Spatio-Temporal Compressive Sensing and Internet Traffic Matrices

Yin Zhang* Matthew Roughan‡ Walter Willinger§ Lili Qiu*
*The University of Texas at Austin ‡University of Adelaide §AT&T Labs – Research

ABSTRACT

Many basic network engineering tasks (*e.g.*, traffic engineering, capacity planning, anomaly detection) rely heavily on the availability and accuracy of traffic matrices. However, in practice it is challenging to reliably measure traffic matrices. Missing values are common. This observation brings us into the realm of *compressive sensing*, a generic technique for dealing with missing values that exploits the presence of structure and redundancy in many real-world systems. Despite much recent progress made in compressive sensing, existing compressive-sensing solutions often perform poorly for traffic matrix interpolation, because real traffic matrices rarely satisfy the technical conditions required for these solutions.

To address this problem, we develop a novel spatio-temporal compressive sensing framework with two key components: (i) a new technique called SPARSITY REGULARIZED MATRIX FACTORIZATION (SRMF) that leverages the sparse or low-rank nature of real-world traffic matrices and their spatio-temporal properties, and (ii) a mechanism for combining low-rank approximations with local interpolation procedures. We illustrate our new framework and demonstrate its superior performance in problems involving interpolation with real traffic matrices where we can successfully replace up to 98% of the values. Evaluation in applications such as network tomography, traffic prediction, and anomaly detection confirms the flexibility and effectiveness of our approach.

Categories and Subject Descriptors

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

General Terms

Measurement, Performance

Keywords

Compressive Sensing, Traffic Matrix, Interpolation, Tomography, Prediction, Anomaly Detection

1. INTRODUCTION

Traffic Matrices (TMs), which specify the traffic volumes between origin and destination pairs in a network, are critical inputs to many network engineering tasks, such as traffic engineering [11,24], capacity planning, and anomaly detection. Due to their importance, there is now a substantial body of work on TMs, for instances see [2] and the references therein. The thrust of much of this research has been on measurement [10,28] and inference [9,17, 25,27,31–34] of TMs, and more recently on topics such as anomaly detection [13, 14, 21, 29, 30]. A key challenge that lies at the heart of many of these problems is how to cope with missing values that frequently arise in real-world TMs. In this paper, we propose novel interpolation techniques to accurately reconstruct missing values in TMs based on partial and/or indirect measurements. In the process, we provide a unified approach to several common tasks involving measurement and analysis of traffic matrices; *e.g.*, TM estimation, prediction, and anomaly detection. Our approach uses the first truly spatio-temporal model of TMs, borrows ideas from the active area of compressive sensing, and exploits domain knowledge regarding TMs that has accumulated over the years.

Motivation: In practice it is challenging to reliably measure TMs for large networks. First, in many networks the TM is not directly observable, and can only be estimated through link load measurements. Such measurements, while linearly related to the TM itself, are not sufficient to unambiguously identify the true TM. Typically, the problem was posed as an underconstrained linear-inverse problem, where the solution relied on a prior model of the TM (*e.g.*, the Poisson model of Vardi [27], the gravity model [31, 33], or the independent flow model [9]). Second, although many networks now collect (sampled) flow-level measurements for at least part of their network, there are still serious impediments to *reliable* large-scale collection of TMs: data collection systems can fail, flow collectors often use an unreliable transport protocol, and legacy network components may not support flow collection or be resource challenged. Third, scalability requirements may mean that flow-level collection doesn't occur at the edge of a network (where we would wish it for true TM recovery [10]), but often only on some subset of the routers. Recovery of the actual ingress-egress TM from such data is non-trivial. Finally, when we find an anomaly in a set of TMs, we often need to know the non-anomaly-related traffic either for other network tasks, or just so that we can infer the cause of the anomaly. The result is that any large set of TM measurements has some, and quite often, a significant number of missing values.

Since many network engineering tasks that require TMs are either intolerant or highly sensitive to missing data, it is important to accurately reconstruct missing values based on partial and/or indirect TM measurements. *Interpolation* is the mathematical term for filling in these missing values. *Compressive sensing* is a generic methodology for dealing with missing values that leverages the presence of certain types of structure and redundancy in data from many real-world systems. Compressive sensing has recently attracted considerable attention in statistics, approximation theory, information theory, and signal processing. Several effective heuristics have been proposed to exploit the sparse or low-rank nature of

SIGCOMM'09, August 17–21, 2009, Barcelona, Spain.

data [5, 6, 8, 19, 20]. Meanwhile, the mathematical theory of compressive sensing has also advanced to the point where the optimality of many of these heuristics has been proven under certain technical conditions on the matrices of interest.

Contributions: Despite much recent progress in the area of compressive sensing, our extensive evaluation of the existing compressive sensing algorithms on real TMs shows that they do not perform well for TM interpolation, especially under structured, high data loss (see Section 4). The main reason is that real TMs often exhibit characteristics that violate the mathematical conditions under which existing compressive sensing algorithms are designed to operate and are provably optimal. Specifically, the optimality results for existing compressive sensing algorithms often assume that (i) the matrix elements are drawn from a Gaussian or Gaussian-like distribution, (ii) the matrix is exactly low-rank, (iii) data loss is independent for different matrix elements, and (iv) the measurement constraints on the matrix satisfy a certain technical conditions (*e.g.*, the restricted isometry property [19]). Unfortunately, none of these conditions are likely to hold for real TMs. Real TM elements often exhibit a highly skewed distribution, where the largest and smallest elements often differ in size by several orders of magnitude. Moreover, real TMs are only approximately low-rank, and data loss in real TMs tends to be highly structured — data may be missing either spatially (we may be missing entire rows or columns of the TM), or temporally (we may be missing matrix elements over entire segments in time), or in some combination. Finally, there is no guarantee that the constraints arising from real-world TM measurements satisfy the required technical condition.

To address the above challenge, we develop in this paper a novel spatio-temporal compressive sensing framework for TM interpolation. Our framework has two key components:

- We develop SPARSITY REGULARIZED MATRIX FACTORIZATION (SRMF), which finds sparse, low-rank approximations of TMs that account for spatial and temporal properties of real TMs. To the best of our knowledge, SRMF represents the first genuine spatio-temporal model of TMs. In contrast, most past approaches can be best described as purely spatial (*e.g.*, [9, 13, 14, 31, 33]) or temporal (*e.g.*, [3, 27]). SRMF is also quite general; it includes as special cases many of the existing techniques (*e.g.*, PCA, Tomo-gravity [31, 33] and independent flows [9]), but admits a much larger variety of algorithms.
- We augment low-rank approximations of TMs with local interpolation. In this way, we can leverage additional local structure and redundancy that are difficult to capture using strictly low-rank approximations of a TM. For example, there may not exist any strictly low-rank approximation that can satisfy all the linear constraints between the link load measurements and the original TM. Similarly, a strictly low-rank global structure may be too inflexible to capture the local similarity between individual TM elements. Our strategy is to use the low-rank approximation obtained by SRMF as a prior and derive a refined approximation that is no longer strictly low-rank but is close to the low-rank prior and can also account for existing local structure and redundancy.

We use real TMs from three operational networks to evaluate the effectiveness of our approach. Our most successful algorithm, SPARSITY REGULARIZED MATRIX FACTORIZATION combined with local interpolation, has many desirable properties. Its performance when applied to real TMs is excellent. We can reconstruct TMs with up to half their data missing with errors of the order of 10%, and even when 98% of the data points are missing our approach only has an error of the order of 30%. While it may be surprising that such good reconstructions are possible with so little data, our results are an indication of the degree of structure present in real-world TMs. The fact that we can perform such reconstructions could change the way TMs are collected. Much as sampling has enabled network-wide flow collection, reconstructions of this type can enable truly large-scale collections of TM data.

The technique has been applied to matrices with over 700 thousand entries, and we can process these in only a few seconds. The algorithm scales linearly with the size of the data so that much larger datasets can be analyzed. Moreover, tests of the proposed approach in applications such as network tomography, traffic prediction, and anomaly detection all confirm its effectiveness and robustness to real-world measurement issues.

Paper organization: The remainder of the paper is organized as follows. We provide background on traffic matrices and compressive sensing in Section 2. We describe our spatio-temporal compressive sensing framework in Section 3. We present evaluation results for TM interpolation in Section 4 and for applications of TM interpolation in Section 5. We conclude in Section 6.

2. BACKGROUND

2.1 Traffic Matrices

A Traffic Matrix (TM) is a non-negative matrix $Z(i, j)$ that describes volumes of traffic (in bytes, packets, or flows) between a source i and a destination j. For a network with N locations the TM is a square $N \times N$ matrix. In practice we need a number of addenda to this simple definition. First, a TM is typically measured over some time interval, and the value reported is an average. So we denote $Z(i, j; t)$ to be the traffic from i to j averaged over $[t, t + \Delta t)$. We call the TM $Z(*, *, t)$ a *snapshot* despite the fact that it really represents an interval. Second, although it is common to speak of "origin-destination" TMs, it is often difficult to accurately map IP addresses present in traffic to the true origin and destination of traffic when we examine a network or Autonomous System (AS). So typically the matrix is aggregated into a router-level *ingress-egress* TM, where $Z(i, j; t)$ describes the traffic entering a network at router i, and leaving at router j.

The TM may be thought of as a 3-dimensional array $Z \in \mathbb{R}^N \times \mathbb{R}^N \times \mathbb{R}^m$ (where there are m time intervals present). It is common to take a TM snapshot and stack the columns to form a column vector which we denote $\mathbf{x}_t$. We can compile these vectors into the columns of a larger matrix $X \in \mathbb{R}^n \times \mathbb{R}^m$ (where $n = N^2$), and this form of the TM is often more convenient for algebraic manipulation than a 3-dimensional array. Note that the columns of X represent the TM at different times, while the rows represent the time evolution of a single element of the TM.

