

DISSERTATION PROPOSAL

Large-Scale Network Measurement and Analysis

by

Han Hee Song

Committee:

Yin Zhang, Supervisor	Dept. of CS, UT Austin
James Browne	Dept. of CS, UT Austin
Inderjit Dhillon	Dept. of CS, UT Austin
Lili Qiu	Dept. of CS, UT Austin
Jia Wang	AT&T Labs - Research

THE UNIVERSITY OF TEXAS AT AUSTIN

11th August 2010

Large-Scale Network Measurement and Analysis

Han Hee Song

Supervisor: Yin Zhang

Scalable measurement and accurate analysis of networks is essential to a wide variety of existing and emerging network systems. Specifically, the network measurement and analysis helps to understand the networks, improve the existing services, and enable new applications to be developed. To effectively support various services and applications, the network measurement and analysis must address various challenges: (i) how to conduct scalable measurement in networks with large number of nodes and links, (ii) how to flexibly accommodate multiple design objectives and constraints, (iii) and how to cope with the dynamic changes in the networks.

This dissertation proposes novel network analysis schemes that effectively address the above challenges in analyzing different types of large-scale networks. In doing so, we make three major contributions:

First, we design and implement a framework for large-scale path performance measurement and analysis in large-scale IP networks. We propose approaches for exact as well as approximate reconstruction of network path properties.

Second, we develop techniques for capturing proximity between nodes in massive online social networks. The general and scalable techniques can estimate a wide range of the node proximity measures in near real-time.

Third, as ongoing work, a scheme for service quality assessment on a large commercial IPTV system is in development. In our approach, we aim to devise a video quality monitoring scheme in the IPTV and provide a simple yet comprehensive view of the end-to-end system structure.

Table of Contents

Abstract	ii
Chapter 1. Introduction	1
1.1 Challenges	2
1.2 Research	3
1.2.1 Measurement and Analysis of IP Network Path Properties	4
1.2.1.1 Exact Reconstruction of Network Path Properties	4
1.2.1.2 Approximate Reconstruction of Network Path Properties	5
1.2.2 Proximity Estimation in Online Social Network	5
1.2.2.1 Proximity Embedding	6
1.2.2.2 Clustered Graph Embedding	6
1.2.3 Assessment of Service Quality in IPTV Network	7
Chapter 2. Measurement and Analysis of IP Network Path Properties	8
2.1 Problem Formulation	8
2.2 Exact Reconstruction of Network Path Properties	11
2.2.1 Algebraic Model	11
2.2.2 Basic Static Algorithms	13
2.2.2.1 Measurement Paths Selection	13
2.2.2.2 Path Loss Rate Calculations	14
2.2.3 Scalability Analysis	14
2.3 Approximate Reconstruction of Network Path Properties	17
2.3.1 Design of Experiments	17
2.3.1.1 Bayesian Experimental Design	17
2.3.1.2 Flexibility	22
2.3.1.3 Non-Bayesian Designs	24
2.3.2 Inference Algorithms	26
2.3.2.1 L_2 Norm Minimization	26
2.3.2.2 L_1 Norm Minimization	27
2.3.2.3 Maximum Entropy Estimation	27

2.3.3	Evaluation	28
2.3.3.1	Evaluation Methodology	28
2.3.3.2	Experimental Parameters	29
2.3.3.3	Basic Framework Evaluation Results	30
2.4	Summary of IP network measurement and analysis	32
Chapter 3.	Proximity Estimation in Online Social Networks	33
3.1	Problem Formulation	34
3.2	Proximity Embedding	35
3.2.1	Preparation	35
3.2.2	Proximity Embedding	36
3.2.3	Evaluation	38
3.3	Clustered Graph Embedding	40
3.3.1	Clustered Graph Embedding	40
3.3.2	Extensions	42
Chapter 4.	Assessment of service quality in IPTV networks	44
Bibliography		46

Chapter 1

Introduction

Scalable measurement and accurate analysis of networks is essential to a wide variety of existing and emerging network systems such as content distribution and file sharing (*e.g.*, CoDeeN [68] and BitTorrent [1]), traffic engineering (*e.g.*, for supporting IP Telephony [3]), overlay networks (*e.g.*, RON [4] and OpenDHT [57]), interactive media streaming systems (*e.g.*, IPTV [7], VoIP [60], and video conference systems [2]), and online social networks (*e.g.*, Facebook [28] and MySpace [52]). Specifically, for the networks, the network measurement and analysis can help in different ways: (i) understanding the network, (ii) improving the service being provided, (iii) and enabling new applications.

Understanding the network. The increasing complexity of large-scale ISPs and enterprise networks makes it difficult to understand their network architecture, especially the dynamic changes of link and path performances represented as delay, loss-rate, bandwidth, and etc. The holistic view of path performance measurement in IP networks can help the network operations to efficiently manage their network resources. For example, content distribution applications can select servers with low latency to clients to reduce content transfer time.

Improving the service being provided. For emerging commercial media streaming services with high requirements in quality of service (QoS) and quality of experience (QoE), end-to-end service quality assessment can help accommodate the system capacity and troubleshoot performance problems. Comprehensive service level monitoring allows quality sensitive services to support the rapidly growing number of subscribers while maintaining reliable service.

Enabling new applications. Computational analysis of massive online social networks (OSNs) quantifying the proximity (*e.g.*, closeness or similarity) between nodes allows analysts to perceive the basic social structure of the network. This proximity estimation in OSNs forms the basis for enabling a wide range of important new applications in social and natural sciences (*e.g.*, modeling complex networks [38, 8, 20, 53]), business (*e.g.*, viral marketing [37], fraud detection [18]), and information technology (*e.g.*, improving Internet search [50], and collaborative filtering [9]).

1.1 Challenges

Large-scale network analysis creates significant new research challenges and opportunities. To effectively support various services and applications, the network measurement and analysis must address the following challenges:

- **Scalability in network measurement space**

Large-scale network management applications often require the ability to efficiently monitor the whole network. As networks evolve to gather more users and subscribers and support more functionalities, the size of networks quickly increases over time. In a network with $O(N)$ nodes where N is the number of nodes in the network, the number of paths interconnecting the nodes can be as many as $O(N^2)$. The quadratic growth in the number of network paths with respect to the number of network nodes makes it impractical to measure every path.

The brute force way of simultaneously measuring all network paths — from each end node to all other end nodes, in a large networked system — can lead to inaccurate measurement results and pose an enormous load on node and network resources. A naive scheduling solution that schedules few measurements at a time on each node can reduce the loads on node and network and avoid interference with other processes and network flows. The increased measurement time, however, can result in stale, inaccurate measurement results.

- **Flexibility in accommodating multiple design objectives and constraints**

Network measurement requires carefully designed experiments. The design space is often large, involving a number of control variables. Control variables in network measurement and analysis include: choosing the sites to launch experiments from, choosing the type of network characteristics to measure, choosing how to randomize, choosing the granularity, frequency, and duration of each experiment, etc.

In addition to the various design choices, there exist multiple design objectives. Different applications (*e.g.*, VoIP gateway services, versus cable access network providers) often have different notions of performance, and want to analyze different metrics (*e.g.*, delay or loss in the case of IP networks, and VoIP quality or video quality in IPTV service networks).

Lastly, the design is often subject to all kinds of constraints imposed by the network and operations, such as the accessibility of measurement infrastructure, the availability of network resources, and operational policies and restrictions.

- **Robustness to the dynamic nature of network**

Today's network environments are highly dynamic: path performance varies over time,

links flap, and new nodes are continually added and deleted. In order to keep track of the changing conditions in real-time, the network measurement and analysis should work in a timely fashion. Specifically, the measurement of performance metric should offer fast adaptation to topology changes and balanced distribution of loads to measurement hosts. Likewise, the network analysis, when processing the measured data, should incorporate efficient computation of measured data such as partial, incremental updates to overcome the overhead of computing the entire network snapshot from scratch.

