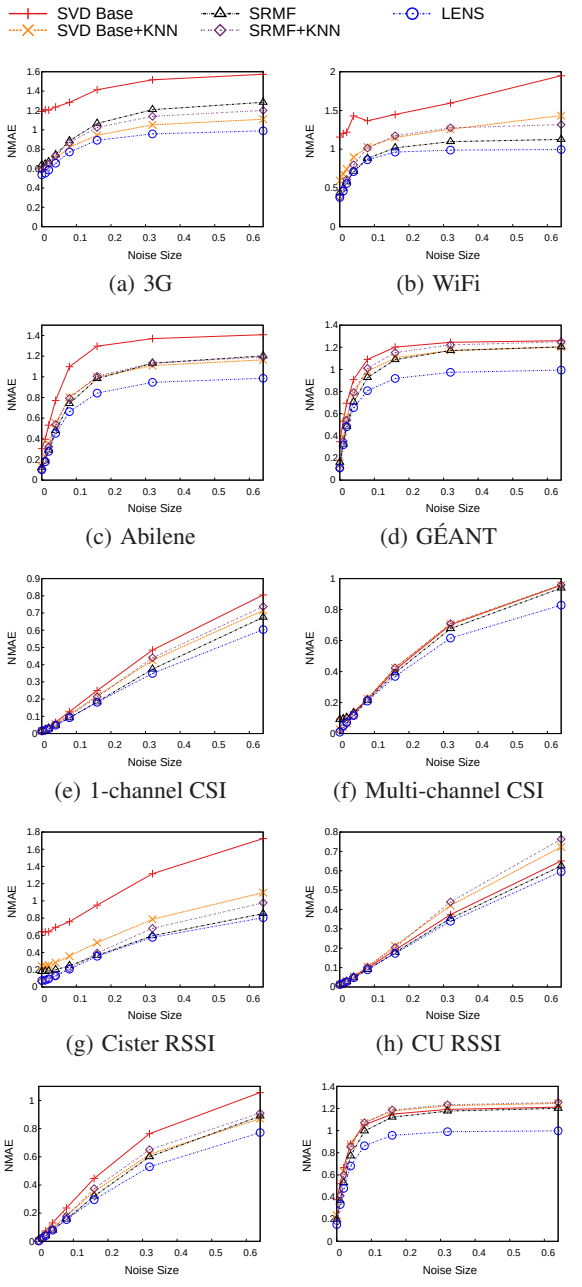
**Varying the number of anomalies:** Figure 8 shows the interpolation performance as we vary the number of anomalies. As before, LENS out-performs SRMF and SVD based schemes. The improvement ranges between 25.3-59.7% with 8% anomalies and 30.1-54.5% with 16% anomalies. In addition, the NMAE increases with the number of anomalies. Among different schemes, the rate of increase is slowest in LENS due to its explicit anomaly detection and removal.

**Varying noise sizes:** Figure 9 shows the interpolation performance as we vary the noise sizes. We inject noise to all the elements in the original matrices. The size of the noise follows normal distribution with mean 0 and standard deviation $\sigma$ where $\sigma$ is varied from 1%




(a) 3G (b) WiFi (c) Abilene (d) GÉANT (e) 1-channel CSI (f) Multi-channel CSI (g) Cister RSSI (h) CU RSSI (i) UMich RSS (j) UCSB Meshnet

**Figure 8: Impact of number of anomalies when the loss rate = 50% and $s = 1$.**




(a) 3G (b) WiFi (c) Abilene (d) GÉANT (e) 1-channel CSI (f) Multi-channel CSI (g) Cister RSSI (h) CU RSSI (i) UMich RSS (j) UCSB Meshnet

**Figure 9: Impact of noise sizes when the loss rate = 50% and no anomaly.**

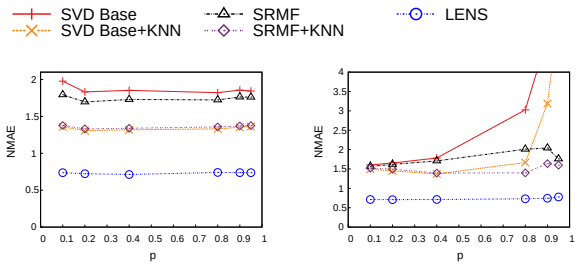
to 64% of the maximal value in the matrix. As before, LENS out-performs the other schemes.