One example of how this notation is useful is in TM inference (the so-called network tomography problem [27]). In this problem the TM is related to the more easily measured link loads Y by the following linear matrix equation

$$Y = AX, \tag{1}$$

where A is the routing matrix, which expresses which links are used by which routes[1]. TM inference involves finding the "best" solution $\hat{X}$ to (1) given a set of link-load measurements Y.

More generally, we can combine link measurements with additional TM measurement strategies, which often yields a better estimate of the TM than using each individual type of measurements by itself [34]. For example, flow-records are typically collected at ingress routers [10]. In this case, each router sees one row of a TM snapshot, so over time, router i sees $Z(i, *, *)$. Missing data

[1]Typically issues such as changing network topology or routing, or number of routers are ignored in the mathematical literature, but such issues have been successfully dealt with in practical instantiations of network tomography algorithms [30, 33].

from a single router means we will be missing a row of Z, or a group of rows of X. Flow-records could also be collected at egress or backbone routers. In this case, although it is difficult to unambiguously determine the ingress router for the observed traffic, we can still form a set of linear constraints on where the traffic could have originated. An alternative measurement strategy [28, 33] is to collect local TMs at each router, which can again be represented as linear constraints on the global TM. In combination we have a set of linear constraints on the TM, *i.e.*,

$$\mathcal{A}(X) = B, \tag{2}$$

where $\mathcal{A}(\cdot)$ is a linear operator, and the matrix B contains the measurements. The operator expresses the information available in our measurements. Note that the presence of missing data is implicit in (2); for instance, the operator $\mathcal{A}$ could include TM measurements at ingress routers with no measurement errors (but with missing data), by writing (2) as

$$M. * X = M. * D, \tag{3}$$

where $D(i,j)$ contains the direct measurements (where available) and M is a $N^2 \times m$ matrix given by

$$M(i,j) = \begin{cases} 0, & \text{if } X(i,j) \text{ is missing.} \\ 1, & \text{otherwise.} \end{cases} \tag{4}$$

and $.*$ denotes an element-wise product, *i.e.*, $A = C. * B$ means $A(i,j) = B(i,j)C(i,j)$. When both link measurements and direct measurements are available, then constraints (3) will (typically) be incorporated into (2) to simplify notation.

In addition to the above concerns we note that all data sources contain errors. Flow-level collection usually involves sampling, often at quite high rates, and the Simple Network Management Protocol (SNMP) used for collecting link measurements is often noisy.

We seek an estimated TM $\hat{X}$ that satisfies the conditions imposed by the set of measurements. However, as is the case in many such linear-inverse problems, there may not be enough information to unambiguously determine X. We call these *underconstrained* linear-inverse problems (in the case of TM estimation from link data, the problem is very highly underconstrained). To solve such problems, we can use side information about the nature of the TM being considered, for instance the gravity model of [31, 33] or independent-flows model of [9]. Regularization is a process used to solve such problems in which we "regularize" towards some prior model of the data in question. The low-rank model we will propose here is motivated by the recent literature on compressive sensing.

2.2 Compressive Sensing

We have seen that interpolation is necessary because of missing values in the data we collect. In addition, we can set up missing data problems deliberately as part of the design of scalable measurement systems. As networks grow, it becomes more difficult to maintain the associated measurement infrastructure. Methods to reduce the required infrastructure have started to appear, the most common of which is sampling. A new idea in signal processing is that of compressive sensing [6, 8]. The main idea behind compressive sensing is that since many real-world signals or datasets exhibit some structure or redundancy (*i.e.*, they are not pure noise), one should be able to utilize this prior knowledge for both acquisition and reconstruction of the signal or dataset at hand.

Structure and redundancy in data are often synonymous with *sparsity*. A *sparse* vector is simply a vector that has only a few non-zero elements. Often our vectors of interest might have only a few large elements, and many small elements. We call such a vector *compressible*, in the sense that most of its information is carried in the larger elements. Note that the majority of work on compressive sensing has concerned vectors of data, so a naive approach to TMs might be to compile these into vectors and then apply vector techniques. However, some of the structure of a TM is inherent in the matrix itself, so there is value in treating our matrix X as a genuine matrix. In the context of matrices, low rank is analogous to sparsity, because the spectrum formed by the singular values of a low-rank matrix is sparse (see below). It is now well known that TMs may be approximated by matrices of low rank [13, 14], and so this concept fits well here. We explicitly use this type of sparsity as our approach to resolve the underconstrained nature of the measurement problems we face. In the following section we draw on the recent matrix compressive-sensing literature [5, 19, 20] to explain how such "sparsity regularized" algorithms can be constructed.

2.3 Singular Value Decomposition

A basic tool for creating low-rank matrix approximations is the Singular Value Decomposition (SVD). Simply stated, any $n \times m$ real matrix X can be decomposed into three matrices such that

$$X = U\Sigma V^T, \tag{5}$$

where V^T is the transpose of V, and U is a $n \times n$ unitary matrix (*i.e.*, $U^TU = UU^T = I$), and V is a $m \times m$ unitary matrix (*i.e.*, $V^TV = VV^T = I$), and Σ is a $n \times m$ diagonal matrix containing the singular values σ_i of X. Typically the singular values are arranged so that $\sigma_i \geq \sigma_{i+1}$. The rank of a matrix is the number of linearly independent rows or columns, which equals the number of non-zero singular values.

To understand the SVD's use in matrix approximations, consider the following interpretation of the SVD. The matrix Σ is diagonal, so the SVD of a matrix X can be rewritten as:

$$X = U\Sigma V^T = \sum_{i=1}^{\min(n,m)} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sum_{i=1}^{\min(n,m)} \sigma_i A_i, \tag{6}$$

where $\mathbf{u}_i$ and $\mathbf{v}_i$ are the ith columns of U and V respectively, and the matrices A_i are (by construction) rank-1. We can then create a rank-r approximation $\tilde{X}$ from the SVD by keeping only the r largest singular values in the summation and dropping the others:

$$\tilde{X} = \sum_{i=1}^{r} \sigma_i A_i. \tag{7}$$

The above $\tilde{X}$ is known to be the best rank-r approximation with respect to the Frobenius norm $||\cdot||_F$ of the approximation errors, where $||Z||_F \triangleq \sqrt{\sum_{i,j} Z(i,j)^2}$ for any matrix Z. That is, truncation of the SVD provides the natural solution to:

$$\begin{array}{ll} \text{minimize} & ||X - \tilde{X}||_F, \\ \text{subject to} & \text{rank}(\tilde{X}) \leq r. \end{array} \tag{8}$$

In Internet measurement, the SVD has most commonly appeared in the form of the Principal Components Analysis (PCA), which has been used, for instance in anomaly detection [13, 14]. PCA is directly related to SVD by the fact that the columns of U form the principal axes of the PCA coordinate transform.

Sparsity Regularized SVD: Many matrix interpolation techniques try to create a SVD-like factorization of a matrix X, though it is common to write this in the equivalent form

$$X = U\Sigma V^T = LR^T, \tag{9}$$

where $L = U\Sigma^{1/2}$ and $R = V\Sigma^{1/2}$, and so we will subsequently use this form for consistency.

Now SVD by itself is not an interpolation algorithm. Typical algorithms for calculating the SVD assume that X is completely known. Instead we look for a factorization that satisfies the measurement equations, *i.e.*, $\mathcal{A}(LR^T) = B$. Previous studies have

suggested that typical TMs inhabit a relatively low-dimensional subspace [13, 14], so an intuitive approach for finding such a factorization is to solve the following rank minimization problem:

$$\begin{array}{ll} \text{minimize} & \text{rank}(LR^T), \\ \text{subject to} & \mathcal{A}(LR^T) = B. \end{array} \tag{10}$$

Rank minimization has a non-convex objective and is thus difficult to solve. A key insight from the matrix compressive-sensing literature [5, 19, 20] is that under certain conditions, we can solve a simpler problem instead and obtain equivalent results. Specifically, when a certain technical condition (the restricted isometry property [19]) holds on $\mathcal{A}(\cdot)$, then a heuristic — minimizing the nuclear norm — can perform rank minimization exactly for a matrix of low enough rank. Further, if the rank of X is less than the rank of LR^T then (10) is equivalent to

$$\begin{array}{ll} \text{minimize} & ||L||_F^2 + ||R||_F^2, \\ \text{subject to} & \mathcal{A}(LR^T) = B. \end{array} \tag{11}$$

In TM interpolation, looking for a low-rank solution that strictly satisfies the measurement equations is likely to fail, because (i) the real TM X is often only approximately low-rank, and (ii) the measurements often contain errors. So instead we solve the following

$$\text{minimize } ||\mathcal{A}(LR^T) - B||_F^2 + \lambda \left(||L||_F^2 + ||R||_F^2 \right). \tag{12}$$

This solution regularizes towards the low-rank approximation but does not strictly enforce the measurement equations. The regularization parameter λ allows a tunable tradeoff between a precise fit to the measured data and the goal of achieving low rank.

We derive L and R from (12) using an alternating least squares procedure. We initialize L and R randomly. We then solve the above optimization taking one of L and R to be fixed and the other to be the optimization variable (which is a standard linear least squares problem). We then swap their roles, and continue alternating towards a solution till convergence. Our implementation of the alternating least squares procedure in Matlab further utilizes sparse matrix operations to minimize memory requirement and maximize speed (details are omitted due to space restriction, but we will supply Matlab code on request). The above approach will be referred to as *Sparsity Regularized SVD (SRSVD)* interpolation.