1.2 Research

This dissertation proposes novel network analysis schemes that effectively address the above challenges in analyzing different types of large-scale networks. In doing so, we consider following three analyses conducted on IP networks, online social networks, and IPTV service networks, respectively:

- **Measurement and analysis of IP network path properties**

We design and implement a framework for large-scale path performance measurement and analysis in IP networks. Here, we consider two approaches: (i) exact reconstruction of network path properties and (ii) approximate reconstruction of path properties. In the former, we approach the problem of selecting a set of monitored paths algebraically to reconstruct the *exact* properties of all the paths comprising a network. In the latter, we develop a measurement scheme based on Bayesian experimental design, which provides better scalability with the cost of minimal accuracy sacrifice. In addition, the approach flexibly supports a variety of design requirements.

- **Proximity estimation in online social network**

We propose techniques for measurement and analysis of social structures by quantifying the proximity between nodes in massive online social networks. The first technique is able to estimate a limited sub-family of node proximity measures in social networks with millions of nodes. As a new phase of research, we are developing a more general and scalable technique, that allows a wider range of node proximity measures to be approximated in near real-time on massive social networks. As a concrete application of such techniques, a new link prediction scheme is planned to significantly improve the scalability and accuracy in predicting the generation of network edges.

- **Assessment of service quality in IPTV network**

Also as ongoing work, a scheme for service quality assessment on a large commercial IPTV system is in development. In our approach, we aim to devise a video quality

monitoring scheme in the IPTV that provides a simple yet comprehensive view of the end-to-end system structure.

In the following subsections, we provide detailed introductions of each analysis, beginning from the measurement of path properties in IP networks.

1.2.1 Measurement and Analysis of IP Network Path Properties

We address two steps towards the development of scalable and flexible network measurement. First, in Exact Reconstruction of Network Path Properties, we devise a network monitoring system that only requires a small subset of paths in order to rebuild the *exact* properties of all the paths in the network. For an overlay network of N nodes, the linear algebraic approach we develop bounds the number of measurements to $O(N \log N)$ while the existing *general metric* systems [5] require $O(N^2)$ measurements.

Once we set up an upper-bound of the monitoring cost this way, in Approximate Reconstruction of Network Path Properties, we present a framework that provides better scalability to conduct experiments that span thousands of routers and end hosts. The framework also provides better flexibility to accommodate different design requirements of different users.

1.2.1.1 Exact Reconstruction of Network Path Properties

In this work [14], we develop a scalable overlay loss rate monitoring system which provides the best accuracy achievable by selecting a minimal subset of paths to monitor so that the loss rates and latencies of all other paths can be inferred.

In our proposed method, we selectively monitor a *basis set* of k paths. Any end-to-end path can be written as a unique linear combination of paths in the basis set. Consequently, by monitoring loss rates for the paths in the basis set, we infer loss rates for all end-to-end paths. The end-to-end path loss rates can be computed even when the paths contain *unidentifiable links* for which loss rates cannot be computed.

Given the measurements for paths corresponding to the basis set, this linear algebraic scheme can reconstruct the end-to-end performance on all paths *exactly*. Moreover, we show that the size of the basis set(k) grows as $O(N \log N)$ through linear regression tests.

1.2.1.2 Approximate Reconstruction of Network Path Properties

To further deal with the problem of scalability and flexibility, we develop NetQuest [62], a flexible measurement framework that can support large-scale continuous network monitoring. NetQuest consists of two key components: *design of experiments* and *network inference*.

We apply Bayesian experimental design to determine the set of active measurements that maximize the amount of information we gain about the network path properties subject to given resource constraints (*e.g.*, probing overhead). Bayesian experimental design is built on solid theoretical foundations, and has found numerous applications in scientific research and practical applications, ranging from software testing to medicine, to biology, and to car crash test. Recognizing its potential, we bring Bayesian experimental design into large-scale network measurement.

To address the above issues, we first formulate the problem under the Bayesian experimental design framework. We then explore a series of Bayesian design schemes, and use extensive evaluation to identify the design scheme best suited for network monitoring. In addition, we develop techniques to achieve flexibility by designing measurement experiments that maximize the information gain for different design objectives and constraints. In particular, our approach can support the following requirements: (i) *differentiated design* for providing better resolution to certain parts of the network, (ii) *augmented design* for conducting additional measurements given existing observations, and (iii) *joint design* for supporting multiple users interested in different parts of the network.

1.2.2 Proximity Estimation in Online Social Network

A social network[69] is a social structure modeled as a graph, where nodes represent people or other entities embedded in a social context, and edges represent specific types of interdependency among entities, *e.g.*, values, visions, ideas, financial exchange, friendship, kinship, dislike, conflict or trade.

A central concept in the computational analysis of social networks is *proximity measure*, which quantifies the closeness or similarity between nodes in a social network. Traditionally, studies on social networks often focus on relatively small social networks (*e.g.*, [42, 43] examine co-authorship networks with about 5000 nodes). Unfortunately, the explosive growth of online social networks imposes difficulties on applying the traditional proximity estimation to OSNs with millions of nodes and tens of millions of edges. We propose two steps towards scalable social network analysis: *Proximity Embedding* as accomplished work and *Clustered Graph Embedding* as ongoing work.

1.2.2.1 Proximity Embedding

As a first step, we made some initial progress in *proximity embedding* on approximating a restricted sub-family of path ensemble based proximity measures which involve infinite sums over the ensemble of all paths between two users in social graph structures(*e.g.*, Katz measure [40]).

Compared with more direct proximity measures such as shortest graph distances and numbers of shared neighbors, path-ensemble based proximity measures incorporate more information about the underlying social structure and have been shown to be more effective in social networks [42, 43, 67].

Despite the effectiveness of path-ensemble based proximity measures, it is computationally expensive to summarize the ensemble of all paths between two nodes. The state of the art in estimating path-ensemble based proximity measures (*e.g.*, [67]) typically can only handle social networks with tens of thousands of nodes. Our proximity embedding technique applies matrix factorization to approximate proximity matrix as the product of two lower-rank factor matrices, which can handle massive online social networks with millions of nodes.

1.2.2.2 Clustered Graph Embedding

Despite significant progress in *proximity embedding*, *proximity embedding* can only efficiently estimate proximity when the underlying matrix of interest is extremely low-rank. For example, the proximity estimation techniques in *proximity embedding* use factor matrices with 60 columns. Unfortunately, given the massive size of online social networks, proximity measures requiring higher dimensionality cannot be approximated accurately by the existing techniques.

We propose to develop a novel technique called *clustered graph embedding* to significantly improve the generality, accuracy, scalability, and flexibility of social network analysis.

- **Proximity estimation via clustered embedding**

We plan to develop a dimensionality technique combining graph clustering and spectral graph embedding in order to embed the originally massive but sparse social graph into a much smaller but dense graph. The embedded graph can capture the fundamental structure of the original social graph and allows a wide range of computational analysis tasks to be performed on the embedded graph instead of the original graph. By using the smaller embedded graph, we expect to improve the approximation accuracy by accommodating many of the not very low-rank proximity measures while achieving near real-time computation.

- **Supervised link prediction**

To demonstrate the practical values of our techniques, we plan to consider a significant application of proximity estimation: link prediction, *i.e.*, predicting which user pairs will become new friends based on past snapshots of a social network. The finding will show the improvement of link prediction accuracy of the derived new proximity measure compared to the existing measures.

1.2.3 Assessment of Service Quality in IPTV Network

Although a relatively new technology, Internet Protocol Television (IPTV) is being successfully rolled out within numerous large-scale commercial deployments across the globe [56]. IPTV offers exciting new opportunities that go well beyond traditional TV, such as new services made available through the TV interface [11]. However, in an already competitive market, high service quality and strong customer satisfaction are imperative for commercial success. Thus, IPTV service providers are continually striving for further service improvements in their IPTV deployments.