**Different dropping modes:** Next we compare the interpolation accuracy under different dropping modes. In the interest of brevity, Figure 10 shows interpolation error for UCSB Meshnet traces. NMAE is similar for the other traces. As we can see, LENS yields lowest NMAE under all dropping modes. It out-performs SRMF-based schemes by 52.9%, and out-perform SVD-based schemes by 60.0%.

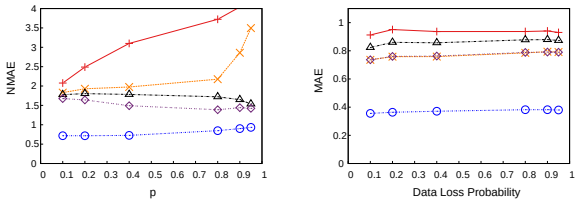
**Prediction:** Prediction is different from general interpolation because consecutive columns are missing. SVD is not applicable in this context. KNN does not work well either since temporally or spatially near neighbors have missing values. Figure 11 shows the prediction error as we vary the prediction length (*i.e.*, prediction length $l$ means that the first $1 - l$ columns are used to predict the remaining $l$ columns). We include Base in the figure since [50] shows Base is effective in prediction. LENS out-performs SRMF, which out-performs Base.

Figure 12 further compares Base, SRMF, and LENS as we vary anomaly size. LENS continues to out-perform SRMF and Base. On average, it improves SRMF by 17.7%, and improves Base by 30.4%. Figure 13 shows the performance as we vary the number of anomalies. LENS continues to perform the best, out-performing SRMF by 29.6% and Base by 34.6%.

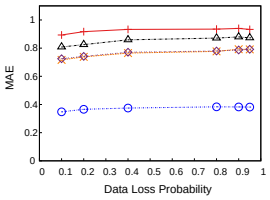
**Anomaly detection:** We further compare the accuracy of anomaly detection as we inject anomalies to 5% elements with $s = 1$. SRMF detects anomalies based on the difference between the actual and estimated values, and consider the entry has an anomaly if its difference is larger than a threshold. LENS considers all entries whose Y values are larger than a threshold as anomalies. Following [50], for each of the schemes, we choose a threshold to achieve




(a) xxTimeRandLoss. xx = 50. (b) xxElemRandLoss. xx = 50.

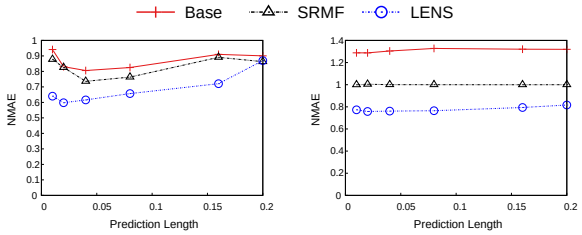



(c) xxElemSyncLoss. xx = 50. (d) RowRandLoss



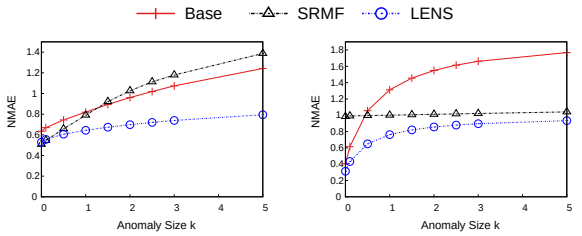

(e) ColRandLoss

**Figure 10: UCSB Meshnet: interpolation performance under various dropping models and 5% anomalies. $xx$ and $p$ in (a)-(c) are defined in Section 4.1**