2.4 Other Interpolation Algorithms

There are a number of approaches that have been proposed in the recent literature for matrix interpolation besides SVD. These algorithms can be classified as either low-rank approximation algorithms or local interpolation algorithms, depending on whether they exploit the global low-rank structure or the local structure and redundancy. We describe them here for completeness and for comparison with our approach detailed in Section 3.

2.4.1 Low-Rank Approximation Algorithms

Baseline Approximation: Many techniques (for instance PCA) implicitly assume that the data has zero mean. So our first step for dealing with network matrices might be to "center" them. However, centering the matrices where we do not have all the data also requires interpolation. Our baseline approximation algorithm implicitly constructs such an interpolation matrix X_{base} to compute row and column means of the matrix. For instance, if we knew all elements of the input X, then the row and column sums of $X - X_{\text{base}}$ would all equal zero. Apart from its use in zeroing the mean, it also forms an interpolation in its own right, and so we will compare its performance below.

To compute X_{base}, we use the variables described in Table 1. In matrix form, we can represent X_{base} as a rank-2 approximation to X: $X_{\text{base}} = \bar{X} + X_{\text{row}}\mathbf{1}^T + \mathbf{1}X_{\text{col}}^T$, where $\mathbf{1}$ is a column vector consisting of all ones. We use the regularized least-squares algorithm from [4] to compute $\bar{X}$, X_{row}, X_{col} from input $\mathcal{A}(\cdot)$ and B. That is, we solve the following

variable	description
$\bar{X}$	an estimate of the mean of X over all i and j.
X_{row}	a vector of length m such that $X_{\text{row}}(i) =$ an estimate of $\sum_j (X(i,j) - \bar{X})/n$.
X_{col}	a vector of length n such that $X_{\text{col}}(j) =$ an estimate of $\sum_i (X(i,j) - \bar{X})/m$.
X_{base}	the baseline estimate for X given by $X_{\text{base}}(i,j) = \bar{X} + X_{\text{row}}(i) + X_{\text{col}}(j)$.

Table 1: Outputs of baseline estimation.

$$\begin{array}{ll} \text{minimize} & ||\mathcal{A}(\bar{X} + X_{\text{row}}\mathbf{1}^T + \mathbf{1}X_{\text{col}}^T) - B||_F^2 \\ & + \lambda \left(\bar{X}^2 + ||X_{\text{row}}||_F^2 + ||X_{\text{col}}||_F^2 \right), \end{array} \tag{13}$$

where λ is a regularization parameter. The first term in this formulation minimizes the Frobenius norm of the difference $\mathcal{A}(X_{\text{base}}) - B$, and the second regularization term helps avoid overfitting.

SRSVD-base: Techniques like PCA implicitly assume that the data has zero mean, but in TM interpolation we do not know the true mean. Instead we use X_{base} as an estimate. It is not obvious whether such centering is necessary or desirable in interpolation, so we include results for both SRSVD applied to X and SRSVD applied to $(X - X_{\text{base}})$. We refer to the latter as SRSVD-base.

Nonnegative Matrix Factorization: Nonnegative Matrix Factorization (NMF) [15, 16] tries to find nonnegative factor matrices L and R that minimize the Frobenius norm[2] of the difference $\mathcal{A}(LR^T) - B$ over the observations. The approach is very similar to the SVD, except for the insistence on non-negative factor matrices. We avoid overfitting by regularizing in the same manner that we do for SVD, *i.e.*, we solve (12) but with the additional constraint of non-negativity. We implement the two most common algorithms for NMF: multiplicative update [15] and alternating nonnegative least squares. Both algorithms are designed for the case where the matrix X is completely known. So we extend them to further cope with measurement equations (2). The two algorithms give similar interpolation performance, but multiplicative update is more efficient. So our results are based on this algorithm.

2.4.2 Local Interpolation Algorithms

K-Nearest Neighbors: We also test one completely different approach: k-Nearest Neighbors (KNN). Simple nearest neighbors directly uses the nearest neighbor of a missing value for interpolation. KNN extends this by using a weighted average of the k nearest-neighbors' values. For TMs, it is more difficult to apply KNN because the rows are ordered arbitrarily (for instance based on the names of routers). So the nearest elements in the matrix X may have little correspondence. Hence we need to derive a good distance metric between matrix elements.

We use the approach described in [4]. We can perform the algorithm on either rows or columns of X, but let us start with rows. If two rows are similar (*i.e.*, two TM elements exhibit similar behavior), then it is natural to assume that one might provide a good interpolant of the other. Hence, we base our distance metric on the similarity between rows, *i.e.*, the more similar two rows are, the closer together we consider them. Following [4], we measure the similarity by an approximation to the correlation coefficient of the two rows based on only those directly observed TM elements. To

[2]There is nothing intrinsically special about the Frobenius norm for this approach. The Kullback-Leibler divergence [15] has also been suggested but our experiments found that the performance of this approach was much worse, and it is not presented here.

form this coefficient, we would ideally first subtract the mean, but as the mean is unknown we use our proxy X_{base}. The weights used in the k averaged neighbors are proportional to the similarities [4].

3. OUR SOLUTION: SPATIO-TEMPORAL COMPRESSIVE SENSING

The KNN approach is intrinsically different from the other methods described above. It explicitly targets local structure in a TM, whereas the low-rank methods look for global structure. This difference is one of the key motivations for developing a novel spatio-temporal compressive sensing framework that seeks to capture both global and local structure. It consists of two key components: (i) SPARSITY REGULARIZED MATRIX FACTORIZATION (SRMF) for incorporating global spatio-temporal properties, and (ii) a mechanism for incorporating local interpolation.

3.1 Sparsity Regularized Matrix Factorization

The SRSVD approach starts with (12) to find global low-rank structure in the TM. On the other hand, we may *a priori* know that the matrix has additional spatio-temporal structure, *e.g.*, TM rows or columns close to each other (in some sense) are often close in value. We seek to exploit this insight in the new technique we propose here. We propose to solve the following

$$\begin{aligned} \text{minimize} \quad & ||\mathcal{A}(LR^T) - B||_F^2 + \lambda \left(||L||_F^2 + ||R||_F^2 \right) \\ & + ||S(LR^T)||_F^2 + ||(LR^T)T^T||_F^2, \end{aligned} \tag{14}$$

where S and T are the spatial and temporal constraint matrices, respectively. Matrices S and T express our knowledge about the spatio-temporal structure of the TM (*e.g.*, temporally nearby TM elements have similar values). We solve the above optimization problem again using alternating least squares. We call the resulting algorithm *Sparsity Regularized Matrix Factorization (SRMF)*. It has the advantages of SRSVD, but is more general, allowing us to express other objectives in our TM approximation/interpolation algorithm through different choices of S and T.

Below we discuss how to choose S and T. To better illustrate the idea and benefit of SRMF, we intentionally use relatively simple choices of S and T. In our future work, we will develop techniques to better tailor S and T to dataset characteristics and application requirements. Both SRSVD and SRMF also require specification of the input rank of L and R. Our evaluation in Section 4 shows that SRMF is not sensitive to the input rank parameter.

Choice of T: The temporal constraint matrix T captures the temporal smoothness of the TM. A simple choice for the temporal constraint matrix is $T = Toeplitz(0, 1, -1)$, which denotes the Toeplitz matrix with central diagonal given by ones, and the first upper diagonal given by negative ones, *i.e.*,

$$T = \begin{bmatrix} 1 & -1 & 0 & \dots \\ 0 & 1 & -1 & \ddots \\ 0 & 0 & 1 & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix} \tag{15}$$

This temporal constraint matrix intuitively expresses the fact that TMs at adjacent points in time are often similar. For instance XT^T is just the matrix of differences between temporally adjacent elements of X. By minimizing $||(LR^T)T^T||_F^2$ we seek an approximation that also has the property of having similar temporally adjacent values. We use this simple choice for anomaly detection in Section 5.3 to make comparisons with other anomaly detection algorithms easier. A more sophisticated choice taking into account domain knowledge (say knowledge of the periodicity in traffic data) might result in some improvements. We give such an example for traffic prediction in Section 5.2. In general, it is not difficult to develop such temporal models of TMs.

Choice of S: The spatial constraint matrix S can be used to express which rows of a TM are close to each other, but due to the arbitrary ordering of rows in the TM, a simple matrix of the above form is not appropriate. We find S by first obtaining an initial TM estimate $\mathring{X}$ using a simple interpolation algorithm, and then choosing S based on the similarity between rows of $\mathring{X}$ (which approximates the similarity between rows of X).

1. *Computing $\mathring{X}$.* In our current implementation, we take $\mathring{X} = X_{\text{base}}.*(1-M)+D.*M$, where M is defined in (4) and specifies which TM elements are directly measured, and D contains the direct measurements. That is, we use direct measurements where available, and interpolate using X_{base} at other points.
2. *Choosing S based on $\mathring{X}$.* There are many possible methods for choosing S based on $\mathring{X}$. For example, one general method is to (i) construct a weighted graph G, where each node represents a row of $\mathring{X}$ and each edge weight represents certain similarity measure between two rows of $\mathring{X}$, and (ii) set S to be the normalized Laplacian matrix [7] of graph G, which acts as a differencing operator on G and induces sparsity by eliminating redundancy between similar nodes of G (*i.e.*, rows of $\mathring{X}$).

   We have experimented with several of these methods. The following method for choosing S based on KNN and linear regression consistently yields good performance in our tests, which we will use in our evaluation. For each row i of $\mathring{X}$, we find the K most similar rows $j_k \neq i$ ($k = 1, \dots, K$). We perform linear regression to find a set of weights $w(k)$ such that the linear combination of rows j_k best approximates row i: $\mathring{X}(i,*) \approx \sum_{k=1}^{K} w(k)\mathring{X}(j_k,*)$. Assuming that $\sum_{k=1}^{K} w(k)X(j_k,*)$ approximates $X(i,*)$ well, we then set $S(i,i) = 1$ and $S(i,j_k) = -w(k)$ for $k = 1, 2, \dots, K$ to capture the resulting approximation errors (which are expected to be small).