Conventional network management schemes generally target to monitor the health of individual network element, within the *network* part of a system, for specific set of network performance measures. The performance reports and alerts generated for each and every network component, however, may not be reflecting the actual application service problems at the user-end. In the case of IPTV, the audio and video quality at the customers is not likely to be directly correlated to the CPU utilization at the individual router level.

Additionally, given the growing size of the network, the management overhead for investigating the overwhelming amount of individual alerts is too high. In IPTV systems where there are tens of thousands of routers in the core IP network and order of millions of set-top boxes at residential homes, going through individual alert looking for correlation with user-perceived service quality certainly raises scalability concerns.

In commercial-scale network services such as IPTV, a service management is required to have a much wider view than just the network part of the service. The service assurance needs to encompass the entire end-to-end system beginning from the generation of video stream all the way to the set-top boxes at the customer side. To fulfill the need, our proposed network service management scheme aims to combine the independent low-level system reports from the underlying network components into manageable amount of comprehensive indicator of service quality.

Chapter 2

Measurement and Analysis of IP Network Path Properties

In this section, we focus on monitoring end-to-end path performance in large IP networks. The quantity of interest is a function of the performance on individual links, which may not be directly observable either because those links may belong to a non-cooperative administrative domain, or because full instrumentation of an IP network is considered cost prohibitive. Large-scale network measurement is challenging because the number of paths increases quadratically with the number of nodes, and it is often impractical to probe all the network paths, yet the final quantity of interest may depend on links on all the paths. The goal is then to conduct a small number of active measurements, and infer the quantity of interest based on partial and indirect observations. The problem consists of two key aspects: (i) *design of measurement experiments*, and (ii) *network inference* (also commonly referred to as *network tomography*).

2.1 Problem Formulation

Symbols	Meanings
P	set of all network paths
S	set of subset of P ($S \subseteq P$)
L	set of links on paths in P
m	number of end-to-end paths ($m = P $)
n	number of IP links ($n = L $)
$A \in \{0, 1\}^{m \times n}$	original routing matrix
$A_S \in \{0, 1\}^{S \times n}$	reduced routing matrix
$\mathbf{x} \in \mathbb{R}^n$	vector of unknown performance on individual links
$\mathbf{y} \in \mathbb{R}^m$	vector of observed end-to-end path performance
$\mathbf{y}_S \in \mathbb{R}^S$	reduced vector of observed path performance
v	vector in $\{0, 1\}^m$ (represents path)

Table 2.1: Table of notations

Formally, the problem can be specified as follows.

$$\mathbf{y} = A\mathbf{x}, \tag{2.1}$$

where \mathbf{x} is the vector of some unknown quantity, \mathbf{y} is the vector of observables, A is a matrix that associates \mathbf{y} and \mathbf{x} (often referred to as the *routing matrix*). In the context of network performance monitoring, \mathbf{x} is the vector of unknown performance on individual links, \mathbf{y} is the vector of observed performance on a set of end-to-end paths, and the routing matrix $A = (A_{ij})$ encodes whether link j belongs to path i , *i.e.*,

$$A_{ij} = \begin{cases} 1 & \text{if path } i \text{ contains link } j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

Note that our definition of routing matrix applies to both one-way and round-trip performance measurements. For round-trip measurements, the routing matrix can work for asymmetric routes.

The above formulation applies to any additive performance metric, such as delay or $\log\{1 - \text{loss rate}\}$. In the case of representing loss rate to additive metric, we first represent a path by $v \in \{0, 1\}^m$, where the j th entry v_j is one if link j is part of the path, and zero otherwise. Suppose link j drops packets with probability l_j ; then the loss rate p of a path represented by v is given by

$$1 - p = \prod_{j=1}^n (1 - l_j)^{v_j} \tag{2.3}$$

We take logarithms on both sides of (2.3). Then by \mathbf{x} with elements $\mathbf{x}_j = \log(1 - l_j)$, we can rewrite (2.3) as follows:

$$\log(1 - p) = \sum_{j=1}^n \mathbf{y}_j \log(1 - l_j) = \sum_{j=1}^n \mathbf{y}_j \mathbf{x}_j = \mathbf{y}^T \mathbf{x} \tag{2.4}$$

Let p_i be the end-to-end loss rate of the i th path, and let $\mathbf{y} \in \mathbb{R}^m$ be a column vector with elements $\mathbf{y}_i = \log(1 - p_i)$. Then we can obtain the m equations in form (2.4) as we did in 2.1. Therefore we can represent both delay and loss rate in the same form.

Besides performance estimation, there is another type of tomography problem, commonly referred to as *traffic matrix estimation*, which tries to infer end-to-end traffic demands based on observed link loads. In this context, \mathbf{x} is the vector of unknown traffic demands, \mathbf{y} is the vector of observed link loads. There has been considerable recent progress on traffic

matrix estimation [48, 73, 74]. In this work, we only consider network performance inference, but the framework and the techniques we develop can also be applied to traffic matrix estimation. We plan to explore this direction in our future research.

Our goal is to estimate $f(\mathbf{x})$, which is a function of link properties \mathbf{x} . One interesting example is $f(\mathbf{x}) = A\mathbf{x}$. In this case, the quantity of interest $f(\mathbf{x})$ represents properties of all network paths. More specifically, when \mathbf{x} is link delay, $f(\mathbf{x})$ is delay on all network paths. Another example is $f(\mathbf{x}) = \frac{1}{m}[1, 1, \dots, 1]_{1 \times m}A\mathbf{x}$, which corresponds to a network-wide average metric (*e.g.*, $f(\mathbf{x})$ is the network-wide average path delay, when \mathbf{x} is link delay).

In large-scale network measurement, it is too expensive to directly measure network properties on all paths. So the goal is to select only a small subset of the paths to probe so that we can still accurately estimate the quantity of interest. We formalize the path selection problem as follows.

Let P be the set of all network paths ($|P| = m$). Let L be the set of links appearing on paths in P ($|L| = n$). The performance on paths in P and the performance on links in L are related according to the linear system $\mathbf{y} = A\mathbf{x}$, where \mathbf{x} is a length- n column vector, \mathbf{y} is a length- m column vector, and A is a $m \times n$ routing matrix.

For any subset $S \subseteq P$ (with $s = |S|$), let A_S be the $s \times n$ sub-matrix of A formed by the s rows corresponding to those paths in S . Similarly, let \mathbf{y}_S be the sub-vector of \mathbf{y} corresponding to the observed performance on those paths in S . The experimental design problem is to select a subset of paths S to probe such that we can estimate $f(\mathbf{x})$ based on the observed performance \mathbf{y}_S , A_S , and thus the following linear system

$$\mathbf{y}_S = A_S\mathbf{x}, \tag{2.5}$$

In this work, we consider the case when $f(\mathbf{x})$ is a linear function

$$f(\mathbf{x}) = F\mathbf{x}, \tag{2.6}$$

where F is a given $r \times n$ matrix.

The other major aspect of network performance monitoring is network inference. Its goal is to infer \mathbf{x} based on A_S and \mathbf{y}_S . The major challenge in network inference is that the linear system $\mathbf{y}_S = A_S\mathbf{x}$ is often under-determined due to partial observations, and can thus have an infinite number of solutions.

2.2 Exact Reconstruction of Network Path Properties

In this section, we illustrate the analysis technique for exact reconstruction of path properties. In particular, we address its algebraic model, basic static algorithms, scalability analysis, and evaluation.

