Blind Source Separation Approach to Performance Diagnosis and Dependency Discovery

Gaurav Chandalia and Irina Rish
IBM T.J. Watson Research
Hawthorne, NY 10532, USA
Email: {gachanda,rish}@us.ibm.com

ABSTRACT

We consider the problem of diagnosing performance problems in distributed system and networks given end-to-end performance measurements provided by test transactions, or probes. Common techniques for problem diagnosis such as, for example, codebook and network tomography usually assume a known dependency (e.g., routing) matrix that describes how each probe depends on the systems components. However, collecting full information about routing and/or probe dependencies on all systems components can be very costly, if not impossible, in large-scale, dynamic networks and distributed systems. We propose an approach to problem diagnosis and dependency discovery from end-to-end performance measurements in cases when the dependency/routing information is unknown or partially known. Our method is based on Blind Source Separation (BSS) approach that aims at reconstructing unobserved input signals and the mixing-weights matrix from the observed mixtures of signals. Particularly, we apply sparse non-negative matrix factorization techniques that appear particularly fitted to the problem of recovering network bottlenecks and dependency (routing) matrix, and show promising experimental results on several realistic network topologies.

1. INTRODUCTION

Monitoring and diagnosis of distributed computer systems and networks is an important issue in systems management that becomes increasingly challenging with growing size and complexity of such systems. Given the heterogeneous, decentralized and often noncooperative nature of today's large-scale networks, it is impractical to assume that all statistics related to an individual system's components such as link, routers, or application-layer components can be indeed collected for monitoring purposes. On the other hand, other type of measurements, such as end-to-end transactions, or probes, are typically cheap and easy to obtain. This realization gave rise to the field of *network tomography* [13] which focuses on inference-based approaches to estimate unavailable network characteristics from available measurements.

Most of existing work on network tomography and problem diagnosis, such as commonly used *codebook*-based approaches [5, 9]

assume known dependencies between the observations (e.g., end-to-end delays, system events) and their unobserved causes (e.g., link or node delays, component failures). Such knowledge is represented by routing matrices in network tomography or dependency matrices in problem diagnosis, where rows and columns correspond to observations and causes (components) respectively. However, obtaining dependency information can be too costly or just infeasible in many situations: network topology and routing information may be unavailable due to noncooperative administrative domains blocking the access to topology discovery tools, components that affect probe's performance may be hard to discover (e.g., low-level network elements or high-level application components such as particular set of database tables crucial for transaction performance), maintaining up-to-date information about dynamically changing routing (especially in wireless and mobile networks) may get costly, and constructing dependency matrices for application-level transactions is typically quite a laborious process requiring expert knowledge of system components that may affect probe's performance. Even in cases when the dependency or routing information is available, this knowledge is often only partial, and also dynamically changing.

Therefore the questions are: how much can we infer from only end-to-end observations? Can we infer "hidden causes" of performance degradations? Can we conclude that there exist, say, three common components that would explain most of the end-to-end delays? Can we say whether the performance degradations are due to a single bottleneck or due to an overall performance degradation that affected the whole network? And, if we think it is due to a bottleneck or a few of them, can we identify which probes go through which subset of such bottlenecks?

In this paper, we propose a general framework that attempts to answer the above questions by performing simultaneous dependency discovery and problem diagnosis. We use the Blind Source Separation (BSS) approach that aims at reconstructing both unobserved input signals and the mixing-weights matrix from the combined signals received at a given sequence of time points[1]. A classical example of BSS is the "cocktail party" problem where n speakers are present in the room with m microphones and the task is to reconstruct what each one was saying (n signals) and how close to the microphones they were located ($m \times n$ mixing matrix). In our application, causes of performance problems such as delays at individual components (nodes, links) correspond to the unobserved

---

[1]Note that assuming complete absence of information about dependency (routing) information is the opposite extreme of assuming the full knowledge, and in reality we hope to have some partial information available. However, in this paper, we investigate the most "pessimistic" scenario first, and leave combination of BSS approaches with some partial dependency knowledge as a topic of future investigation.

input signals, while dependency (routing) matrix corresponds to the mixing-weights matrix, and the end-to-end probes to the observed output signals.

Herein, we apply BSS approaches based on sparse nonnegative matrix factorization and report encouraging empirical results in both real and simulated settings.