(a) WiFi (b) UCSB Meshnet

**Figure 11: Prediction performance under 5% anomalies**

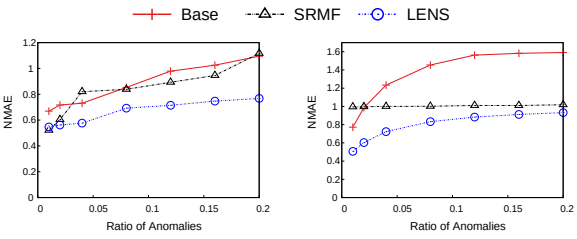



(a) WiFi (b) UCSB Meshnet

**Figure 12: Prediction performance with various anomaly sizes, 5% anomalies, and 10% prediction length.**

the false alarm probability within $10^{-5}$. As shown in Figure 14, LENS consistently out-performs SRMF+KNN. In 3G and Cister RSSI traces, its F1-score is 17.6% higher than that of SRMF+KNN. This shows that LENS is effective in anomaly detection.

**Computational time:** Figure 15 compares the computation time of LENS and SRMF when both use 500 iterations. As we can see, LENS has much smaller computation time due to local optimization in ADM. This makes it feasible to perform efficient search over different parameters. Figure 16 further shows that LENS converges within 200-250 iterations.




(a) WiFi (b) UCSB Meshnet

**Figure 13: Prediction performance with various number of anomalies when the prediction length = 10%**

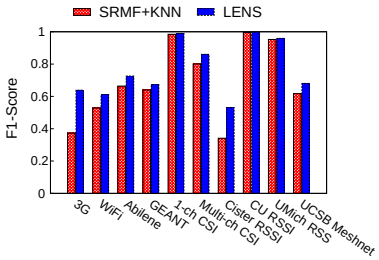
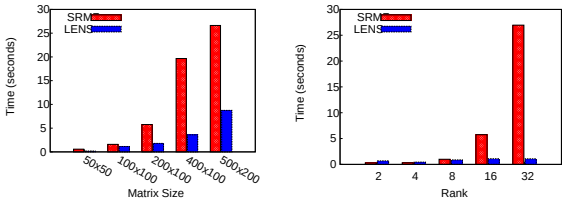



**Figure 14: Anomaly detection performance as we inject anomalies to 5% elements with $s = 1$.**




(a) vary matrix sizes (rank=10) (b) vary ranks ($100 \times 100$ matrices)

**Figure 15: Computation time**

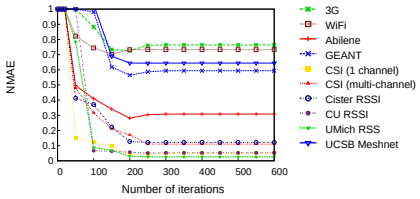



**Figure 16: The interpolation performance of LENS under various number of iterations when the loss rate = 50% and $s = 1$.**

## 5. RELATED WORK

**Compressive sensing:** LENS belongs to the realm of *compressive sensing*, a generic methodology for extracting and exploiting the presence of certain types of structure and redundancy in data from many real-world systems. Compressive sensing has recently attracted considerable attentions from statistics, approximation theory, information theory, and signal processing [8, 14, 7, 37, 38, 50] and is rapidly becoming a vibrant research area of its own.

Most existing compressive sensing works assume that the matrices satisfy low-rank property. However, this assumption may not hold in as we show in Section 2. Violation of such assumption significantly limits the accuracy of these techniques.

Significant work has been done for solving under-determined linear inverse problems. Missing value interpolation, prediction, and network tomography can be cast into the same formulation. As described in [48], many solutions solve the regularized least-squares