Scaling of S and T: Finally, we need to scale S and T properly so that $||S(LR^T)||_F$, $||(LR^T)T^T||_F$, and $||\mathcal{A}(LR^T) - B||_F$ are of similar order of magnitude — otherwise they may overshadow each other during the optimization of (14). In our experiments, we simply scale S and T such that $||S\mathring{X}||_F = 0.1\sqrt{\lambda}\,||B||_F$ and $||\mathring{X}T^T||_F = \sqrt{\lambda}\,||B||_F$, where $\sqrt{\lambda}\,||B||_F$ reflects the level of approximation error $||\mathcal{A}(LR^T) - B||_F$ that we are willing to tolerate. Our results show that such scaling yields good performance over a wide range of scenarios and that the performance is not sensitive to the choice of λ. Note that we intentionally make $||S\mathring{X}||_F$ smaller than $||\mathring{X}T^T||_F$ because we expect the temporal model obtained through domain knowledge is more reliable.

3.2 Combining Global and Local Methods

A quick look at the (non-hybrid) performance results that follow shows that for small amounts of missing data, KNN is the best performer (in most cases). On the other hand, for large amounts of loss, SRMF outperforms KNN. The intuition behind this result is obvious. When only a few data points are missing, the k-nearest neighbors of a missing data point will be close by. There are strong temporal and spatial correlations in our data, so the nearest neighbors provide good interpolants for the missing data. However, when there is substantial missing data, the nearest neighbors will come from further away. As the correlations between data points drop, the low-rank global model of the data expressed by the matrix factorization becomes superior.

To take advantage of local structure and redundancy present in the TM, we use the low-rank approximation obtained by SRMF as

a prior and augment it with a local interpolation procedure. In this way, we obtain a TM estimate that is close to the low-rank prior yet can account for constraints imposed by the local interpolation procedure. Note that such an approach generalizes the Tomo-gravity method for TM estimation [31, 33], which uses a rank-1 approximation (*i.e.*, gravity model) as the prior solution.

The choice of the local interpolation procedure is application dependent. Below we present three such examples. For interpolation of missing values, we present *SRMF+KNN* and *SRSVD-base+KNN*, two hybrid algorithms that both incorporate KNN. For network tomography, we present *Tomo-SRMF*, a hybrid algorithm that combines SRMF and Tomo-gravity [33].

SRMF+KNN: We first compute the SRMF interpolation of X. Call this X_{SRMF}. For each missing data point (i, j) we then examine its row to see if any of the elements $X(i, j-3), \ldots, X(i, j+3)$ are present. If we cannot observe any of these neighbors, then we simply use the value $X_{\text{SRMF}}(i, j)$, but if we do have any of these values, we will use them to better approximate $X(i, j)$.

We do so by forming a local model for the temporal process using all of the other rows of the TM. We perform a regression to find a set of weights $w(k)$ that best approximates $X_{\text{SRMF}}(p, j) = \sum_{k \in nbrs} w(k) X_{\text{SRMF}}(p, k)$ for all $p = 1, 2, \ldots, n$. Then we apply a weighted linear interpolation of the nearest neighbors, using the weights derived above, *i.e.*,

$$X_{\text{SRMF+KNN}}(i, j) = \sum_{k \in nbrs} w(k) X(i, k). \qquad (16)$$

SRSVD-base+KNN: We will show that the above approach is superior, but to understand the importance of incorporating the spatio-temporal constraints (given by S and T), we also consider an algorithm that uses SRSVD-base as the prior in the same procedure. We call the resulting algorithm SRSVD-base+KNN.

Tomo-SRMF: In network tomography, we need to infer TMs based on link-load measurements (possibly in combination with direct measurements of a subset of TM elements). The strictly low-rank approximations obtained by SRMF may not satisfy all the measurement equations because (i) real TMs are only approximately low-rank, and (ii) measurement errors are common. A natural solution is to combine SRMF and Tomo-gravity. Specifically, we use SRMF (instead of the gravity model) as a prior solution. We then follow Tomo-gravity and seek a solution that is close to this prior solution (with respect to the Kullback-Leibler divergence) yet satisfies all the measurement equations. We call the resulting hybrid algorithm Tomo-SRMF.

4. INTERPOLATION PERFORMANCE

4.1 Data

The data we use here is real TM data: two standard sets, and one new set. The first two are the Abilene (Internet2) [1] dataset used previously in various studies [13, 14, 30], and the GÉANT TM dataset provided in [26], and previously examined in [2]. Although these are now older datasets, we use them because they are valuable for comparisons with other work. In addition, we use one longer and more recent commercial TM dataset from a large Internet service provider. The properties of the data are summarized in Table 2.

Network	Date	Duration	Resolution	Size
Abilene	Apr. 2003	1 week	10 min.	121×1008
Commercial	Oct. 2006	3 weeks	1 hour	400×504
GÉANT	Apr. 2005	1 week	15 min.	529×672

Table 2: Datasets under study.

4.2 Methodology

The methodology we use here is to drop some data from existing measurements, and then apply the interpolation algorithm. This provides us with ground truth for comparison. The pseudo-missing data is not used in the interpolation algorithms in any way.

The typical approach when measuring the performance of an interpolation algorithm is to drop data at random. We will start our experiments with this case. However, in real measurements of TMs there are different mechanisms that result in missing data, and these result in the missing data having structure. Such structure is obviously important for interpolation, so we will explore several structured models of missing data in Section 4.5 below.

We measure performance using the Normalized Mean Absolute Error (NMAE) in the interpolated values. That is, we calculate

$$NMAE = \frac{\sum_{i,j:M(i,j)=0} |X(i,j) - \hat{X}(i,j)|}{\sum_{i,j:M(i,j)=0} |X(i,j)|}, \qquad (17)$$

where $\hat{X}$ is the estimated matrix. Note that we *only measure errors on the missing values*. So the NMAE is defined only when there is at least one missing value. We computed three other performance metrics (root mean squared error, normalized root mean squared error, and the correlation-coefficient) but the results are substantively the same. In each case we perform the process of randomly dropping data and reconstructing the matrix 10 times. The results presented show the mean NMAE.

4.3 Initial Comparisons

Figure 1 shows a comparison of algorithms for independent random loss for data loss rates ranging from 0.02 to 0.98 (NMAE is undefined when the loss probability is 0). We perform these algorithms using the same regularization and input rank parameters $\lambda = 0.1$ and $r = 8$ for each global algorithm, and $k = 4$ in KNN (we defer justification of these choices to the section below).

For low loss probabilities KNN achieves better performance than SRMF. For high loss probabilities we see that SRMF's performance exceeds KNN. However, the hybrid SRMF+KNN outperforms all algorithms over the whole range of loss values. Interestingly, the hybrid is noticeably better than either method individually.

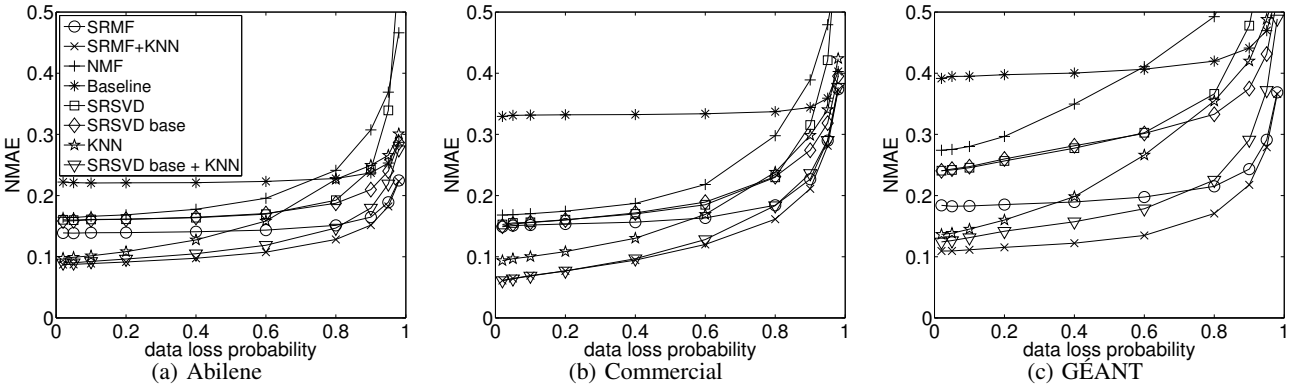
Meanwhile, the hybrid SRSVD-base+KNN also performs well, though not as well as SRMF+KNN. The performance gap typically widens for large amounts of loss. This is because under independent random loss, when the loss rate is not too high, it is likely that the near neighbors of a missing value are directly observed, making KNN an effective recovery strategy. However, when loss is large or when the loss is highly structured (see Section 4.5), the performance gap between SRSVD-base+KNN and SRMF+KNN widens.

The other methods all have worse performance. For low loss, the baseline method is the worst (as we might expect given it is only a rank-2 approximation). However, for high loss, the baseline performs surprisingly well, certainly better than SRSVD, whose performance is very bad for high loss. However, the SRSVD applied after baseline removal achieves reasonable performance over the whole loss range, in some cases almost as good as the simple SRMF. NMF performs poorly for all loss probabilities.

We have examined many such graphs. NMF and SRSVD (without baseline removal) are uniformly poor. So we do not examine them in further results to simplify our presentation.