2.2.1 Algebraic Model

Symbols	Meanings
N	number of end hosts on the overlay
t	number of identifiable links
$k \leq n$	rank of A
l_i	loss rate on i th link
p_i	loss rate on i th measurement path
\mathbf{x}_i	$\log(1 - l_i)$
\mathbf{y}_i	$\log(1 - p_i)$
p	loss rate along a path
$\mathcal{N}(A)$	null space of A
$\mathcal{R}(A^T)$	row(path) space of A ($== \text{range}(A^T)$)

Table 2.2: Additional table of notations

In this section, we introduce an algebraic model of this work. We expand our notations as described in Table 2.2, in addition to Table 2.1.

Suppose an overlay network spans n IP links. Normally, the number of paths m is much larger than the number of links n (see Fig. 2.1(a)). This suggests that we could select n paths to monitor, use those measurements to compute the link loss rate variables \mathbf{x} , and infer the loss rates of the other paths from (2.1).

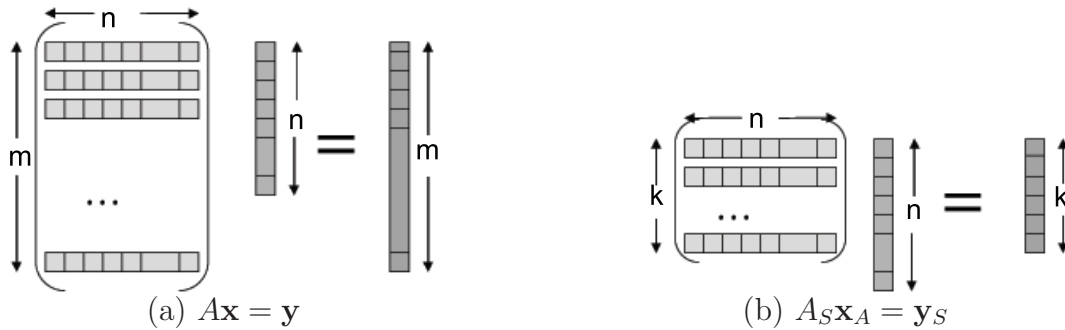


Figure 2.1: Matrix size representations.

However, as we discussed in Section 2.1, A is rank deficient: *i.e.*, $k = \text{rank}(A)$ and $k < n$. If A is rank deficient, we will be unable to determine the loss rate of some links from (2.1). These links are also called *unidentifiable* in network tomography literature [10].

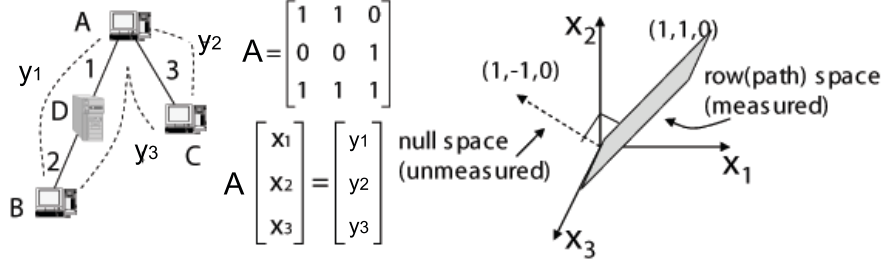


Figure 2.2: Sample overlay network.

Fig. 2.2 illustrates how rank deficiency can occur. There are three end hosts (A, B and C) on the overlay, three links (1, 2 and 3) and three paths between the end hosts. We cannot uniquely solve \mathbf{x}_1 and \mathbf{x}_2 because links 1 and 2 always appear together. We know their sum, but not their difference.

Fig. 2.2 illustrates the geometry of the linear system, with each variable \mathbf{x}_i as a dimension. The vectors $\{\alpha [1 \ -1 \ 0]^T\}$ comprise $\mathcal{N}(A)$, the *null space* of A . No information about the loss rates for these vectors is given by (2.1). Meanwhile, there is an orthogonal *row(path) space* of A , $\mathcal{R}(A^T)$, which for this example is a plane $\{\alpha [1 \ 1 \ 0]^T + \beta [0 \ 0 \ 1]^T\}$. Unlike the null space, the loss rate of any vector on the row space can be uniquely determined by (2.1).

To separate the identifiable and unidentifiable components of \mathbf{x} , we decompose \mathbf{x} into $\mathbf{x} = \mathbf{x}_A + \mathbf{x}_N$, where $\mathbf{x}_A \in \mathcal{R}(A^T)$ is its projection on the row space and $\mathbf{x}_N \in \mathcal{N}(A)$ is its projection on the null space (*i.e.*, $A\mathbf{x}_N = 0$). The decomposition of $[\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3]^T$ for the sample overlay is shown below.

$$\mathbf{x}_A = \frac{(\mathbf{x}_1 + \mathbf{x}_2)}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + \mathbf{x}_3 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1/2 \\ \mathbf{y}_1/2 \\ \mathbf{y}_2 \end{bmatrix} \quad (2.7)$$

$$\mathbf{x}_N = \frac{(\mathbf{x}_1 - \mathbf{x}_2)}{2} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \quad (2.8)$$

Thus the vector \mathbf{x}_A can be uniquely identified, and contains all the information we can know from (2.1) and the path measurements. The intuition of our scheme is illustrated through *virtual links* in [13].

Because \mathbf{x}_A lies in the k -dimensional space $\mathcal{R}(A^T)$, only k independent equations of the m equations in (2.1) are needed to uniquely identify \mathbf{x}_A . We measure these k paths to compute \mathbf{x}_A . Since $\mathbf{y} = A\mathbf{x} = A\mathbf{x}_A + A\mathbf{x}_N = A\mathbf{x}_A$, we can compute all elements of \mathbf{y} from \mathbf{x}_A , and thus obtain the loss rate of all other paths. Next, we present more detailed algorithms.

2.2.2 Basic Static Algorithms

The basic algorithms involve two steps. First, we select a basis set of k paths to monitor. Such selection only needs to be done once at setup. Then, based on continuous monitoring of the selected paths, we calculate and update the loss rates of all other paths.

2.2.2.1 Measurement Paths Selection

To select k linearly independent paths from A , we use standard rank-revealing decomposition techniques [32], and obtain the reduced system (Eq. 2.5) discussed in Section 2.1. where $A_S \in \mathbb{R}^{k \times n}$ and $\mathbf{y}_S \in \mathbb{R}^k$ consist of k rows of A and \mathbf{y} , respectively. The equation is illustrated in Fig. 2.1(b) (compared with $A\mathbf{x} = \mathbf{y}$).

As shown below, our algorithm is a variant of the QR decomposition with column pivoting [32, p.223]. It incrementally builds a decomposition $A_S^T = QR$, where $Q \in \mathbb{R}^{n \times k}$ is a matrix with orthonormal columns and $R \in \mathbb{R}^{k \times k}$ is upper triangular.

```

procedure SelectPath(A)
1 for every row(path)  $v$  in  $A$  do
2    $\hat{R}_{12} = R^{-T} A_S v^T = Q^T v^T$ 
3    $\hat{R}_{22} = \|v\|^2 - \|\hat{R}_{12}\|^2$ 
4   if  $\hat{R}_{22} \neq 0$  then
5     Select  $v$  as a measurement path
6     Update  $R = \begin{bmatrix} R & \hat{R}_{12} \\ 0 & \hat{R}_{22} \end{bmatrix}$  and  $A_S = \begin{bmatrix} A_S \\ v \end{bmatrix}$ 
   end if
end for

```

Figure 2.3: Path (row) selection algorithm

In general, the A matrix is very sparse; that is, there are only a few nonzeros per row. We leverage this property for speedup. We further use optimized routines from the LAPACK library [6] to implement row selection algorithm so that it inspects several rows at a time. The complexity of row selection algorithm is $O(mk^2)$, and the constant in the bound is modest. The memory cost is roughly $k^2/2$ single-precision floating point numbers

for storing the R factor. Notice that the path selection only needs to be executed once for initial setup.