problem: $\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 J(\mathbf{x})$, where $\|\cdot\|_2$ denotes the $L_2$ norm, $\lambda$ is a regularization parameter, and $J(\mathbf{x})$ is a penalization functional. In $L_2$ norm minimization, which is a widely used solution to linear inference problem, $J(\mathbf{x}) = \|\mathbf{x}\|_2^2$. In $L_1$ norm minimization, another commonly used scheme, $J(\mathbf{x}) = \|\mathbf{x}\|_1$ (*i.e.*, the $L_1$ norm of $\mathbf{x}$). Other regularization terms include $\|X\|_*$, the nuclear norm of matrix $X$, and spatio-temporal terms $\|SX\|_F^2$ and $\|XT^T\|_F^2$ in [50]. Unique advantages of LENS formulation include (i) its general formulation to account for a low-rank component, a sparse anomaly component, a dense but small noise term, and domain knowledge, (ii) its effective optimization algorithm to solve the general decomposition problem, and (iii) a data-driven procedure to learn the parameters. Its formulation is more general than existing work (*e.g.*, [50]) in that [50] requires the original matrix to be well approximated by the product of two low-rank matrices, whereas LENS relaxes this constraint and allows the delta between the original matrix and sparse anomaly matrix to be approximated by the product of two low-rank matrices. Moreover, LENS allows a linear coefficient in each of the decomposed terms and supports general forms of domain knowledge.

**Anomaly detection:** Anomaly detection has been extensively studied. PCA (*e.g.*, [19, 22, 23]) has been widely used for anomaly detection. PCA has several well known limitations: it is sensitive to how many principal components are used [39] and vulnerable to data poisoning [40]. Barford et al. [5] uses wavelets to decompose an original signal into low-, mid-, and high-frequency components and use the high-frequency components for anomaly detection. [46] presents a framework that incorporates a variety of anomaly detectors. [50] uses SRMF for anomaly detection based on the difference between estimated and actual matrix values. [26] proposes a multi-scale robust subspace algorithm to identify changes in performance. NICE [29] uses statistical correlation to detect chronic network problems. Mercury [28] detects persistent behavior changes using the time-alignment for distributed triggers. [27] combines different anomaly detection methods, such as EWMA, FFT, Holt-Winters, and Wavelets to boost the performance. [17] uses both customer call dataset and network crash log in IPTV to detect anomalies. It first finds heavy hitters where there might be anomalies with high probability and then use EWMA to detect anomalies. We complement the above work by developing a general framework that explicitly accounts for anomalies during missing value interpolation to avoid contamination.

# 6. CONCLUSION

This paper presents *LENS decomposition* to decompose a network matrix into a low-rank matrix, a sparse anomaly matrix, an error matrix, and a dense but small noise matrix. Our evaluation shows that it can effectively perform missing value interpolation, prediction, and anomaly detection and out-perform state-of-the-art approaches. As part of our future work, we plan to apply our framework to network tomography (*e.g.*, traffic matrix estimation based on link loads and link performance estimation based on end-to-end performance). As part of our future work, we plan to apply LENS to enable several important wireless applications, including spectrum sensing, channel estimation, and localization.

# Acknowledgements

This work is supported in part by NSF Grants CCF-1117009, CNS-1343383, NSFC 61170237, and Singapore NRF CREATE E2S2.