4.4 Parameter Sensitivity and Settings

The algorithms we consider have several input parameters. The performance of these algorithms in relation to these parameters is (in most cases) dependent on the dataset in question. In practice, when interpolating a real dataset, we would not be able to precisely optimize λ and r for the dataset in question, so it is desirable to have



(a) Abilene (b) Commercial (c) GÉANT

Figure 1: Interpolation performance for random loss (note that the legend is the same for all three plots).

algorithms that are not very sensitive to their values. In fact, all algorithms display some dependence on the parameter settings, and no single parameter setting is optimal for all datasets. However, we found rough parameter settings that are never too far from optimal.

The first input parameter is the rank. Given our motivation from the compressive sensing literature, *i.e.*, that we will aim to minimize matrix sparsity or rank, it may seem strange that we input a rank when performing the algorithm. However, although they seek to minimize the rank of the decomposition, the algorithms work by optimizing an L and R that have a fixed number of columns r (the input rank). The final rank of the solution might be smaller.

In theory, as long as the input rank is greater than the real rank of X, the various algorithms will converge to the correct matrix [5,19, 20]. However, note that the theoretical results that inform our intuition here concern matrices with exact ranks, whereas our matrices typically have a number of small, but non-zero singular values. Moreover, there are measurement errors in our data, so we cannot expect to get zero error reconstructions.

Figure 2 shows a sample of performance results with respect to rank (note that the baseline algorithm is excluded here because it is a fixed rank-2 approximation). We find that most of the rank-dependent methods have better performance as the input rank increases. Although this is not always the case, the deviations are minor. However, note the logarithmic x-axis, so that the results suggest a decrease in the marginal improvement with increasing rank. There is also an additional computational cost for higher ranks, and we find that an input rank of $r = 8$ is a reasonable operating point for our comparisons. Going to $r = 16$ yields only a very small incremental improvement at the expense of extra computation.

The most important finding in these results, however, is the relative insensitivity of the hybrid algorithm, SRMF+KNN. In general it is the least dependent on the input rank of all the algorithms. There is some improvement for higher ranks, but it is typically smaller than those of other algorithms.

KNN does not use input rank, but rather k, the size of neighborhood. Figure 2 also shows the effect of k on the performance of KNN. We choose to use $k = 4$ for our experiments, since it consistently avoids the worst results.

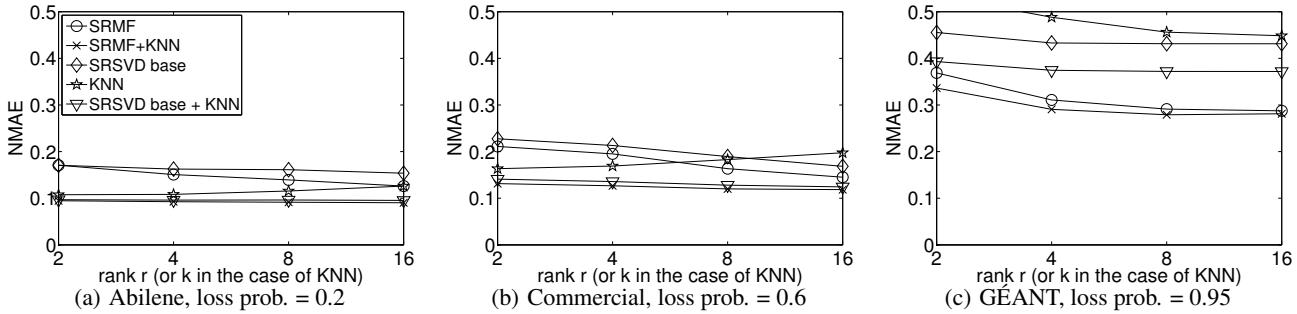
The final parameter of importance is the regularization parameter λ, which determines the tradeoff (in the optimization) between the measurement constraints and the importance of rank. Larger λ leads to lower rank approximations, whereas smaller values lead to approximations that are a better fit to the data. Figure 3 presents three examples showing the type of variability we encounter over a range of values of λ, for three different loss rates and networks. KNN is omitted because it does not use regularization. Once again note the logarithmic x-axis – we are looking for order of magnitude effects here, not fine tuning. None of the techniques is too sensitive. Among them, SRSVD is the most sensitive (overall). Larger values of λ typically perform better although again sometimes this trend is reversed, and there are a number of cases where the optimal case is around $\lambda = 0.1$. So we use this value in our experiments.

Note again that SRMF+KNN is the most insensitive algorithm with Figure 3 (c) showing the most extreme case of parameter sensitivity that we observed for this algorithm.

4.5 Comparison: Other Loss Models

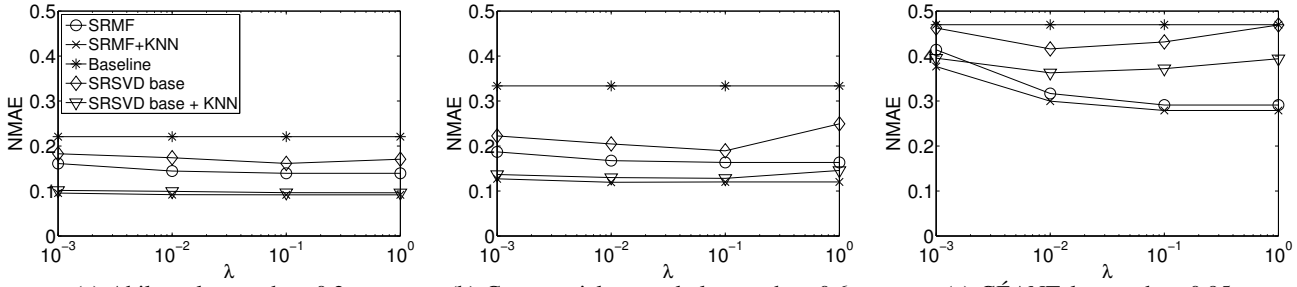
As earlier noted, not all data loss is random. Losses are often highly structured, and in this section we examine the effect this has on the results. The boldface name denotes the label used in our datasets, where **xx** is replaced by the percentage of rows (or columns) effected.

1. **PureRandLoss:** This is the simple random loss model. Data points in the matrix X are dropped independently at random with probability q.
2. **xxTimeRandLoss:** This simulates a structured loss event where we suffer data loss at certain times. For example a certain time points our monitoring equipment might become overloaded, or a disk might fills up. In these cases, we may loose some random proportion of the data at a particular point in time. We simulate this loss by choosing, at random **xx**% of the columns of X, and dropping data from these at random with probability q. Note that the case 100ElemRandLoss corresponds to PureRandLoss, so we do not repeat this case.
3. **xxElemRandLoss:** This simulates a structured loss event where a set of randomly chosen TM elements suffers from lost data. This type of loss might occur where unreliable transport mechanisms are used to transport measurements back to the network measurement station. Often the problems with such transport depend on the locations where measurements are made (*e.g.*, locations close to the management station are less likely to suffer congestion based losses). We randomly select **xx**% of the rows of X to be effected. Note that the case 100ElemRandLoss corresponds to PureRandLoss, so we do not repeat this case.
4. **xxElemSyncLoss:** This simulates a structured loss event where a group of TM elements all suffer from missing data from the same cause. Hence, the losses on each element are synchronized. We do so by selecting **xx**% of rows of X to be effected, and a set of times with probability q. Lost data comes from the intersection of the selected rows and columns.
5. **RowRandLoss:** Random element loss, as presented above, is not a particular realistic model for data loss. With flow level measurements, data are collected by a router. If that router cannot collect data, then an entire row of each TM snapshot Z will be missing. The effect on X is to remove a set of structurally associated rows. We simulate this by dropping rows from the original TM Z (before it is formed into the matrix X).



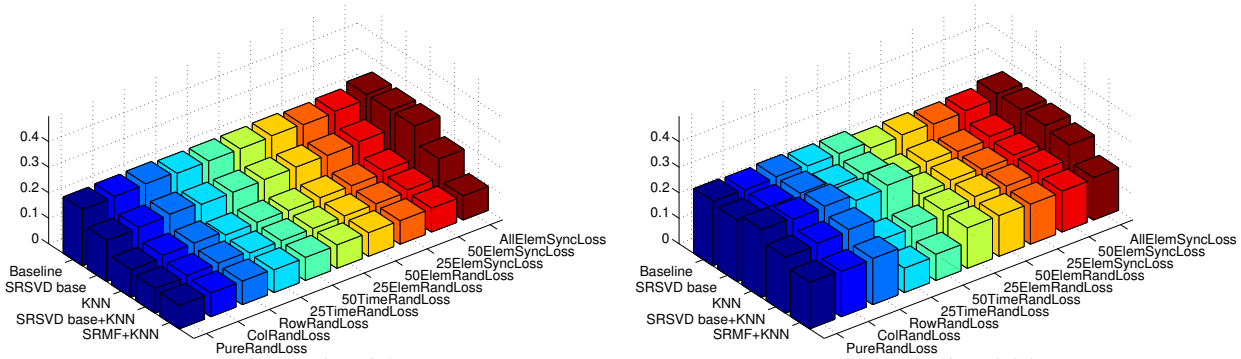
(a) Abilene, loss prob. = 0.2 (b) Commercial, loss prob. = 0.6 (c) GÉANT, loss prob. = 0.95

Figure 2: Sensitivity with respect to the input rank r (or k in the case of KNN).



(a) Abilene, loss prob. = 0.2 (b) Commercial network, loss prob. = 0.6 (c) GÉANT, loss prob. = 0.95

Figure 3: Sensitivity with respect to λ.



(a) loss prob. = 0.2. (b) loss prob. = 0.95.

Figure 4: Comparison between algorithms for the different loss models.