2.2.2.2 Path Loss Rate Calculations

To compute the path loss rates, we must find a solution to the underdetermined linear system $A_S \mathbf{x}_A = \mathbf{y}_S$. The vector \mathbf{y}_S comes from measurements of the paths. Zhang *et al.* report that path loss rates remain operationally stable in the time scale of an hour [71], so these measurements need not be taken simultaneously.

Given measured values for \mathbf{y}_S , we compute a solution \mathbf{x}_A using the QR decomposition we constructed during measurement path selection [32, 22]. We choose the unique solution \mathbf{x}_A with minimum possible norm by imposing the constraint $\mathbf{x}_A = A_S^T \mathbf{z}$ where $\mathbf{z} = R^{-1} R^{-T} \mathbf{y}_S$. Once we have \mathbf{x}_A , we can compute $\mathbf{y} = A \mathbf{x}_A$, and from there infer the loss rates of the unmeasured paths. The complexity for this step is only $O(k^2)$. Thus we can update loss rate estimates online.

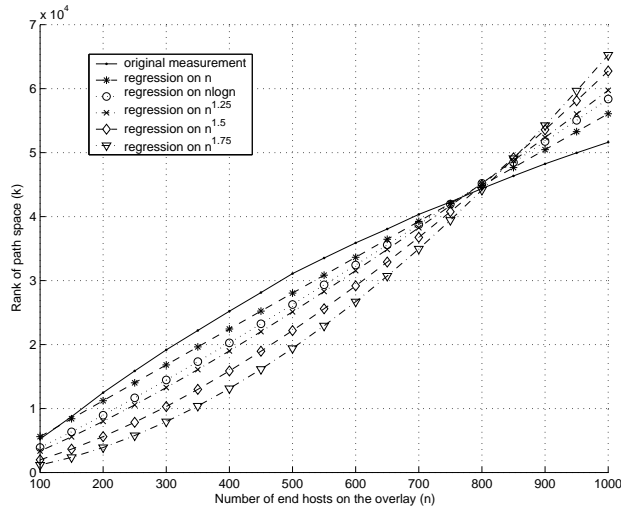


Figure 2.4: Regression of k in various functions of n under different router-level topologies.

2.2.3 Scalability Analysis

An overlay monitoring system is scalable only when the size of the basis set, k , grows relatively slowly as a function of n . Given that the Internet has moderate hierarchical structure [66, 29], we proved that the number of end hosts is no less than half of the total number of nodes in the Internet. Furthermore, we proved that when all the end hosts are on the overlay network, $k = O(N)$ [13].

But what about if only a small fraction of the end hosts are on the overlay? Because A is an m by n matrix, k is bounded by the number of links n . If the Internet topology is a strict hierarchy like a tree, $n = O(N)$, thus $k = O(N)$. But if there is no hierarchy at all (*e.g.* a clique), $k = O(N^2)$ because all the $O(N^2)$ paths are linearly independent. Tangmunarunkit *et al.* found that the power-law degree Internet topology has moderate hierarchy [66]. It is our conjecture that $k = O(N \log n)$.

In this section, we will show through linear regression on real topologies that k is indeed bounded by $O(N \log N)$ for reasonably large N (*e.g.*, 100). We explain it based on the power-law degree distribution of the Internet topology and the AS (Autonomous System) hierarchy.

We randomly select end hosts which have the least degree (*i.e.*, leaf nodes) to form an overlay network. We test by linear regression of k on $O(N)$, $O(N \log N)$, $O(N^{1.25})$, $O(N^{1.5})$, and $O(N^{1.75})$. As shown in Fig. 2.4, results are averaged over three runs with different random sets of end hosts. We find that $O(N)$ regression has the least residual errors - actually k even grows slower than $O(N)$. Considering the simulated topological models not included in this dissertation, we can conservatively say $k = O(N \log N)$.

Note that such trend still holds when the end hosts are sparsely distributed in the Internet, *e.g.*, when each end host is in a different access network. One extreme case is the “star” topology - each end host is connected to the same center router via its own access network. In such a topology, there are only N links. Thus $k = O(N)$. Only topologies with very dense connectivity, like a full clique, have $k = O(N^2)$. Those topologies have little link sharing among the end-to-end paths.

The key observation is that when N is sufficiently large, such dense connectivity is very unlikely to exist in the Internet because of the power-law degree distribution. Tangmunarunkit *et al.* found that link usage, as measured by the set of node pairs (source-destination pairs) whose traffic traverses the link, also follows a power-law distribution, *i.e.*, there is a very small number of links that are on the shortest paths of the majority of node pairs. So there is significant amount of link sharing among the paths, especially for backbone links, customer links, and peering links.

Such link sharing can easily lead to rank deficiency of the path matrix for overlay networks. As an example, consider an overlay within a single AS. The AS with the largest number of links (exclusive of customer and peering links) in [63] has 5,300 links. Even considering the coverage factor (55.6% as in Table 2 of [63]), there are at most 9,600 links. Since there are $N(N - 1)$ paths among n nodes, link sharing must occur before $N = 100$; in fact, substantial link sharing is likely to occur for even smaller N .

Now consider an overlay network that spans two ASes connected by c customer/peering links, with $N/2$ nodes in one AS and $N/2$ nodes in the other. The $N^2/2$ cross-AS paths can be modelled as linear combination of $2c \times N + 2c$ virtual links - bi-directional links from each end host to its c peering link routers, and c bi-directional peering links. Thus given c is normally much less than N and can be viewed as a constant, only $O(N)$ paths need to be measured for the $O(N^2)$ cross-AS paths.

Now consider an overlay on multiple ASes. According to [64], there are only 20 ASes (*tier-1* providers) which form the dense core of the Internet. These ASes are connected almost as a clique, while the rest of the ASes have far less dense peering connectivity. So when the size of an overlay is reasonably big (*e.g.*, $N > 100$), the number of customer and peering links that cross-AS paths traverse tends to grow much slower than $O(N^2)$. For example, a joining end host may only add one customer link to the overlay topology, and share the peering links that have been used by other end hosts. Meanwhile, only a few nodes are needed in a single AS before link sharing occurs in paths within an AS.

We believe this heavy sharing accounts for our empirical observation that $k = O(N)$ in a real router-level topology, and k grows at worst like $O(N \log N)$ in several generated topologies. Note that the real 284,805-router topology represents a fairly complete transit portion of the Internet [35]. In our analysis, we conservatively assume that there is only one end host connecting to each edge router to reduce the possible path sharing, but we still find $k = O(N)$ when $N > 100$.

2.3 Approximate Reconstruction of Network Path Properties

In this section, we illustrate a network measurement framework named NetQuest which approximately reconstructs path properties. In the subsequent subsections, we describe design of experiments, inference algorithms, and evaluation of NetQuest.

2.3.1 Design of Experiments

Network measurement, especially when conducted at large scale, requires carefully designed measurement experiments. The design involves specifying all aspects of an experiment and choosing the values of variables that can be controlled before the experiment starts. Making the design decisions is challenging given the complexity involved in experiments. Manually making all the design decisions is both time consuming and error prone. It is therefore highly desirable to automate the design process and do so in a mathematically sound manner. Not all aspects of experimental design are amenable to formal mathematical treatment. Choosing the values for the control variables however can be expressed in a coherent mathematical framework through the use of *Bayesian experimental design*, which has gained considerable popularity in the past three decades. Below we first give a brief overview of Bayesian experimental design (see [12, 17] for a detailed review). We then put it into the context of large-scale network measurement and demonstrate how the general framework can be applied to meet common design requirements.

2.3.1.1 Bayesian Experimental Design

The basic idea in experimental design is that one can improve the statistical inference about the quantities of interest by properly choosing the values of the control variables. This can be formally described in a Bayesian decision theoretic framework as proposed by Lindley in 1972 [45, page 19 and 20]. Below we first set up the framework using decision theoretic terminologies, and then put it into the context of network performance monitoring.