# 7. REFERENCES

[1] Lagrange multiplier. http://en.wikipedia.org/wiki/Lagrange_multiplier.
[2] The Abilene Observatory Data Collections. http://abilene.internet2.edu/observatory/data-collections.html.
[3] LAN/MAN Standards Commmittee of the IEEE Computer Society. Part 11: Wireless LAN Medium Access Control and Physical Layer (PHY) Specifications . *IEEE Standard 802.11*, 2009. http://standards.ieee.org/getieee802/download/802.11n-2009.pdf.
[4] Augmented lagrangian method. http://en.wikipedia.org/wiki/Augmented_Lagrangian_method.
[5] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proc. of IMW*, 2002.
[6] K. Bauer, D. McCoy, D. Grunwald, and D. C. Sicker. CRAWDAD data set CU/RSSI (v. 2009-06-15). http://crawdad.cs.dartmouth.edu/cu/rssi/, May 2009.
[7] E. Candes and B. Recht. Exact matrix completion via convex optimization.
[8] E. Candes and T. Tao. Near optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans. on Information Theory*, 52(12):5406–5425, 2006.
[9] E. J. Candes, X. Li, Y. Ma, and J. Wright. Robust principal component analysis?, 2009. Manuscript. Available from http://www-stat.stanford.edu/~candes/papers/RobustPCA.pdf.
[10] S. S. Chen, D. L. Donoho, Michael, and A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20:33–61, 1998.
[11] G. Cheng, H. Chen, D. Cheng, Z. Zhang, and J. Lan. Anomaly detections in internet traffic using empirical measures. *International Journal of Innovative Technology and Exploring Engineering*, Feb. 2013.
[12] Crawdad. http://crawdad.cs.dartmouth.edu.
[13] W. Deng, M.-J. Lai, Z. Peng, and W. Yin. Parallel multi-block ADMM with o(1/k) convergence. *UCLA CAM 13-64*, 2014.
[14] D. Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 52(4):1289–1306, 2006.
[15] B. George, S. Kim, and S. Shekhar. Spatio-temporal network databases and routing algorithms: A summary of results. In *Proceedings of the 10th International Conference on Advances in Spatial and Temporal Databases*, SSTD'07, pages 460–477, Berlin, Heidelberg, 2007. Springer-Verlag.
[16] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Predictable 802.11 packet delivery from wireless channel measurements. In *Proc. of SIGCOMM*, 2010.
[17] C.-Y. Hong, M. Caesar, N. Duffield, and J. Wang. Tiresias: Online anomaly detection for hierarchical operational network data. In *Proc. of ICDCS*, 2012.
[18] L. Huang, X. Nguyen, M. Garofalakis, and J. M. Hellerstein. Communication-efficient online detection of network-wide anomalies. In *Proc. of IEEE INFOCOM*, 2007.
[19] Y. Huang, N. Feamster, A. Lakhina, and J. J. Xu. Diagnosing network disruptions with network-wide analysis. In *Proc. of ACM SIGMETRICS*, 2007.
[20] 802.11n channel measurement tool. http://ils.intel-research.net/projects/80211n-channel-measurement-tool.
[21] J. Kowalski and B. Warfield. Modeling traffic demand between nodes in a telecommunications network. In *Proc. of ATNAC*, 1995.
[22] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proc. of ACM SIGCOMM*, 2004.
[23] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *Proc. of ACM SIGCOMM*, 2005.
[24] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft. Structural analysis of network traffic flows. In *Proc. of ACM SIGMETRICS*, 2004.
[25] L. F. Lu, Z.-H. Huang, M. A. Ambusaidi, and K.-X. Gou. A large-scale network data analysis via sparse and low rank reconstruction. *Discrete Dynamics in Nature and Society*, Feb. 2013.
[26] A. Mahimkar, Z. Ge, J. Wang, J. Yates, Y. Zhang, J. Emmons, B. Huntley, and M. Stockert. Rapid detection of maintenance induced changes in service performance. In *Proc. of ACM CoNEXT*, 2011.
[27] A. Mahimkar, A. Lall, J. Wang, J. Xu, J. Yates, and Q. Zhao. Anomaly detection in large operational networks. In *Proc. of ICDCS*, 2011.
[28] A. Mahimkar, H. H. Song, Z. Ge, A. Shaikh, J.Wang, J. Yates, Y. Zhang, and J. Emmons. Detecting the performance impact of upgrades in large operational networks. In *Proc. of ACM SIGCOMM*, 2010.
[29] A. Mahimkar, J. Yates, Y. Zhang, A. Shaikh, J. Wang, Z. Ge, and C. T. Ee. Troubleshooting chronic conditions in large IP networks. In *Proc. of ACM CoNEXT*, 2008.
[30] Y. Mao and L. K. Saul. Modeling distances in large-scale networks by matrix factorization. In *Proc. of IMC*, New York, NY, USA, 2004.
[31] S. Nagaraja, V. Jalaparti, M. Caesar, and N. Borisov. P3ca: Private anomaly detection across ISP networks. In *Proc. of PETS*, volume 6794. Springer, 2011.
[32] A. Navaz, S.Gopalakrishnan, and R.Meena. Anomaly detections in internet traffic using empirical measures. *International Journal of Innovative Technology and Exploring Engineering*, Feb. 2013.
[33] D. Needell. Non-asymptotic theory of random matrices. Lecture 6: Norm of a random matrix, 2007. Stanford Math 280 Lecture Notes. Available at http://www-stat.stanford.edu/~dneedell/lecs/lec6.pdf.
[34] T. E. Ng and H. Zhang. Predicting internet network distance with coordinate-based appoaches. In *Proc. of INFOCOMM*, 2002.
[35] C. Noda, S. Prabh, M. Alves, T. Voigt, and C. A. Boano. CRAWDAD data set Cister/RSSI (v. 2012-05-17). http://crawdad.cs.dartmouth.edu/cister/rssi/, 2005.
[36] S. Rallapalli, L. Qiu, Y. Zhang, and Y.-C. Chen. Exploiting temporal stability and low-rank structure for localization in mobile networks. In *Proc. of ACM MobiCom*, 2010.
[37] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum rank solutions to linear matrix equations via nuclear norm minimization. preprint.