2. RELATED WORK

Most of recent work on network tomography falls into two categories: (i) estimating link-level performance such as link delays or losses based on end-to-end performance measurements [6, 12, 10] and (ii) estimating origin-destination (OD) traffic flows based on link-level traffic measurements [7, 16, 15] (see [1] for an overview and an extended list of references). A typical approach to both problems is to assume a noisy linear model $\mathbf{y}_t = A\mathbf{x}_t + \epsilon$ where $\mathbf{y}_t$ denotes a vector of observations (e.g., end-to-end delays or link-level traffic intensities), A denotes a *routing matrix*, $\mathbf{x}_t$ is an unobserved vector (e.g., link delays, or OD flows) and ϵ is noise. In this paper, we focus on the first problem, although our method can be easily applied to the second problem as well. We also generalize the problem from the network to application level where $\mathbf{y}_t$ and $\mathbf{x}_t$ represent arbitrary end-to-end transactions and corresponding delays at system components and A corresponds to a *dependency matrix*, where $a_{ij} = 1$ if the probe i "goes through" component j and 0 otherwise. For example, response time of a web-page request depends not only on network components such as routers and links, but also on the web server performance, various applications invoked on the page, database tables that need to be opened in order to show the content of the page, etc.

There also exists a body of related work in network tomography that focuses on discovering (logical) topology of a network based only on end-to-end measurements (see [1] for a comprehensive summary; a more recent approach was also presented by [11]). These approaches attempt to reconstruct the routing tree (typically assuming multicast, although extensions to unicast probes were also proposed) by comparing the shared loss or delay statistics on probe packets transmitted from a root to a set of leaf nodes, and often can be viewed as hierarchical clustering that uses some similarity measure to group the nodes into a (logical) routing tree. Note, however, that topology discovery may be an overkill in case of bottleneck diagnosis problem, since we only need to know the *set* rather than the *sequence* of components involved in each probe, i.e. only the dependency matrix. Moreover, those approaches are also quite specific to the network topology discovery and cannot be directly applied to the application-level dependency matrix reconstruction. On the contrary, our approach is more general as it applies to arbitrary topologies, does not make multicast assumptions, and can be used with any type of end-to-end probes from network to application layer[2].

3. OUR APPROACH

Our method uses an analogy to the BSS problem. In the application to the systems performance management, we can view the delay experienced by a transaction at each component as an unobserved "signal", the unknown dependency (routing) matrix as a mixing-weights matrix and the observed end-to-end performance as the output signal (e.g., corresponding to a "microphone" in the "cocktail party" problem). The BSS problem is solved by matrix

factorization: given the $p \times T$ matrix of end-to-end probe observations Y where p is the number of probes and T is the number of time points, find two matrices A and X that provide the best possible approximation to Y as a factorization $Y = A \cdot X$, where A corresponds to the found $p \times n$ dependency matrix (rows correspond to probes, columns correspond to n system components) and X corresponds to the $n \times T$ delay matrix containing reconstructed delays at each component and at each time point (rows correspond to system components, columns correspond to the time points).

More specifically, in order to find the matrices A and X we have to solve a constrained optimization problem that minimizes the reconstruction error between Y and $\hat{Y}$ where $\hat{Y} = A \cdot X$. There are several choices of loss functions to minimize the error, for example the squared error or the KL-divergence. The optimization is constrained since both delays and dependency matrices have specific properties. For our application we impose the constraints of non-negativity since link delays are clearly non-negative and additive. Matrix factorization with the non-negativity constraint is called Non-negative matrix factorization (NMF). In addition to non-negativity, we also require A and X to be sparse. For A, sparsity is imposed on each row indicating that each probe goes through a few nodes[3]. For X, sparsity is imposed on each column indicating that at each timepoint the number of simultaneous bottlenecks causing the delay is typically small.

There exist other approaches to solve the BSS problem like ICA and PCA/SVD. However, since these approaches may result into negative values in reconstructed matrices, interpretation of results is less clear (if not impossible), especially for dependency matrix. Our approach requires nonnegativity and uses an appropriate NMF algorithm, and then does some postprocessing to transform the real-valued solution A into binary dependency matrix.

We used the following sparse NMF problem formulation and algorithm[4] proposed by Hoyer [4] (called H-NMF herein):

$$\min_{A,X} \quad \left\| Y - \hat{Y} \right\|_F^2$$
$$\text{subject to} \quad \text{sparsity}(\mathbf{a}) = s_A \qquad (1)$$
$$\text{sparsity}(\mathbf{x}) = s_X$$

where $\| \cdot \|_F$ is the Frobenius norm and sparsity($\cdot$) takes the following form:

$$\text{sparsity}(\mathbf{u}) = \frac{1}{\sqrt{d} - 1} \left(\sqrt{d} - \frac{\sum |u_i|}{\sqrt{\sum u_i^2}} \right) \qquad (2)$$

where the vector $\mathbf{u}$ has d dimensions. The values s_A and s_X are user-defined. The above notion of sparsity varies smoothly between 0 (indicating minimum sparsity) and 1 (indicating maximum sparsity). It exploits the relation between L_1 and L_2 norms thus giving great flexibility to achieve desired sparse solutions. This is a marked contrast from previous NMF algorithms that impose sparsity by adding either the L_1 norm or the L_2 norm as a regularization term to the objective function. The algorithm can be explained by a two step process: in each iteration, matrices A and X are first updated by taking a step in the direction of the negative gradient and then each row vector of A and column vector of X is (non-linearly) projected onto a non-negative vector with desired sparsity.