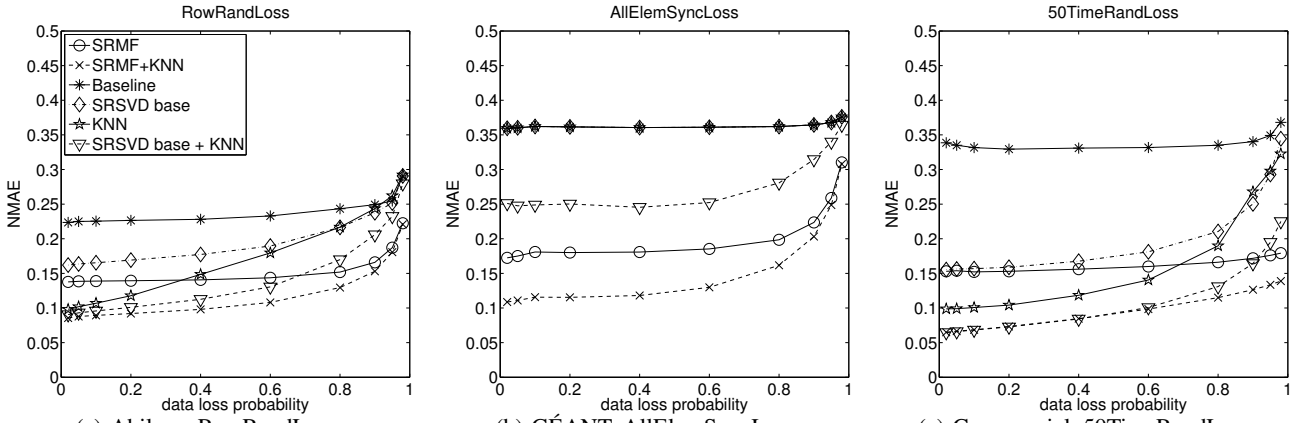
6. **ColRandLoss:** It is perhaps less likely that a column of the original TM Z is dropped from measurement. One can construct scenarios where a software bug causes such an error, but in fact we primarily consider the random column loss scenario for completeness.

In this section we examine the impact of the loss model on the performance of the interpolation algorithms. Obviously there are many ways of viewing this data. Due to space limitations, we present here only a few representative ones. First, Figure 4 shows bar charts of the performance of the key algorithms for two different loss levels, across all loss models. The key observations are that for low- to moderate loss, SRMF+KNN performs significantly better across all loss models. When loss is higher, there are some cases where the performance of SRSVD-base and KNN is similar to SRMF+KNN, and occasionally slightly better, but where losses are highly structured (*e.g.*, AllElemSyncLoss) SRMF+KNN is always clearly superior.

We show three of these cases in more detail in Figure 5. Figure 5(a) shows the Abilene network data, with random row loss. The results for random column loss are similar, and both are qualitatively the same as those for pure random loss. The reasons are clear when we consider the high loss case where both the baseline (used in SRSVD-base) and the similarity metric used for KNN are hard to calculate because entire rows or columns have no data. On the other hand, our approach combines the spatial and temporal components in its model.

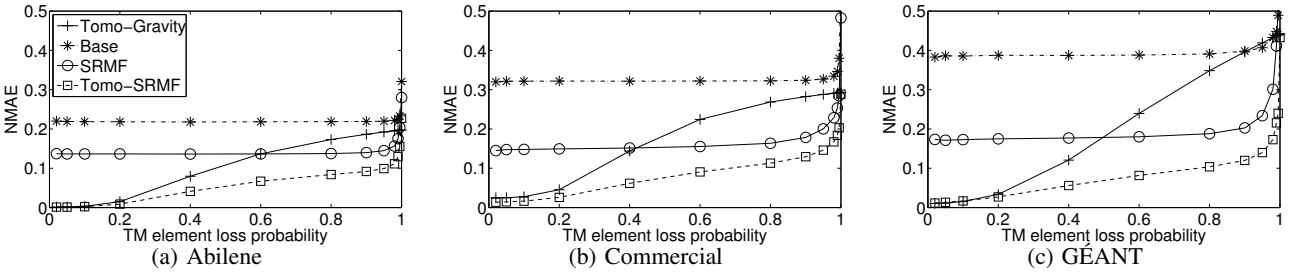
Figure 5(b) shows the extreme of very structured loss (synchronized in time). In this case, the baseline is so poor that all of the other techniques collapse back to this baseline. Our approach still performs reasonably. Figure 5(c) shows the case of random column damage (with about half of the rows affected). In this case, our approach performs surprisingly well given that so much of the structure of the matrix has been lost. This is yet another indication of the importance of the spatio-temporal model.

Summary: These results show that SRMF+KNN is the best algorithm over a wide range of loss scenarios and loss rates. In the few cases where SRMF+KNN does not win, it is not far behind. Meanwhile, SRSVD-base+KNN consistently performs better than SRSVD-base, but not as well as SRMF+KNN, especially when there is a large amount of structured loss. These results clearly demonstrate the power of our spatio-temporal compressive sensing framework to simultaneously exploit global and local structures and leverage spatial and temporal properties. We expect more detailed modeling of the spatial and temporal constraint matrices S and T to further improve the accuracy.



(a) Abilene, RowRandLoss. (b) GÉANT, AllElemSyncLoss. (c) Commercial, 50TimeRandLoss.

Figure 5: Comparison for different loss models.



(a) Abilene (b) Commercial (c) GÉANT

Figure 6: Network tomography performance. Note, loss probability=1 corresponds to the traditional network tomography problem.

4.6 Computational Times

We measure the computation times of SRMF (KNN takes a small amount of additional time) on matrices of various sizes. The computation time is linear in the number of matrix elements, and quite reasonable. A 200×1000 matrix (with rank $r = 10$ used in the decomposition) can be approximated in less than 3.5 seconds (using Matlab on a 2.66 GHz Linux machine). Meanwhile, the computation time with respect to r is reasonably modeled by $O(r^2)$.

5. APPLICATIONS

The previous section considered the simple problem of interpolation. We now consider different applications of matrix interpolation, where the meaning or importance of the missing values are determined by the application in question.

5.1 Tomography

A special case of the our approach is the network tomography problem of inferring a TM from link-load measurements. In the previous cases, the constraints come from direct measurements. In network tomography the constraints are given by (1). However, it is common to have some combination of these sets of measurements. So it is desirable to combine them to obtain the best possible approximation to the TM. In this case, we can simply define $\mathcal{A}$ to incorporate both (1) and (3), resulting in a combined penalty term of the form $||A(LR^T) - Y||_F^2 + ||(LR^T - D). * M||_F^2$.

Due to lack of space, we do not compare all possible algorithms for TM estimation, but concentrate on two simple and relatively well known algorithms. Further performance improvements might be obtained by using more recently developed algorithms (*e.g.*, [9]), but the insights are still useful. The two existing algorithms we consider are the gravity model and Tomo-gravity. The gravity model [33] is a simple rank-1 approximation to a single TM. It is known to be a poor estimator of real TMs, but it has been successfully used as the first step in the Tomo-gravity [33] algorithm. The latter is yet another regularization based on the Kullback-Leibler divergence between the gravity model and the measurements.

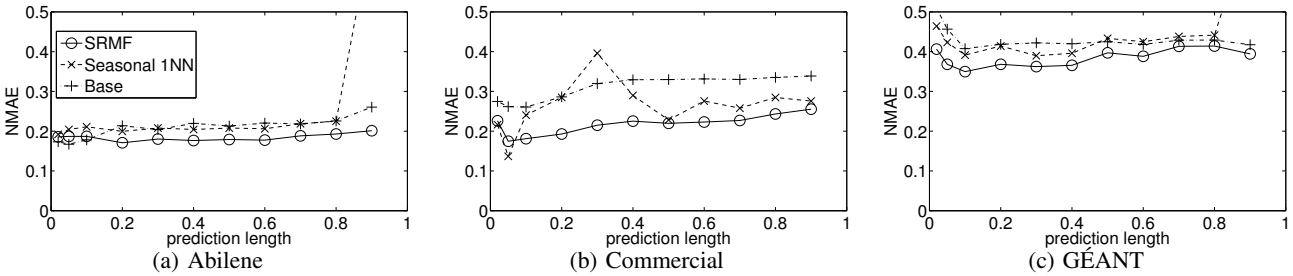
	Abilene	Commercial	GÉANT
Tomo-gravity	0.197 / 0.197	0.292 / 0.292	0.441 / 0.439
Base	0.321 / 0.233	0.566 / 0.380	1.198 / 0.489
SRMF	0.280 / 0.204	0.483 / 0.285	1.185 / 0.516
Tomo-SRMF	0.227 / 0.155	0.288 / 0.203	0.433 / 0.240

Table 3: Network tomography performance: the first number is the performance where we have no direct TM measurements, the second shows where we measure only 0.5% of the elements.

The notable feature (for the purpose of this paper) of the gravity model and Tomo-gravity is that neither involve temporal information. They operate purely on a TM snapshot. The gravity model is based purely on row and column sums of the TM (snapshot) and so has no need (or method) for incorporating additional information. However, Tomo-gravity is just a standard regularization, and so additional measurement equations can be easily added.

In this section we compare these algorithms against three alternatives: the baseline approximation, SRMF, and Tomo-SRMF. In Figure 6 we show the performance of the algorithms with respect to the proportion of the TM elements that are missing, but note that in addition to direct measurement of the matrices, we assume we can measure all of the link loads on the networks. So in this figure, 100% data loss corresponds to the standard network tomography problem. As this part of the figure is important, but relatively hard to read, we have duplicated key performance metrics in Table 3.

First, note that the gravity model is poor enough that its results lie off the scale. The baseline technique is the second worst in most cases, but is still much better than the gravity model. Second, SRMF performs poorly at the pure network tomography task where no direct measurements are available. However, if even a few (as few as 0.5%) of the TM elements are directly observed, then SRMF's performance improves dramatically, whereas Tomo-gravity's performance improves roughly linearly with respect to the



(a) Abilene (b) Commercial (c) GÉANT

Figure 7: Network prediction performance.

increase in information. Finally, by combining SRMF and Tomogravity, Tomo-SRMF gets the best of both worlds and significantly outperforms each individual method by itself. Figure 6 and Table 3 show the improvements.