[38] B. Recht, W. Xu, and B. Hassibi. Necessary and sufficient condtions for success of the nuclear norm heuristic for rank minimization. In *47th IEEE Conference on Decision and Control*, December 2008.
[39] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of PCA for traffic anomaly detection. In *Proc. of ACM SIGMETRICS*, 2007.
[40] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S. h. Lau, S. Rao, N. Taft, and J. D. Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proc. of ACM IMC*, 2009.
[41] UCSB meshnet. `http://www.crawdad.org/ucsb/meshnet/`.
[42] S. Uhlig, B. Quoitin, S. Balon, and J. Lepropre. Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM Computer Communication Review*, 36(1):83–86, January 2006.
[43] Umich RSS. `http://crawdad.cs.dartmouth.edu/umich/rss/`.
[44] M. Wang, A. Ailamaki, and C. Faloutsos. Capturing the spatio-temporal behavior of real traffic data. *Perform. Eval.*, 49(1-4):147–163, Sept. 2002.
[45] Wikipedia. F1 score. http://en.wikipedia.org/wiki/F1_score.
[46] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *Proc. of ACM IMC*, 2005.
[47] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *Proceedings of ACM SIGMETRICS*, June 2003.
[48] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proc. of ACM SIGCOMM*, Aug. 2003.
[49] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. Estimating point-to-point and point-to-multipoint traffic matrices: An information-theoretic approach. *ACM/IEEE Transactions on Networking*, 13(5):947–960, Oct. 2005.
[50] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu. Spatio-temporal compressive sensing and Internet traffic matrices. In *Proceedings of ACM SIGCOMM*, 2009.

# APPENDIX

# A. DETAILS OF THE ALTERNATING DIRECTION METHOD

Each iteration of the Alternating Direction Method involves the following steps:

1. Find $X$ to minimize the augmented Lagrangian function $\mathcal{L}(X, \{X_k\}, Y, Y_0, Z, W, M, \{M_k\}, N, \mu)$ with other variables fixed. Removing the fixed terms, the objective is:

$$\text{minimize: } \alpha\|X\|_* + \frac{\mu}{2}\sum_{k=0}^{K}\|X_k + M_k/\mu - X\|_F^2.$$

Let $J = \frac{1}{K+1}\sum_{k=0}^{K}(X_k + M_k/\mu)$, and $t = \frac{\alpha/\mu}{K+1}$. We can simplify the objective to the following:

$$\text{minimize: } t\|X\|_* + 1/2\|X - J\|_F^2.$$

According to matrix completion literature, this is a standard nuclear norm minimization problem and can be solved by applying soft thresholding on the singular values of $J$. Specifically, we have:

$$X = \mathsf{SVSoftThresh}(J, t).$$

For a given $J$ and $t$, let $J = USV^T$ be the singular value decomposition of $J$. We have: $\mathsf{SVSoftThresh}(J, t) \triangleq U\mathsf{SoftThresh}(S, t)V^T$, where

$$\mathsf{SoftThresh}(S[i, j], t) \triangleq \mathsf{sign}(S[i, j])\max(0, |S[i, j]| - t).$$