---

[2] An interesting direction for future work is combining our method with a recently proposed approach of [8] that recovers topology given the information of which subsets of components belong to each path (i.e. dependency matrix in our terminology).

[3] For example, a network probe often follows shortest route so the number of components it depends on is much smaller than the total number of components.

[4] We also experimented with an alternative sparse NMF algorithm proposed by Cichocki et al. [3] (C-NMF), but since the results produced by the two sparse NMF methods were quite similar, our discussion will focus only on H-NMF.

4. EXPERIMENTS

We have performed extensive experiments with sparse NMF algorithms both on simulated and real network topologies, and used both simulated and real traffic data.

4.1 Simulated traffic

We simulated network traffic on two large network topologies: one real (Gnutella) and one simulated (INET)[5]. However, due to space limitations, we only present the results for the real topology, i.e. for the snapshot of the *Gnutella* network (maintained by Limewire.org) that contained 127 nodes[6]. See [2] for an extended version of this paper that contains empirical results not included herein.

Given the topology, dependency matrix was constructed as follows. Assuming that the columns (components) correspond to network nodes and rows correspond to the end-to-end probes: the shortest path between each pair of nodes was considered as a potential probe[7], and a subset of probes was selected using the greedy information-gain-based approach [9] that ensures uniqueness of each column and thus identifiability of a single-node fault or bottleneck in the absence of noise[8]. The resulting matrix contained 50 rows (probes) and 127 columns (nodes).

4.1.1 Delay Simulator

We simulated end-to-end delay data using the dependency matrices constructed above. Herein, A_G and X_G will denote the ground truth matrices for their respective estimated counterparts. Given the dependency matrix A_G, the following procedure is used to produce the node delays X_G and the corresponding end-to-end delays Y. We generate a set of random (nonoverlapping) intervals $\{t_1, t_2, \ldots t_m\}$ within $[1, T]$ that correspond to periods when performance degradations occur in the network. For each interval t_i, k random nodes from $\{1 \ldots n\}$ are selected as current bottlenecks. The actual duration of each bottleneck is selected as another random subinterval of t_i, while the delay values for each bottleneck at each time point within the corresponding time interval are drawn randomly from $[200, 250]$. The rest of the entries in X_G, corresponding to "low" delays (absence of a bottleneck) are just set to zero. Finally, the end-to-end delays Y are obtained by adding linear Gaussian noise with mean 0, standard deviation σ and a scaling factor of 200, on top of the linear combination $A_G \cdot X_G$ of the node delays.

4.1.2 Evaluation Objectives

Typical evaluation of matrix-factorization algorithms focuses on reconstruction error between Y and $\hat{Y}$, but our objective is different as we want to actually reconstruct both the dependency matrix A and the node delay matrix X. Clearly, it is impossible to identify actual components (nodes or links) from the end-to-end observations only, and thus the reconstructed matrices A and X will correspond to some (unknown) permutation of the columns of A_G and the corresponding rows of X_G, respectively[9].

Also, since both sparse NMF algorithms first normalize the data matrices, dividing them first by their largest element, the reconstructed delay nodes will estimate the true ones up to a constant factor. Finally, the evaluation of the reconstruction quality must be different for A and X, since A will be binarized appropriately and interpreted as a dependency matrix, while X remains real-valued delay matrix.

For real-valued X we will use correlation with the ground truth delays as an evaluation measure. First we must establish a mapping between the actual nodes (rows in X_G) and rows in X, by finding the "best match": namely, each row $\mathbf{x}_G$ in X_G that ever contained a bottleneck will be matched with a row $\mathbf{x}$ in X that has maximum correlation with $\mathbf{x}_G$ (in other words, this node is identified as the bottleneck that is closest in terms of correlation to the true bottleneck from X_G) In our experiments, we report the average correlation over all such matches, i.e. for all bottlenecks in $\mathbf{x}_G$, which provides a measure of reconstruction quality for the node delay matrix.