As summarized in [12], Lindley’s framework is the following. Suppose one wants to conduct an experiment on a system with unknown parameters \mathbf{x} drawn from parameter space \mathcal{X} . Before the experiment, a design η must be chosen from some set \mathcal{H} . Through the experiment, data \mathbf{y} from a sample space \mathcal{Y} will be observed. Based on \mathbf{y} a terminal decision d will be chosen from some set \mathcal{D} . In the context of network performance monitoring, the terminal decision d is an estimate of our quantity of interest $f(\mathbf{x})$, where \mathbf{x} is the vector of unknown link performance. A design η refers to a set of paths, S , which we choose to probe. Through the experiment, we observe end-to-end performance on the set of paths in S : \mathbf{y}_S , which satisfies $\mathbf{y}_S = A_S \mathbf{x}$. Here A_S is the routing matrix formed by the set of

rows corresponding to paths in S . So the goal is to estimate $f(\mathbf{x})$ based on the observed end-to-end performance on the set of paths in S (i.e., \mathbf{y}_S). So the whole decision process consists of two parts: first the selection of η (i.e., S , in our context), and then the choice of a terminal decision d (i.e., an estimate of $f(\mathbf{x})$, in our context). A general utility function of the form $U(d, \mathbf{x}, \eta, \mathbf{y})$ is used to reflect the purpose of the experiment. For example, it may reflect the expected accuracy of an estimator for $f(\mathbf{x})$ in the context of network performance inference.

Bayesian experimental design suggests that a good solution to the experimental design problem is the design that maximizes the expected utility of the best terminal decision. More formally, for a given design η , the expected utility of the best terminal decision is

$$U(\eta) = \int_{\mathbf{y}} \max_{d \in D} \int_{\mathbf{x}} U(d, \mathbf{x}, \eta, \mathbf{y}) p(\mathbf{x}|\mathbf{y}, \eta) p(\mathbf{y}|\eta) d\mathbf{x}d\mathbf{y}, \quad (2.9)$$

where $p(\cdot)$ denotes a probability density function with respect to an appropriate measure. Bayesian experimental design would then choose the design η^* that maximizes the above $U(\eta)$:

$$U(\eta^*) = \max_{\eta \in \mathcal{H}} \int_{\mathbf{y}} \max_{d \in D} \int_{\mathbf{x}} U(d, \mathbf{x}, \eta, \mathbf{y}) p(\mathbf{x}|\mathbf{y}, \eta) p(\mathbf{y}|\eta) d\mathbf{x}d\mathbf{y}. \quad (2.10)$$

1) Bayesian Designs for Path Selection

We now apply Bayesian experimental design to solve the path selection problem. Different Bayesian designs can be obtained by choosing different utility functions to assess the quality of individual designs. Below we introduce two such designs: *Bayesian A-optimal design* and *Bayesian D-optimal design*.

Bayesian A-optimal design: For estimating $f(\mathbf{x}) = F\mathbf{x}$, we can use the squared error $\|F\mathbf{x} - F\mathbf{x}_e\|_2^2 = (F\mathbf{x} - F\mathbf{x}_e)^T (F\mathbf{x} - F\mathbf{x}_e)$ to assess the inaccuracy of an estimator $F\mathbf{x}_e$. So a design η can be chosen to maximize the following expected utility

$$U_A(\eta) = - \int (F\mathbf{x} - F\hat{\mathbf{x}})^T (F\mathbf{x} - F\hat{\mathbf{x}}) p(\mathbf{y}, \mathbf{x}|\eta) d\mathbf{x}d\mathbf{y}, \quad (2.11)$$

where $\hat{\mathbf{x}}$ is the estimated \mathbf{x} under the best decision rule d .

To derive an easy-to-use design criterion, we will assume a normal linear system. Specifically, we assume that $\mathbf{y}_S|\mathbf{x}, \sigma^2 \sim A_S\mathbf{x} + N(0, \sigma^2 I)$, where σ^2 is the known variance for the zero mean Gaussian measurement noise, and I is the identity matrix. Suppose the prior information is that $\mathbf{x}|\sigma^2$ is randomly drawn from a multivariate normal distribution

with mean vector μ and covariance matrix $\Sigma = \sigma^2 R^{-1}$, where μ and matrix R are known *a priori*.

Let $D(\eta) = (A_S^T A_S + R)^{-1}$. The Bayesian procedure yields

$$U_A(\eta) = -\sigma^2 \text{tr}\{FD(\eta)F^T\}, \quad (2.12)$$

where $\text{tr}\{M\}$ (the trace of a matrix M) is defined as the sum of all the diagonal elements of M .

Maximizing $U_A(\eta)$ thus reduces to minimizing the function

$$\phi_A(\eta) = \text{tr}\{FD(\eta)F^T\}, \quad (2.13)$$

which is commonly referred to as the *Bayesian A-optimality*.

Bayesian D-optimal design: It is also common to replace the quadratic error function in Bayesian *A-optimality* with an information-theoretic metric. Specifically, one can choose a design that maximizes the expected gain in Shannon information or, equivalently, maximizes the expected Kullback-Leibler distance between the posterior and the prior distributions:

$$\int \log \frac{p(F\mathbf{x}|\mathbf{y}, \eta)}{p(F\mathbf{x})} p(\mathbf{y}, \mathbf{x}|\eta) d\mathbf{x}d\mathbf{y}. \quad (2.14)$$

Since the prior distribution of $F\mathbf{x}$ does not depend on the design η , maximizing (2.14) is equivalent to maximizing

$$U_D(\eta) = \int \log\{p(F\mathbf{x}|\mathbf{y}, \eta)\} p(\mathbf{y}, \mathbf{x}|\eta) d\mathbf{x}d\mathbf{y}. \quad (2.15)$$

Under a normal linear model, the Bayesian procedure yields

$$U_D(\eta) = -\frac{n}{2} \log(2\pi) - \frac{n}{2} - \frac{1}{2} \log \det\{FD(\eta)F^T\}, \quad (2.16)$$

where $\det\{M\}$ denotes the determinant of a matrix M .

Maximizing $U_D(\eta)$ thus reduces to minimizing the function

$$\phi_D(\eta) = \det\{FD(\eta)F^T\}, \quad (2.17)$$

which we define as the *Bayesian D-optimality*.

One problem with (2.17) is that when $\text{rank}(F) < r$ (r is the number of rows in F), $\det\{FD(\eta)F^T\}$ is always 0 and thus cannot distinguish different designs. Our solution is to redefine

$$\phi_D(\eta) = \Pi_{\text{rank}(F)}\{FD(\eta)F^T\}, \quad (2.18)$$

where $\Pi_k\{M\}$ is the product of the k largest eigenvalues of M .

It is often reasonable to require that a design η remains good under a small perturbation to the function of interest: $f(\mathbf{x}) = F\mathbf{x}$. As a result, if F is not full-rank, we can perturb F slightly to make it full-rank. Therefore, we only need to consider the case when $\text{rank}(F) = \min(r, n)$. In this case, we can simplify (2.18) into

$$\phi_D(\eta) = \begin{cases} \det\{F^T F\} \det\{D(\eta)\} & \text{if } r \geq n, \\ \det\{FD(\eta)F^T\} & \text{otherwise.} \end{cases} \quad (2.19)$$

Note that when $r \geq n$, minimizing $\phi_D(\eta)$ reduces to minimizing $\det\{D(\eta)\}$ or, equivalently, maximizing $\det\{A_S^T A_S + R\}$.