2. Find $X_k$ to minimize $\mathcal{L}(X, \{X_k\}, Y, Y_0, Z, W, M, \{M_k\}, N, \mu)$ with other variables fixed ($k = 1, 2, ..., K$). This gives:

$$\text{minimize: } \frac{\gamma}{2\sigma}\|P_k X_k Q_k^T - R_k\|_F^2 + \frac{\mu}{2}\|X_k + M_k/\mu - X\|_F^2.$$

This is a least square problem with respect to $X_k$. The optimal solution can be obtained by forcing the gradient of the objective to be zero. That is,

$$\frac{\gamma}{\sigma}P_k^T(P_k X_k Q_k^T - R_k)Q_k^T + \mu(X_k + M_k/\mu - X) = 0. \quad (20)$$

Let $J = X - M_k/\mu$, and $R = P_k^T R_k Q_k + \frac{\mu\sigma}{\gamma}J$. Eq. (20) simplifies to

$$P_k^T P_k X_k Q_k^T Q_k + \frac{\mu\sigma}{\gamma}X_k = R. \quad (21)$$

Perform eigendecomposition on $P_k^T P_k$ and $Q_k^T Q_k$ and let $USU^T = P_k^T P_k$; $VTV^T = Q_k^T Q_k$, where $U$ and $V$ are orthogonal matrices, $S$ and $T$ are diagonal matrices. We have: $S(U^T X_k V)T + \frac{\mu\sigma}{\gamma}(U^T X_k V) = U^T RV$. Through a change of variable, let $H = U^T X_k V$, Eq. (21) becomes:

$$SHT + \frac{\mu\sigma}{\gamma}H = U^T RV. \quad (22)$$

Let $\mathbf{s} = diag(S), \mathbf{t} = diag(T)$ be the diagonal vector of $S$ and $T$, respectively. Eq. (22) is equivalent to $(\mathbf{s}\mathbf{t}^T + \frac{\mu\sigma}{\gamma}). * H = U^T RV$. Since $U$ and $V$ are orthogonal matrices, we can easily find $X_k$ from $H$ as $X_k = UHV^T$. So we have: $H = (U^T RV)./(\mathbf{s}\mathbf{t}^T + \frac{\mu\sigma}{\gamma})$, where $./$ is an operator for element-wise division. Thus,

$$X_k = UHV^T = U\left((U^T RV)./(\mathbf{s}\mathbf{t}^T + \frac{\mu\sigma}{\gamma})\right)V^T.$$

3. Find $X_0$ to minimize $\mathcal{L}(X, \{X_k\}, Y, Y_0, Z, W, M, \{M_k\}, N, \mu)$ with other variables fixed ($k = 1, 2, ..., K$). This gives:

$$\begin{aligned}\text{minimize: } & \langle M, D - AX_0 - BY_0 - CZ - W\rangle \\ & +\langle M_0, X_0 - X\rangle \\ & +\mu/2\|D - AX_0 - BY_0 - CZ - W\|_F^2 \\ & +\mu/2\|X_0 - X\|_F^2\end{aligned}$$

That is,

$$\begin{aligned}\text{minimize: } & \|X_0 - X + M_0/\mu\|_F^2 \\ & +\|D - BY_0 - CZ - W + M/\mu - AX_0\|_F^2\end{aligned}$$

Let $J_0 = X - M_0/\mu$ and $J = D - BY_0 - CZ - W + M/\mu$. It becomes:

$$\text{minimize:} \|X_0 - J_0\|_F^2 + \|AX_0 - J\|_F^2$$

Letting the gradient be zero leads to: $X_0 - J_0 + A^T(AX_0 - J) = 0$. Therefore, $X_0 = \text{inv}(A^T A + I)(A^T J + J_0)$.