As mentioned above, the evaluation criteria for the dependency matrix has to be different since the mixing-weight, real-valued matrix A obtained by BSS will be converted to a binary dependency matrix, using suitable postprocessing. Herein, we simply set the threshold to the mean of the minimum and maximum of the values of A, which provides an intuitive threshold point, especially if the distribution of the values in A is bimodal, which we actually observed in our experiments. After A is binarized, we reorder its columns according to the mapping found above for the actual nodes and their best matches in delay matrix. Now we can compute for each column of A the corresponding reconstruction error as an average number of 0/1 flips (mistakes made) with respect to the ground truth dependency matrix, and average the result over all columns. In the experiments, we will actually report the accuracy of the reconstruction which is $(1 - \text{error})$.

4.1.3 Results

As described above, we measured two types of reconstruction quality: average correlation for delay matrix X, and reconstruction accuracy for dependency matrix obtained by binarization of A. We performed extensive experiments with both H-NMF and C-NMF algorithms, on both Gnutella and INET networks, evaluating the effects of various factors such as the number of bottlenecks k occurring within each performance degradation time interval, the noise level σ, and the sparsity parameters s_A and s_X for the matrices A and X, respectively. However, due to space limitations, we only present the results for H-NMF on Gnutella network, since C-NMF produced quite similar results in similar settings, and INET results were also quite similar to the Gnutella results (see [2] for details). For all the experiments, we set the number of performance degradation periods $m = 4$. In all figures below, we varied the level of noise and the two sparsity parameters along the x-axis, while plotting different curves for following numbers of bottlenecks $k = 1, 5, 10, 15, 20$. All the figures show results averaged over 20 runs.

Varying noise. First, we explored the effect of noise σ and the number of bottlenecks k on the reconstruction quality of both matrices. The noise was varied from $\sigma = 0.01$ to 0.51, while both sparsity parameters s_A for dependency matrix and s_X for node delay matrix were fixed at 0.5 for H-NMF. Figure 1a shows the correlation

---

[5]INET generator [14] simulates an arbitrary-size Internet-like topology at the Autonomous Systems level by enforcing a power-law node degree distribution that coincides with the one empirically observed for the Internet.

[6]The collection method focused on getting accurate snapshots of small portions of the network rather than attempting to crawl the entire network, so severe sampling biases were hopefully avoided.

[7]For Gnutella network only, we actually generated breadth-first search trees from a small number of randomly selected sources, instead of considering all-pairs shortest path.

[8]Note that optimal dependency matrix design, i.e. selection of the minimal subset of end-to-end probes over a given topology that would guarantee the unique diagnosis of a single failure is an NP-hard problem; however the greedy approach adopted herein was shown to work well in practice[9].

[9]Clearly, we can only hope to identify the components that "reveal" themselves, e.g. experience bottlenecks at some points.
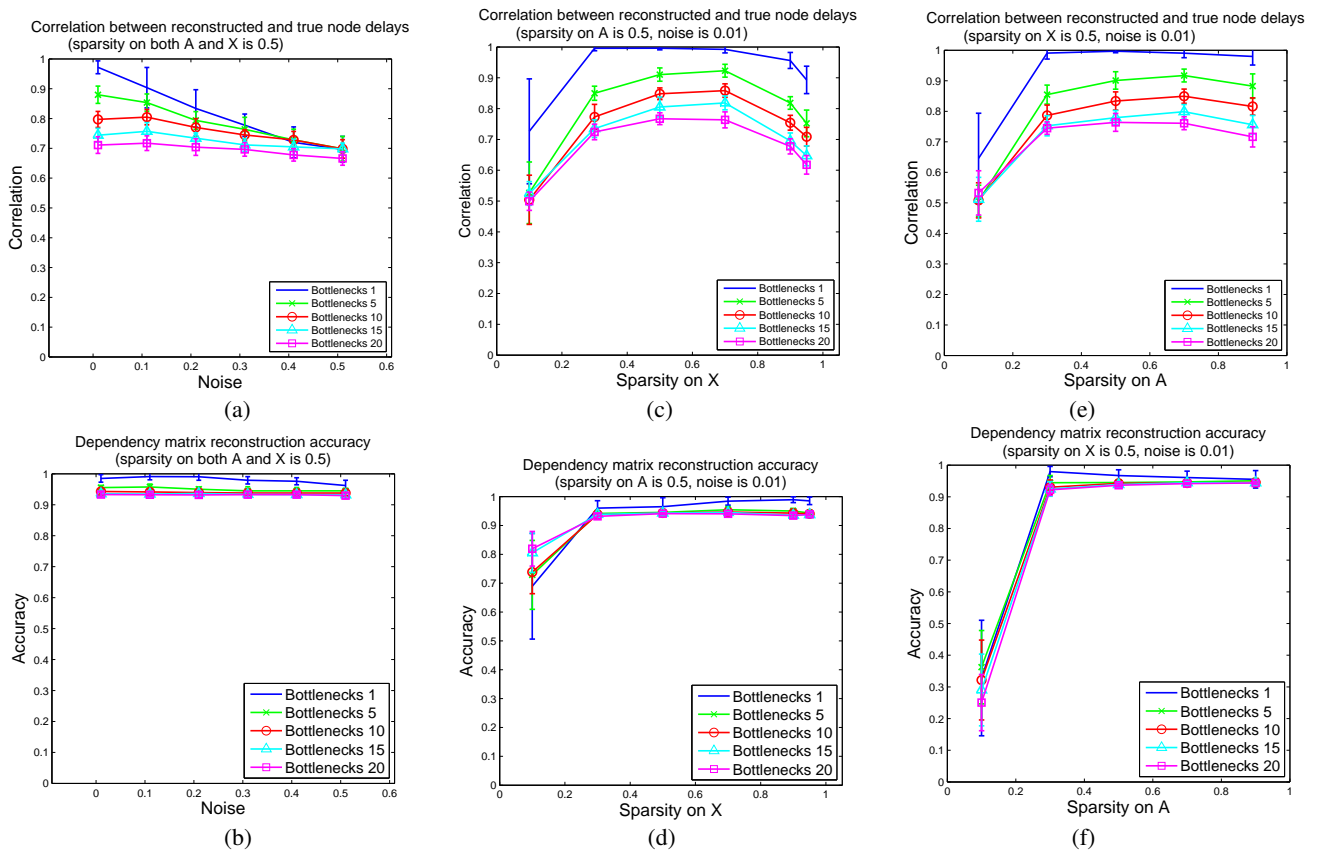
Figure 1: Results on Gnutella network.