Note that Soule *et al.* [25] also propose to incorporate flow-level measurements in TM estimation. Compared with their "third generation" TM estimation methods, Tomo-SRMF has two key advantages: (i) it does not require any expensive calibration phase in which *entire* TMs are directly measured, and (ii) it is highly accurate and can reduce the error of Tomo-gravity by half with only 0.5-5% observed TM elements (whereas 10-20% directly observed TM elements are required according to [25]).

5.2 Prediction

In this section we consider the behavior of SRMF with respect to TM prediction. We do so by dividing our data into two segments, an initial training segment up to some time t, and then a test segment over which we try to predict the TM.

Prediction is rather different from the general problem of interpolation. Several techniques (SRSVD and NMF) just fail. KNN does not work well because there are no temporally "near" neighbors, and no spatial neighbors at all. However, if we can use the temporal pattern in the data more creatively we can make progress. For instance, rather than using a simple nearest neighbors technique, we use seasonal nearest neighbors. TMs show strong diurnal patterns, and so it is not surprising that offsetting neighbors by the 24 hours period has benefits. In essence, the seasonal nearest neighbor approach assumes that today's traffic has the same pattern as yesterdays.

Likewise for SRMF, we do not need to use the spatial constraint matrix, as an entire slab of the data is missing (the future data we are trying to predict). However, to allow a fair comparison with seasonal nearest neighbors, we also use seasonality in constructing our T matrix. We construct a difference matrix, but where the interval between the differences is 24 hours.

Figure 7 shows the results with respect to the proportion of data being predicted. Note that SRMF outperforms the other techniques, and further that SRMF's performance degrades very slowly as the length of data being predicted increases (and the training data gets correspondingly smaller). This shows that typical TMs exhibit temporal regularity and SRMF can effectively take advantage of it.

5.3 Anomaly Detection

A common task in network operations is finding problems. There are specific tools for finding some problems (*e.g.*, SNMP is commonly used to find link failures), and other problems such as specific attacks can be characterized by a *signature*, which signals the attack. However, both of the above approaches rely on preknowledge of the problems that we will encounter. There is a complementary need to find unanticipated problems in networks.

Such problems cannot be characterized before-hand, and so the method commonly used to detect such *anomalies* is to find significant differences from historical observations. Most approaches involve some transformation of the data followed by outlier detection [30]. Common examples include simple filtering of the data, Fourier transform, wavelets, or PCA. The transform is aimed at separating the "normal" space of historical events, from the anomalous space. Techniques such as PCA do this explicitly, while others rely on commonly observed properties. For example, Fourier techniques rely on the normal data primarily inhabiting a low- to mid-frequency space, so that anomalies involve high-frequencies such as those incurred by a rapid change. Outlier detection can be performed by taking the normal model of the data, and comparing its values at each time point with the real data, and then seeking points where the difference exceeds some threshold T.

In this section we will compare several approaches to anomaly detection. To keep things simple so that we can gain an intuitive understanding of the various properties of different approaches, we will consider only three algorithms one temporal, one spatial, and our spatio-temporal approach. The three approaches we use are

1. **Differencing:** Differencing is a standard time-series technique to remove linear trends (typical traffic data are non-stationary, and over periods of minutes to hours can often be reasonably approximated as having a linear trend). Differencing also highlights sudden changes, such as we would see in a traffic *spike* or a level shift [30]. Implicitly, differencing is using the data from the previous time step as a model for the current time, and so it has not received a great deal of consideration in the networking literature, but it provides a simple temporal benchmark against which we can gain some intuition. We can write the differencing operator as postmultiplication of X by $T = Toeplitz(0, 1, -1)$, a purely temporal operation that makes no use of spatial correlations between TM elements.
2. **PCA/SVD:** PCA/SVD has received much interest for network anomaly detection in recent years [12–14, 21, 29, 30], and is the only common spatial technique for anomaly detection. As noted earlier, PCA/SVD is applied by choosing the rank r of the normal subspace (based on the power contained in the first r singular values), and projecting the input data X into the abnormal subspace, where artifacts are then tested for size. Implicitly, we are looking at the difference between the normal model of the data created by the low-rank SVD approximation and the data itself. Intuitively, the process builds a (simple) model from the historical relationships between TM elements. New time points are compared to see if they satisfy this relationship. If not, they are declared to be anomalies. It is a purely spatial technique, since reordering of the data in time (the columns of X) has no effect on the results. Interestingly, compressive sensing ideas have already appeared in the context of PCA based anomaly detection [12], though in that context the goal was to reduce the volume of data transmitted to a NOC, and the missing data could be controlled, whereas in our context the missing data are out of our control.
3. **SRMF:** In this context we apply SRMF directly to the traffic data *including* the anomalies, much as one would with SVD. Our technique, however, is truly spatio-temporal as the model that we create involves both the spatial and temporal properties

of the underlying TM. The low-rank approximation is then used as a model for the normal data, and the differences between this and the real traffic are used to signal anomalies. Once again, we use the standard method of thresholding these differences to detect outliers.

We will compare each of these algorithms using simulations. Ringberg *et al.* [21] explain in detail why simulation should be used for accurate comparisons of anomaly detection techniques. In brief their reasons are: (i) accurate and complete ground truth information is needed to form both false-alarm and detection probability estimates (both are needed for comparisons, as one by itself can be entirely misleading); (ii) many more results are needed (than one can obtain from data) to form accurate estimates of probabilities, and (iii) simulation allows one to vary parameters (say the anomaly size) to study their effects. Simulation is necessary, but not sufficient for validation, so we expect that further work is needed on this type of anomaly detection before it is used by network operators.

Our approach to simulation is intended to highlight the features of the different techniques. We make no claim that the simulation is completely realistic, only that it clearly illustrates the properties of the different anomaly detection techniques. We simulate in two steps: we first create the normal traffic, and then inject anomalies. We create the TM by an orthogonal composition of a synthetic gravity model TM [22] in the spatial domain, and a periodic Norros model [18, 23] in the temporal domain. Both models have arguments in their favor but principally we need to create a TM with low rank, but some local spatio-temporal structure that we might find in a real TM.

We use this model to generate 1000 instances of the TM X consisting of one weeks worth of measurements at 15-minute intervals. In each instance we inject one anomaly. The anomaly is a spike added to the TM at a single randomly chosen time point, so that one anomaly cannot interfere with the detection of another. The value of the spike is a vector of Gaussian random variables (in each element of the TM) but we normalize the total size of the spike (measured by its L_2 norm) to be a fixed size, which we vary from 0.1 to 100. Spikes of size 0.1 (in our dataset) are almost indistinguishable (by the naked eye) from the standard random variations in the traffic. Spikes of size 100 are much larger than the typical TM elements, and so are easily observed. We then apply each of the three techniques above to create a "normal" traffic vector, and detect anomalies by thresholding on differences between the normal and measured vector. Note that we do not have missing data in the inputs (it is not obvious how to fairly compare the three algorithms when there are missing data, given the better interpolation performance of SRMF). However the anomalies *are* included in the inputs, so that both SVD and SRMF can be compared fairly.

An important detail is the choice of thresholds for outlier detection. Non-stationarity in our data makes setting thresholds more difficult than in some problems. When the anomalies are small, a little fine tuning allows us to find threshold values that are statistically indistinguishable for all three methods. Figure 8 shows a comparison between the false-alarm probabilities for the three techniques, showing 95% confidence intervals for the estimates. For larger anomalies it is hard to tune the false-alarm probabilities for PCA/SVD. The anomaly pollutes the data used to create the normal subspace, and so invalidates the standard statistical assumptions used to determine thresholds [21]. So it is hard to obtain thresholds that produce the same false-alarm probabilities for large anomalies, but the differences in these cases will be inconsequential for the results. Likewise, it is hard to tune the false-alarm rate for SRMF and large anomalies, but for the opposite reason: the false-alarm rate drops almost to zero too quickly. Given the high detection rate, this is not a problem in the comparison of results.

Figure 9 shows the detection probabilities for the three techniques. First, let us compare PCA/SVD and Differencing for small anomalies. When anomalies are small (in value) and hence the differences they create will be small, the probability of detection of these differences will be small. Note that, even though we allow a larger false-alarm probability as the anomalies get smaller, the detection probability for Differencing drops dramatically. On the other hand, PCA/SVD exploits the spatial relationships between the data elements. It uses all the information available at a particular time point instead of processing the information from each time sequence independently. While the performance of PCA/SVD also declines for smaller anomalies, it is much better than Differencing.

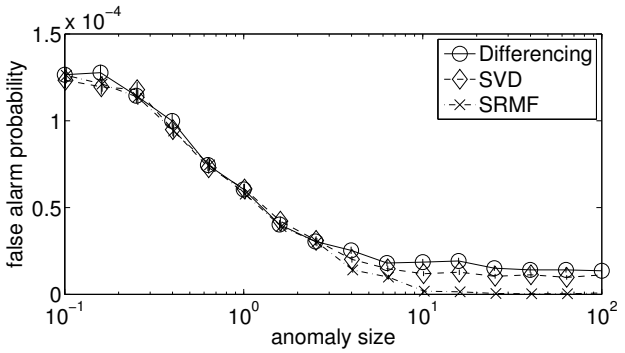


Figure 8: False alarm probability.