4. Find $Y$ to minimize $\mathcal{L}(X, \{X_k\}, Y, Y_0, Z, W, M, \{M_k\}, N, \mu)$ with other variables fixed. This gives:

$$\text{minimize:} \beta\|Y\|_1 + \langle N, Y_0 - Y\rangle + \mu/2\|Y_0 - Y\|_F^2$$

That is:

$$\text{minimize:} \beta/\mu\|Y\|_1 + 1/2\|Y_0 + N/\mu - Y\|_F^2.$$

Let $J = Y_0 + N/\mu$. $t = \beta/\mu$. It becomes: $t\|Y\|_1 + 1/2\|J - Y\|_F^2$. This can be easily solved as $Y = \mathsf{SoftThresh}(J, t)$. To see why, the problem can be solved for each element of $Y$ separately. So we just need to find $Y[i, j]$ that minimizes: $t|Y[i, j]| + 1/2(J[i, j] - Y[i, j])^2$.

5. Find $Y_0$ to minimize $\mathcal{L}(X, \{X_k\}, Y, Y_0, Z, W, M, \{M_k\}, N, \mu)$ with other variables fixed. This gives:

$$\begin{aligned}\text{minimize: } & \langle M, D - AX_0 - BY_0 - CZ - W\rangle \\ & +\langle N, Y_0 - Y\rangle \\ & +\mu/2\|D - AX_0 - BY_0 - CZ - W\|_F^2 \\ & +\mu/2\|Y_0 - Y\|_F^2\end{aligned}$$

Let $J_0 = Y - N/\mu$, $J = D - AX_0 - CZ - W + M/\mu$. It becomes minimize: $\|Y_0 - J_0\|_F^2 + \|BY_0 - J\|_F^2$. Letting the gradient = 0, we obtain: $Y_0 - J_0 + B^T(BY_0 - J) = 0$. So $Y_0 = \text{inv}(B^T B + I)(B^T J + J_0)$.

6. Find $Z$ to minimize $\mathcal{L}(X, \{X_k\}, Y, Y_0, Z, W, M, \{M_k\}, N, \mu)$ with other variables fixed. This gives:

$$\begin{aligned}\text{minimize: } & \frac{1}{2\sigma}\|Z\|_F^2 + \langle M, D - AX_0 - BY_0 - CZ - W\rangle \\ & +\frac{\mu}{2}\|D - AX_0 - BY_0 - CZ - W\|_F^2.\end{aligned}$$

Let $J = D - AX_0 - BY_0 - W + M/\mu$, it becomes:

$$\frac{1}{2\mu\sigma}\|Z\|_F^2 + \frac{1}{2}\|CZ - J\|_F^2.$$

Letting the gradient = 0 yields: $Z = \text{inv}(\frac{1}{\mu\sigma}I + C^T C)(C^T J)$.

7. Find $W$ to minimize $\mathcal{L}(X, \{X_k\}, Y, Y_0, Z, W, M, \{M_k\}, N, \mu)$ with other variables fixed. This gives:

$$\begin{aligned}\text{minimize: } & \langle M, D - AX_0 - BY_0 - CZ - W\rangle \\ & +\mu/2\|D - AX_0 - BY_0 - CZ - W\|_F^2.\end{aligned}$$

That is:

$$\text{minimize: } \|D - AX_0 - BY_0 - CZ - W + M/\mu\|_F^2$$

So $W = E. * (D - AX_0 - BY_0 - CZ + M/\mu)$ (recall that $W = E. * W$).

8. Update estimate for $\sigma_D$ as follows. Let $J = D - AX_0 - BY_0 - W$. We then compute $\sigma_D$ as the standard deviation of $J[E = 0]$ and update $\sigma = \theta\sigma_D$. In our implementation, we fix $\theta = 10$.
9. Update estimates for the Lagrangian multipliers $M$, $M_k$ and $N$ according to: $M = M + \mu\cdot(D - AX_0 - BY_0 - CZ - W)$, $M_k = M_k + \mu\cdot(X_k - X)$ ($k = 0, \cdots, K$), $N = N + \mu\cdot(Y_0 - Y)$.
10. Update $\mu = \mu\cdot\rho$. In our implementation, initially $\mu = 1.01$ and $\rho = 1.01$. Every 100 iterations, we multiply $\rho$ by 1.05.