results for reconstructed delay matrix on the Gnutella network. For low level of noise and small number of bottlenecks (which is a realistic assumption since it is not very typical to see many bottlenecks at once in real networks), the reconstruction of the delay matrix is pretty good: correlation is between 1 and 0.8 for the number of bottlenecks not exceeding 10, and approaches 1 for single bottleneck and noise less than $\sigma = 0.1$. As expected, the reconstruction quality decreases as the noise level increases, since the noise level gets closer to the signal level. Also, as expected, accurate reconstruction becomes more challenging as the number of bottlenecks increases. Figure 1b shows the corresponding results for the reconstruction accuracy of the dependency matrix that looks excellent: it appears to be much less sensitive to noise and remains within 0.9 to 1 even for larger number of bottlenecks, while for small number of bottlenecks it is practically 100% accurate. An intuitive explanation of such results versus the correlation measure is that accuracy measure only penalizes incorrect binary decisions about the dependency matrix values that result from thresholding, being more lenient than correlation measure about comparing real values.

Varying sparsity parameters. Next, we varied the parameter s_X that imposes sparsity on X. s_X was varied from 0.1 to 0.95 for H-NMF algorithm. We fixed the noise level σ to a low value of 0.01, leaving the parameter s_A imposing sparsity on A at the same value of 0.5 as before. Figures 1c and 1d show the results for the reconstruction of delay and dependency matrices respectively, suggesting that higher sparsity over delay matrix generally improves the reconstruction of both matrices, although for the delay matrix, it might be better to avoid extremely high values, e.g. for $s_X > 0.9$ performance starts to degrade. As before, delay matrix reconstruction is more sensitive to both sparsity and number of bottlenecks

than the dependency matrix reconstruction, and the accuracy results for dependency matrix are quickly getting to practically 100% for higher sparsities.

Finally, we performed the same set of experiments with varying the second sparsity parameter, s_A that imposes sparsity on A, while keeping fixed $s_X = 0.5$, and observed quite similar behavior in Figures 1e and 1f.

4.2 Real traffic: controlled experiment

We also evaluated our method on a real network traffic collected in a controlled environment. We used a small test lab containing six routers (Cisco and Juniper) and six links, where end-to-end delays were measured using probes such as SAA/rtr on Cisco routers and RPM on Juniper routers. Note that here we focus on reconstruction of link (rather than 'node') delays. Only four probes were necessary to ensure identifiability of a single link problem. The topology of the network, and the corresponding dependency matrix are shown in Figure 2a, where each probe name (e.g., 312) describes the path of that probe through the routers. We performed a controlled experiment, stressing a particular link (or a combination of links) with 30K 1400-byte ECHO_REQUEST packets in order to induce performance bottlenecks[10]. Particularly, we first induced a bottleneck on link 16 that only affected the two probes going through that link, 4216 and 4316 (Figure 2b shows the probe delay data averaged over one-minute intervals). Then, we stressed

---

[10]Note that in our controlled environment we can measure not only the end-to-end probe delays, but also all individual link delays, which is hard to do in real, uncontrolled network environments, and is a common challenge when evaluating approaches to link delay inference from end-to-end measurements.

simultaneously two links, 12 and 13, and observed performance degradation of probes 1249 and 312.