2) Search Algorithm

Once we have chosen a design criterion $\phi(\eta)$, the next step is to find the optimal design $\eta^* = \arg \min_{\eta} \phi(\eta)$. However, the problem of finding s rows of A to minimize a given design criterion $\phi(\eta)$ is known to be NP -complete and it is computationally infeasible to compute the optimal design exactly. To address the issue, we search for a good design using a simple *sequential search algorithm*. The algorithm starts with an empty initial design and then sequentially adds rows to the design. During each iteration, it greedily selects the row that results in the largest reduction in $\phi(\eta)$. The basic algorithm is illustrated in Figure 2.5.

It is possible to further improve the design obtained by the sequential search algorithm using an *exchange algorithm* (e.g., [30, 51, 49, 55]), which tries to reduce $\phi(\eta)$ by iteratively exchanging paths. We have implemented Fedorov's exchange algorithm, which has been shown to yield the best performance among many existing algorithms [55]. However, our experience suggests that the additional path exchanges do not yield noticeable performance improvement in the context of network performance inference. *So we disable the exchange algorithm in the interest of efficiency.*

3) Incremental Update of Design Criterion

For large-scale network measurement, it is essential for the search algorithm to be highly efficient, because the design space is often very large due to the quadratic growth in the number of network paths with respect to the number of network nodes. The major bottleneck

```

1   $S = \emptyset$  // initialize the path set to be empty
2  for  $iter = 1$  to  $s$  //  $s$  is the desired number of paths
3    for each path  $i \in \{1, 2, \dots, m\} - S$ 
4      // compute the new design criterion after adding path  $i$ 
5       $criteria[i] = \phi(S \cup \{i\})$ 
6    end for
7    // select the path that minimizes the new design criterion
8     $S = S \cup \{\arg \min_i criteria[i]\}$ 
9  end for
10 return  $S$ 

```

Figure 2.5: Sequential path selection for Bayesian designs.

of the above search algorithm is computing the new design criterion $\phi(S \cup \{i\})$ after adding a path i (line 5 in Figure 2.5). Recall that both $\phi_A(\eta)$ and $\phi_D(\eta)$ are functions of $FD(\eta)F^T = F(A_S^T A_S + R)^{-1}F^T$. Given the size of $(A_S^T A_S + R)$ and F , it is inefficient to compute $\phi(S \cup \{i\})$ from scratch due to the expensive matrix inversion and matrix multiplication operations involved.

In NetQuest, we significantly improve the efficiency of the search algorithm by applying incremental methods to update the design criterion. With such optimizations, NetQuest has successfully handled routing matrices A with 1,000,000 rows, 50,000 columns, and over 5,000,000 non-zero entries (corresponding to paths among 1,000 nodes). Below we present these incremental update methods in detail.

Notations: Let $S' = S \cup \{i\}$, $M = A_S^T A_S + R$, $M' = A_{S'}^T A_{S'} + R$, $N = FM^{-1}F^T$, $N' = F(M')^{-1}F^T$. With these notations, we have $\phi_A(S) = \text{tr}\{N\}$, $\phi_A(S') = \text{tr}\{N'\}$, $\phi_D(S) = \det\{N\}$, $\phi_D(S') = \det\{N'\}$.

Incremental computation of $\phi_A(S \cup \{i\})$: Let \mathbf{a}_i^T denote the i -th row vector of A . It is easy to verify that $M' = M + \mathbf{a}_i \mathbf{a}_i^T$. That is, M' can be derived from M using a rank-1 matrix update. We have the following result from matrix algebra

$$(M')^{-1} = (M + \mathbf{a}_i \mathbf{a}_i^T)^{-1} = M^{-1} - \alpha \mathbf{u} \mathbf{u}^T, \quad (2.20)$$

where $\mathbf{u} = M^{-1} \mathbf{a}_i$, $\alpha = 1/(1 + \mathbf{a}_i^T \mathbf{u})$.

Combine (2.20) with $N = FM^{-1}F^T$ and $N' = F(M')^{-1}F^T$, we obtain $N' = N - \alpha(F\mathbf{u})(F\mathbf{u})^T$. As a result, we have

$$\text{tr}\{N'\} = \text{tr}\{N\} - \text{tr}\{\alpha(F\mathbf{u})(F\mathbf{u})^T\} = \text{tr}\{N\} - \alpha \|F\mathbf{u}\|_2^2. \quad (2.21)$$

Therefore, we can compute $\phi_A(S \cup \{i\}) = \text{tr}\{N'\}$ incrementally by computing $\mathbf{u} = M^{-1}\mathbf{a}_i$, $\alpha = 1/(1 + \mathbf{a}_i^T \mathbf{u})$, and $\|F\mathbf{u}\|_2^2$ (note that M^{-1} remains fixed for different i). These operations are much more efficient than matrix inversion and matrix multiplication, which are required to compute $\phi_A(S \cup \{i\})$ from scratch.

Incremental computation of $\phi_D(S \cup \{i\})$: Let $\mathbf{b} = \sqrt{\alpha} \cdot F\mathbf{u}$. We have $N' = N - \mathbf{b}\mathbf{b}^T$. Using results in matrix algebra, we have

$$(N')^{-1} = (N - \mathbf{b}\mathbf{b}^T)^{-1} = N^{-1} + \beta\mathbf{v}\mathbf{v}^T, \quad (2.22)$$

$$\det\{N'\} = \det\{N - \mathbf{b}\mathbf{b}^T\} = \frac{1}{\beta} \det\{N\}, \quad (2.23)$$

where $\mathbf{v} = N^{-1}\mathbf{b}$, and $\beta = 1/(1 - \mathbf{b}^T \mathbf{v})$.

Therefore, we can compute $\phi_D(S \cup \{i\}) = \det\{N'\}$ incrementally by computing $\mathbf{v} = N^{-1}\mathbf{b}$, and $\beta = 1/(1 - \mathbf{b}^T \mathbf{v})$ (note that N^{-1} remains fixed for different i). These operations are much more efficient than the matrix inversion and matrix multiplication operations required for computing $\phi_D(S \cup \{i\})$ from scratch.

Further optimization: With the above incremental update methods, we need to update M^{-1} and N^{-1} each time after a new path is added to S (line 8 in Figure 2.5). This takes $O(n^2)$ operations using (2.20) and (2.22). We can further improve efficiency by maintaining the Cholesky factorization of M and N (instead of M^{-1} and N^{-1}), which in general are much sparser and thus more efficient to update incrementally. Note that the only use of M^{-1} and N^{-1} is to compute quantities $\mathbf{u} = M^{-1}\mathbf{a}_i$ and $\mathbf{v} = N^{-1}\mathbf{b}$. We can compute the same quantities using the Cholesky factorization. For example, if we write $M = LL^T$, where L is the lower-triangular factorization of M , we have $\mathbf{u} = (L^T)^{-1}(L^{-1}\mathbf{a}_i)$, which can be computed efficiently using back-substitution without inverting L and L^T .

In NetQuest, we use LDL [21], a MATLAB package highly optimized for incremental Cholesky factorization of sparse matrices. Our experience suggests that the resulting algorithm achieves an order of magnitude speedup over directly updating M^{-1} and N^{-1} .

2.3.1.2 Flexibility

Our Bayesian experimental design framework is very flexible and can accommodate common requirements in large-scale network measurement. Below we cover three common scenarios.

Differentiated design: In large-scale network performance monitoring, different quantities of interest may not always have the same importance. For example, a subset of paths may belong to a major customer and it is important to monitor those paths more extensively than the other paths. We can accommodate such differentiated design requirement in Bayesian A -optimality by assigning higher weights to those important rows of matrix F in the objective function $f(\mathbf{x}) = F\mathbf{x}$.