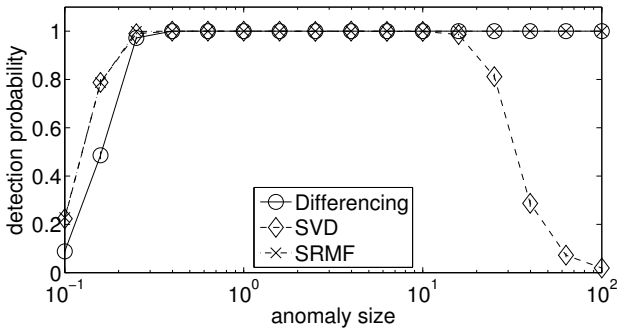


Figure 9: Detection probability.

For moderate sized anomalies, both techniques have perfect detection records. These anomalies are not particularly hard to detect whichever method one uses. On the other hand, when we consider large anomalies, a different story emerges. The larger anomalies pollute the data used to create the normal subspace in PCA/SVD, and this has a tangible effect in that the detection probability drops dramatically. Note that although the false-alarm probabilities of the Differencing and PCA/SVD methods are slightly different, the detection probability of PCA/SVD drops almost to zero, whereas Differencing maintains its perfect detection record. So we can discount the minor difference in false-alarm probability as causing this drop. The result is consistent with those observed in data [21].

A solution might be to temporally pre-filter the data to remove large anomalies, and then apply PCA/SVD. This introduces the problem of interpolating data, which we have noted before. A preferable approach would be to use an approach that deals well with both ends of the spectrum. SRMF provides such a method. Figures 8 and 9 show its performance. We can see that its detection probability is statistically indistinguishable from the better of the two other methods for each anomaly size. The false alarm probability is either indistinguishable or well below that of the other two methods. So SRMF provides a method that deals well with the complete range of anomalies.

SRMF does this through its use of a spatio-temporal model. In the case where the spatial model is more effective (small anomalies) this is the part of the algorithm that is "biting", whereas when the anomalies are large, they are being detected in the temporal do-

main, essentially by differencing. What we see here is that by imposing temporal smoothness constraint on LR^T, the effect of contamination is much smaller. Intuitively, if too much energy leaks into the normal subspace (as in PCA), then the projection of X into the normal subspace is no longer smooth, which would then result in a too big penalty in the smoothness term. Thus the smoothness term helps to limit the damage of contamination, and avoids the problem seen in PCA/SVD.

Note that we do not argue that with the naive choices of temporal operator T that we use here that SRMF is the best prediction or anomaly technique for TMs. Given the wealth of methods available for these applications (*e.g.*, see [30]), one can undoubtedly do better by more careful choice of T. However, there is a lesson to be learned here. First, our regularization approach can be generalized to apply to any linear prediction/anomaly detection technique through appropriate choice of T. In each case we would hope for performance improvements as well, but the more important aspect of this work comes from the features we have demonstrated above: (i) our approach naturally deals with missing data, (ii) it can flexibly include additional data sources (*e.g.*, link data), and (iii) anomaly detection (and exclusion) are an inherent part of the algorithm. We argue that these are ideal features for any set of algorithms based on TMs.

6. CONCLUSIONS AND FUTURE WORK

By drawing on recent developments in compressive sensing and relying on readily available domain knowledge in the area of TMs, we present in this paper a unified approach to measurement and analysis of TMs. We achieve this by developing a novel spatio-temporal compressive sensing framework that exploits the presence of both global structure (*e.g.*, low rank) and local structure (*e.g.*, spatio-temporal properties) in real-world TMs. Whether applied to TM estimation (*i.e.*, tomography), TM prediction, or anomaly detection, our algorithms consistently outperform other commonly-used methods and do so across the whole range of missing values scenarios, from purely random ones to highly structured ones where whole columns and/or rows of a TM are missing, and from very low levels of missing values to very high levels (*e.g.*, 90% and more). The main reason for the superior performance of our proposed technique when compared to its most widely-used competitors is its reliance on truly spatio-temporal models of TMs that capture much of the localized structure inherent in actual TMs.

There are a number of avenues for future work. First, we plan to better tailor our approach to exploit the characteristics of real-world TMs through more detailed modeling of the spatial and temporal constraint matrices S and T. Second, many of the techniques described here (including SRMF) naturally extend to tensors (*i.e.*, multi-dimensional arrays), so that the original (unvectorized) TMs can be analyzed directly, *i.e.*, as true 3-d objects with traffic source, traffic destination, and time as the three axis. Such a tensor treatment of TMs has great potential and presents an opportunity to build more sophisticated spatio-temporal descriptions of the TM. Third, we want to more thoroughly explore the application of our approach to enable scalable network measurement and support important network engineering tasks such as anomaly detection. Finally, we would like to formally understand the theoretical properties of our spatio-temporal compressive sensing framework.

Acknowledgments

We thank Patrick Thiran and the anonymous reviewers for their valuable feedback. This work was supported in part by NSF grants CNS-0546720, CNS-0615104, and CNS-0627020, and ARC grant DP0665427. We would also like to thank UCLA IPAM for providing a forum for many valuable discussions related to this work, and the Abilene and GÉANT networks for providing data.

7. REFERENCES

[1] The Abilene Observatory Data Collections. `http://abilene.internet2.edu/observatory/data-collections.html`.

[2] D. Alderson, H. Chang, M. Roughan, S. Uhlig, and W. Willinger. The many facets of Internet topology and traffic. *Networks and Heterogeneous Media*, 1(4):569–600, 2006.

[3] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proc. of ACM SIGCOMM Internet Measurement Workshop*, 2002.

[4] R. Bell, Y. Koren, and C. Volinksy. Chasing the $1,000,000: How we won the Netflix progress prize. *Statistical Computing and Graphics*, 18(2), 2007.

[5] E. Candes and B. Recht. Exact matrix completion via convex optimization. preprint.

[6] 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.

[7] F. R. K. Chung. *Spectral Graph Theory*. CBMS Lecture Notes. AMS Publications, 1996.

[8] D. Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 52(4):1289–1306, 2006.

[9] V. Erramilli, M. Crovella, and N. Taft. An independent-connection model for traffic matrices. In *Proc. of Internet Measurement Conference (IMC)*, 2006.

[10] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. *IEEE/ACM Transactions on Networking*, pages 265–279, 2001.

[11] B. Fortz and M. Thorup. Optimizing OSPF/IS-IS weights in a changing world. *IEEE JSAC Special Issue on Advances in Fundamentals of Network Management*, Spring 2002.

[12] L. Huang, X. Nguyen, M. Garofalakis, J. Hellerstein, M. Jordan, M. Joseph, and N. Taft. Communication-efficient online detection of network-wide anomalies. In *Proc. of IEEE INFOCOM*, 2007.

[13] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proc. of ACM SIGCOMM*, 2004.

[14] 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 / Performance*, 2004.

[15] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proc. of Neural Information Processing Systems (NIPS)*, pages 556–562, 2000.

[16] Y. Mao and L. K. Saul. Modeling distances in large-scale networks by matrix factorization. In *Proc. of Internet Measurement Conference (IMC)*, 2004.

[17] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proc. of ACM SIGCOMM*, 2002.

[18] I. Norros. A storage model with self-similar input. *Queueing Systems*, 16:387–396, 1994.

[19] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum rank solutions to linear matrix equations via nuclear norm minimization. preprint.

[20] B. Recht, W. Xu, and B. Hassibi. Necessary and sufficient conditions for success of the nuclear norm heuristic for rank minimization. In *Proc. of 47th IEEE Conference on Decision and Control*, 2008.

[21] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of PCA for traffic anomaly detection. In *Proc. of ACM SIGMETRICS*, 2007.

[22] M. Roughan. Simplifying the synthesis of Internet traffic matrices. *SIGCOMM Computer Communications Review*, 35(5):93–96, 2005.

[23] M. Roughan and J. Gottlieb. Large-scale measurement and modeling of backbone Internet traffic. In *Proc. of SPIE ITCOM*, 2002.

[24] M. Roughan, M. Thorup, and Y. Zhang. Traffic engineering with estimated traffic matrices. In *Proc. of Internet Measurement Conference (IMC)*, 2003.

[25] A. Soule, A. Lakhina, N. Taft, K. Papagiannaki, K. Salamatian, A. Nucci, M. Crovella, and C. Diot. Traffic matrices: Balancing measurements, inference and modeling. In *Proc. of ACM SIGMETRICS*, pages 362–373, 2005.

[26] S. Uhlig, B. Quoitin, S. Balon, and J. Lepropre. Providing public intradomain traffic matrices to the research community. *ACM SIGCOMM CCR*, 36(1):83–86, 2006.

[27] Y. Vardi. Network tomography. *Journal of the Amer. Stat. Assoc.*, Mar. 1996.

[28] G. Varghese and C. Estan. The measurement manifesto. In *Proc. of 2nd Workshop on Hot Topics in Networks (HotNets-II)*, 2003.

[29] K. Xu, J. Chandrashekar, and Z.-L. Zhang. A first step toward understanding inter-domain routing dynamics. In *Proc. of ACM SIGCOMM Workshop on Mining Network Data (MineNet)*, pages 207–212, 2005.

[30] Y. Zhang, Z. Ge, M. Roughan, and A. Greenberg. Network anomography. In *Proc. of Internet Measurement Conference (IMC)*, 2005.

[31] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale IP traffic matrices from link loads. In *Proc. of ACM SIGMETRICS*, 2003.

[32] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proc. of ACM SIGCOMM*, 2003.

[33] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. Estimating point-to-point and point-to-multipoint traffic matrices: An information-theoretic approach. *IEEE/ACM Transactions on Networking*, 13(5):947–960, 2005.

[34] Q. Zhao, Z. Ge, J. Wang, and J. Xu. Robust traffic matrix estimation with imperfect information: Making use of multiple data sources. *SIGMETRICS Perform. Eval. Rev.*, 34(1):133–144, 2006.