4.2.1 Results

We used the real data produced in our lab setting. Figures 2c-f show the reconstruction results for the link delays and the dependency matrix in our lab setting, as a function of the sparsity on the delay and dependency matrices in H-NMF algorithm, averaged over 50 runs. Particularly, Figure 2c shows for each link (each column in the true dependency matrix in Figure 2a) the correlation between the true link delay corresponding to row in the known true delay matrix X_G and its best-matching (most correlated) row in the reconstructed delay matrix X, as a function of varying H-NMF sparsity parameter on X; Figure 2d shows the accuracy (as measured before) of reconstructing the dependency-matrix column corresponding to that link. Similarly, Figures 2e and 2f, show the correlation and accuracy results as functions of varying sparsity on the dependency matrix A. Note that the highest correlation (up to 0.65) is observed for the two links (12 and 13) that were stressed (and thus "revealed" themselves) for a prolonged period of time; the link 16 which was also stressed but just for a short time period comes next achieving the correlation of about 0.3. The remaining unstressed links have lower correlations; note that increasing sparsity parameters sometimes improves and sometimes hurts the correlation, depending on a particular link. It is interesting to note that the highest correlation between the true and reconstructed delay (e.g., for links 12 and 13) does not necessarily imply the best reconstruction accuracy in dependency matrix. Indeed, the most accurate dependency reconstruction (around 80%) is achieved for the link 49 corresponding to the last column in dependency matrix; note that this link was not stressed and thus did not had a chance to "reveal" itself as a bottleneck, however, it appears on the path of a single probe 1249 which may simplify its reconstruction. Interestingly, the reconstruction accuracy does not change much with varying the sparsity parameters. The next best dependency reconstruction results are achieved for the bottleneck links 13 and 16, as well as for 34 (also on the path of only one probe); for those links, accuracy does change with varying sparsity, but the effects of sparsity can be opposite for those links (higher sparsity on both X and A helps 34 but hurts 13 and 16).

Since the accuracy and especially correlation values were less impressive in real setting as compared with the Gnutella simulations, we decided to perform the simulated traffic experiment as well, in order to better understand the sources of inaccuracies. (see Figure 3). While simulations results, as expected, are somewhat more accurate then the real experiment, they are still much less accurate (both in terms of delay correlation and dependency matrix reconstruction accuracy) than the corresponding results for a much larger Gnutella network. This effect may be particularly due the ratio between the number of bottlenecks and the network size, which yields a much higher sparsity in large networks; on the contrary, just two bottlenecks in our lab environment already constitute $1/3$ of all links. We conjecture that our method works better in larger networks with a small number of bottlenecks. However, other properties of both "ground truth" dependency and link delay matrices may also affect the reconstruction quality and require further investigation.

5. CONCLUSIONS AND OPEN ISSUES

This paper proposes a novel approach to the challenging problem of the network bottleneck/delay diagnosis in the absence of dependency/routing information. On the Gnutella an INET topologies, we can conclude that reconstruction quality of our approach was quite impressive and exceeded our expectations. We learned that using nonnegative constraint on BSS approach, with sufficiently high sparsity imposed on both dependency and node delay matrices, is important for obtaining an accurate reconstruction. Clearly, results deteriorate with increasing noise which might be an issue in realistic scenarios. Preliminary results on real versus simulated traffic in small lab settings were somewhat less impressive; however, since even the simulation results in the same setting were much less accurate than for larger networks, we conjecture that indication that our approach should be more applicable in larger networks with reasonably small number of (more clearly "pronounced") bottlenecks. Further investigation of our approach on real traffic in large-scale networks is the direction of our ongoing work. Another interesting directions for future work include investigation of identifiability and uniqueness conditions for the proposed approaches in practical scenarios, and applying this approach to other end-to-end measures, besides the delays, such as jitter (variability), which is particularly important in VoIP networks. Finally, we plan to extend our approach to a "semi-blind" source separation that can incorporate some partial dependency (routing) information.

6. REFERENCES

[1] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu. Network Tomography: Recent Developments. *Statistical Science*, 19(3):499–517, 2004.

[2] G. Chandalia and I. Rish. Blind source separation approach in distributed systems management. *IBM Technical Report*, 2007.

[3] A. Cichocki, R. Zdunek, and S. Amari. New algorithms for non-negative matrix factorization in applications to blind source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 5, pages V–621–V–624, 2006.

[4] Patrik O. Hoyer. Non-negative matrix factorization with sparseness constraints. *J. of Machine Learning Research*, 5:1457–1469, 2004.

[5] S. Kliger, S. Yemini, Y. Yemini, D. Ohsie, and S. Stolfo. A coding approach to event correlation. In *Proc. 4th International Symposium on Integrated Network Management (IFIP/IEEE), Santa Barbara, CA, USA*, pages 266–277, 1997.

[6] F. LoPresti, N. G. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal delay distributions. *IEEE/ACM Transactions on Networking*, 10:761–775, 2002.

[7] A. Medina, N. Taft, S. Battacharya, C. Diot, and K. Salamatian. Traffic matrix estimation: Existing techniques and new directions. In *ACM SIGCOMM-02*, 2002.

[8] M. Rabbat, M. Figueiredo, and R. Nowak. Inferring network structure from co-occurrences. In *Neural Information Processing Systems (NIPS'06)*, 2006.

[9] I. Rish, M. Brodie, S. Ma, N. Odintsova, A. Beygelzimer, G. Grabarnik, and K. Hernandez. Adaptive diagnosis in distributed systems. *IEEE Transactions on Neural Networks (special issue on Adaptive Learning Systems in Communication Networks)*, 16(5):1088–1109, 2005.

[10] H. Song, L. Qiu, and Y. Zhang. NetQuest: A Flexible Framework for Large-Scale Network Measurement. In *ACM SIGMETRICS*, 2006.

[11] H. Tian and H. Shen. Multicast-based inference for topology and network-internal loss performance from end-to-end measurements. *Computer Communications*, 11(29):1936–1947, 2006.

[12] Y. Tsang, M.J. Coates, and R. Nowak. Network delay tomography. *IEEE Trans. Signal Process.*, 51:2125–2136, 2003.

[13] Y. Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *J. Amer. Statist. Assoc.*, 91:365–377, 1996.

[14] J. Winick and S. Jamin. Inet-3.0: Internet topology generator. Technical Report CSE-TR-456-02, University of Michigan, 2002.

[15] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. In *ACM SIGMETRICS-03*, 2003.

[16] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *ACM SIGCOMM-03*, 2003.
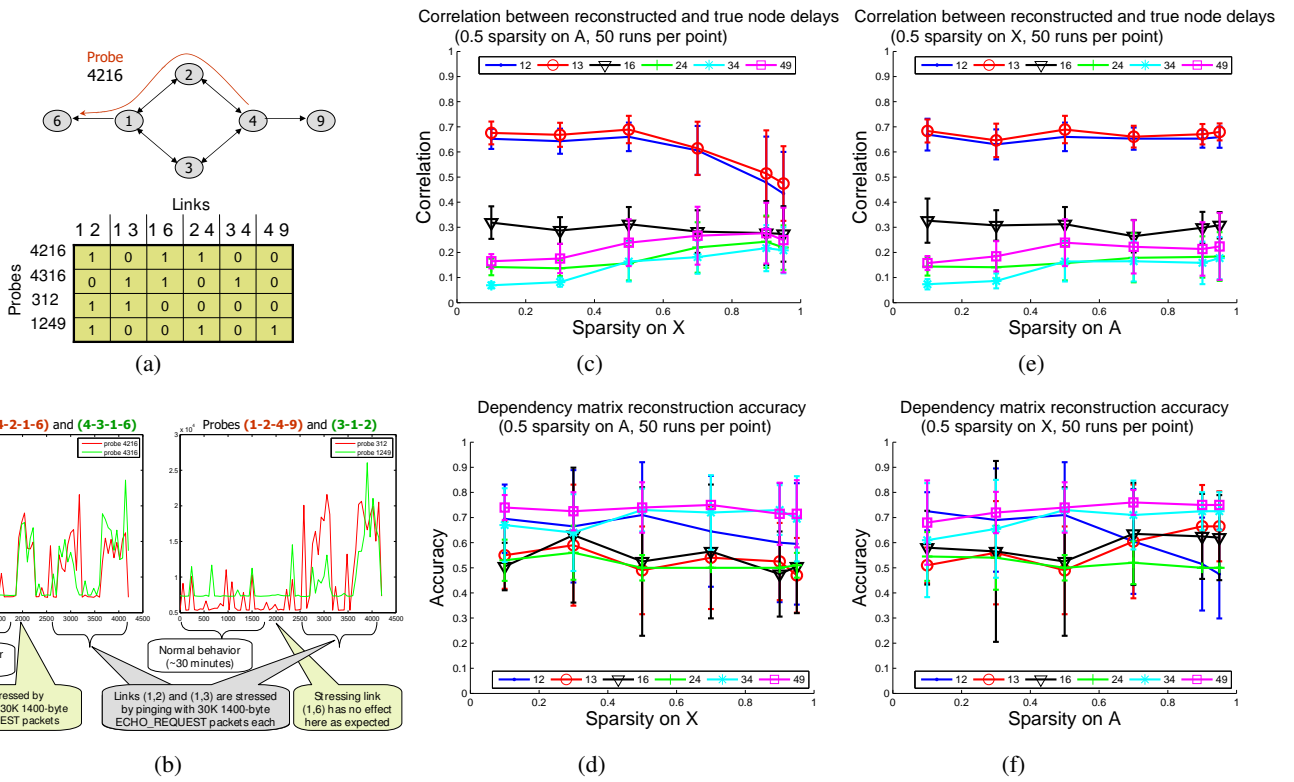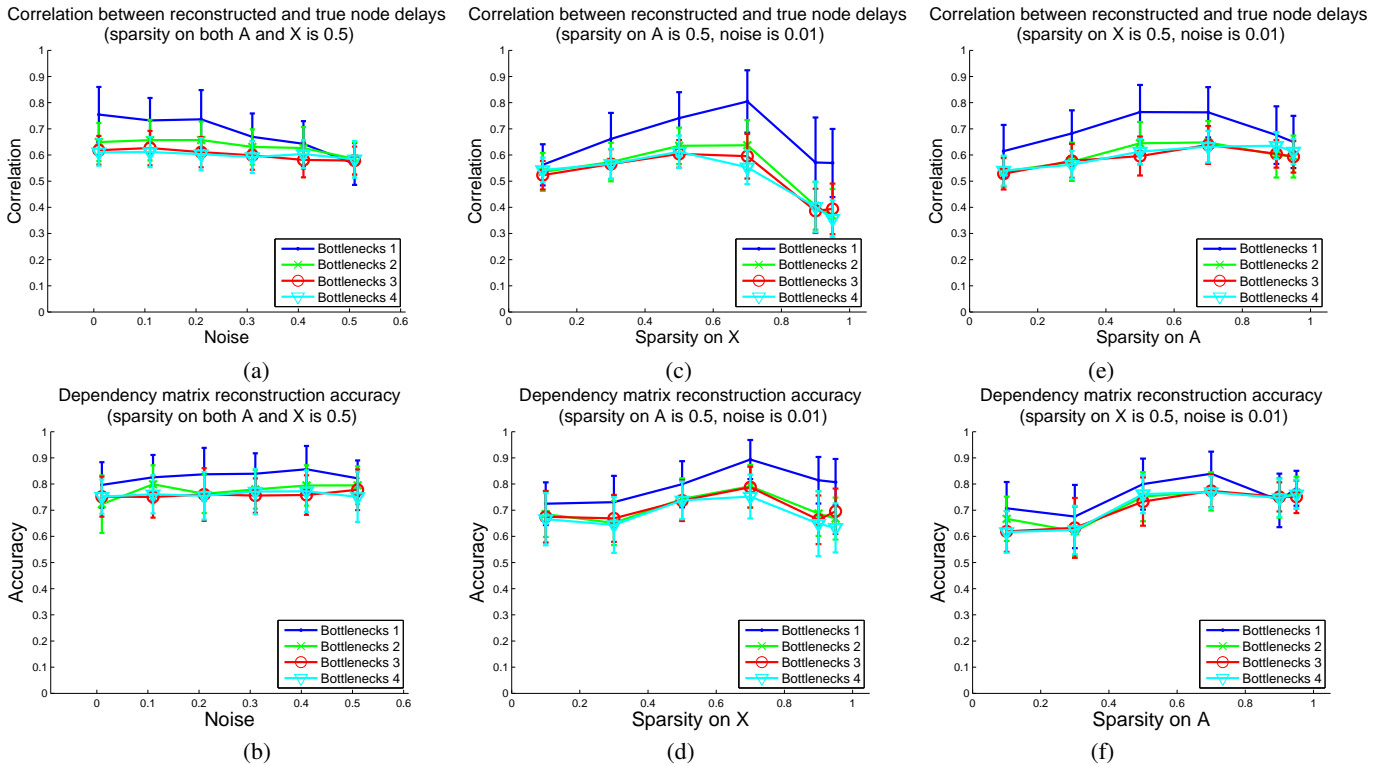
Figure 2: Results in the real lab environment.

Figure 3: Results for simulated traffic on the lab network from Figure 2a.