Augmented design: Augmented design is useful in large-scale network measurement for several reasons. First, when some of the measurements in a previous design failed, we do not want to design a new experiment completely from scratch, but instead would like to leverage the data that we have already observed as much as possible. Second, augmented design can also be used to design active measurement experiments that complement well with the existing passive measurement (*i.e.*, the additional information gain is maximized). Our design framework can naturally support the augmentation of a previous design. Specifically, let S_0 be the set of rows we obtain in the previous design. We just need to redefine

$$D(\eta) = (A_{S \cup S_0}^T A_{S \cup S_0} + R)^{-1} = (A_S^T A_S + R + A_{S_0}^T A_{S_0})^{-1} \quad (2.24)$$

where $S \cap S_0 = \emptyset$. We can then apply the Bayesian A -optimal and D -optimal design criteria to find S , the set of additional paths to probe. In addition, we also extend QR and SVD, which will be described in Section 2.3.1.3, to support augmented design by excluding the paths with successful measurements and applying SVD or QR to select a set of paths from the remaining rows.

Multi-user design: In large-scale network performance monitoring, there may be multiple users who are interested in different parts of the network and may have different functions of interest. We can support such scenarios by using a linear combination of individual users' design criteria as the overall design criterion. Formally, given a set of users $1, 2, \dots, u$, let $\phi_i(\eta)$ be the preferred design criterion for each user i (which may depend on his/her interest: $f_i(\mathbf{x}) = F_i\mathbf{x}$). We can then use $\phi(\eta) = \sum_i w_i \phi_i(\eta)$ as the combined design criterion, where w_i is the weight associated with user i . As a concrete example, consider the case where Bayesian A -optimality is used by all the users. In this case, we have $\phi_i(\eta) = \text{tr}\{F_i D(\eta) F_i^T\}$. We can show

$$\phi(\eta) = \sum_i w_i \text{tr}\{F_i D(\eta) F_i^T\} = \text{tr}\{F D(\eta) F^T\}, \quad (2.25)$$

where $F = \text{vertcat}\{w_i^{1/2} F_i\}$ is the vertical concatenation of matrices $w_i^{1/2} F_i$. Therefore, the optimal design using the combined design criterion is equivalent to the Bayesian A -optimal design for the combined function: $f(\mathbf{x}) = F\mathbf{x}$. Note that if a subset of users (U) share a common row in their F_i , this row will appear as multiple rows in F . These rows can be

merged into a single row with a combined weight of $(\sum_{i \in U} w_i)^{1/2}$. In the special case when $w_i = 1$, the combined weight is simply $|U|^{1/2}$, *i.e.*, the square root of the number of users interested in the row.

Besides the above three common scenarios, our design framework can easily incorporate other constraints in the design space (*e.g.*, the maximum amount of paths that one can probe at each monitoring site, and the number of monitoring sites available). As part of our future work, we plan to further investigate them in the context of specific network monitoring applications.

2.3.1.3 Non-Bayesian Designs

For performance comparison, we will also examine the following non-Bayesian solutions to the path selection problem.

Rank based solution: In Section 2.2, we presented a linear algebraic approach to efficient monitoring of overlay paths. In the context, given N overlay nodes, there are $N(N - 1)$ different paths. So A has $N(N - 1)$ rows. The quantity of interest is $f(\mathbf{x}) = A\mathbf{x}$ (*i.e.*, the performance on all overlay paths). Given the rank of matrix A , k , the solution is based on the observation that any subset of k independent rows of A , denoted as A_S , are sufficient to span the space of A : $\{\mathbf{y} \in \mathbb{R}^m \mid \exists \mathbf{x} \in \mathbb{R}^n \text{ s.t. } \mathbf{y} = A\mathbf{x}\}$. As a result, given the measurements for paths corresponding to the rows in A_S , one can reconstruct the end-to-end performance on all paths *exactly*. As we have seen from Section 2.2.3, k gives an upper bound on the number of paths that one needs to probe.

SVD based solution: Chua *et al.* [15, 16] presented a statistical framework for monitoring a linear summary of the end-to-end path performance. Their quantity of interest is $f(\mathbf{x}) = \ell^T A\mathbf{x}$, where ℓ is a weight vector. This is a network-wide metric, *e.g.*, average delay of all paths in a network. It is a special case of the inference problem we study in this work.

Chua *et al.* observed that routing matrices encountered in practice generally show significant sharing of links between paths. As a result, A tends to have small *effective* rank compared to their actual matrix rank. That is, a small subset of eigenvalues of $A^T A$ tend to be much larger than the rest. Based on the observation, Chua *et al.* proposed to select a subset of s paths such that the corresponding rows span as much of $\mathcal{R}(A)$ as possible, where $\mathcal{R}(A)$ is the subspace formed by all possible linear combinations of the rows in A . Algorithmically, this problem is equivalent to the subset selection problem in the field of computational linear algebra (see [34, Ch 12]). So [15] adapted the method described in Algorithm 12.2.1 of [34, p. 574], which is based on the singular value decomposition (SVD). Subsequently in [16], Chua *et al.* extends their algorithm to incorporate Σ , the covariance

- 1 compute C such that $\Sigma = CC^T$
- 2 compute SVD of AC : $U\nabla V^T = AC$
- 3 extract the first s column vectors of U : $U_s = [\mathbf{u}_1 \mathbf{u}_2 \cdots \mathbf{u}_s]$
- 4 compute the QR factorization with column pivoting on U_s^T :
 $QR = (U_s[E, \cdot])^T$ (E is a permutation vector for rows of U_s)
- 5 return the first s elements of E : $S = \{e_1, e_2, \cdots, e_s\}$

Figure 2.6: SVD based path selection algorithm

- 1 compute C such that $\Sigma = CC^T$
- 2 compute $G = (AC)^T$
- 3 compute the QR factorization with column pivoting on G :
 $QR = G[:, E]$ (E is a permutation vector for columns of G)
- 4 return the first s elements of E : $S = \{e_1, e_2, \cdots, e_s\}$

Figure 2.7: QR based path selection algorithm

matrix of \mathbf{x} . It assumes that Σ is a diagonal matrix (but allows diagonal elements to be different). The resulting algorithm is summarized in Figure 2.6.

Path selection under general link covariance Σ (*e.g.*, when link performance has spatial dependency) is left as an *open* problem in [16]. Our Bayesian experimental design framework works for any link covariance matrix, and therefore solves this problem.

QR based solution: The third alternative solution is directly based on QR factorization with column pivoting. It is one of the two algorithms proposed by Golub *et al.* [33] for selecting *numerically* independent rows/columns (the other algorithm is the SVD based solution described above). As noted in [33], the QR based solution is generally more efficient than the SVD based solution and often achieves comparable performance. We extend the algorithm in [33] to incorporate the link covariance matrix Σ when Σ is a diagonal matrix. The resulting algorithm is illustrated in Figure 2.7.

Summary: Rank based solution requires monitoring $\text{rank}(A)$ number of paths, which can be expensive in large networks. SVD and QR based solutions can monitor fewer paths at the cost of higher error. We further enhance the flexibility of SVD and QR by extending them to support augmented design. Nevertheless, as we will show, Bayesian experimental design can out-perform the alternatives in the following regards: (i) it achieves higher accuracy when monitoring the same number of paths as SVD and QR, (ii) it is scalable and can be applied to networks of thousands of nodes, and (iii) it is flexible to support diverse design requirements.

Notation	Inference algorithm	Section
MinL2	L_2 norm minimization	§2.3.2.1
MinL1	L_1 norm minimization	§2.3.2.2
Entropy	Maximum Entropy Estimation	§2.3.2.3

Table 2.3: Inference algorithms. **MinL2** and **MinL1** can optionally incorporate the nonnegativity constraints: $\mathbf{x} \geq 0$, resulting in **MinL2_nonNeg** and **MinL1_nonNeg**, respectively.

2.3.2 Inference Algorithms

The other major component to the NetQuest framework is network inference, which reconstructs the quantities of interest \mathbf{x} based on partial, indirect observations \mathbf{y} by solving (2.1). Since NetQuest selects only a small number of measurement experiments to conduct, the number of observables can be much smaller than the number of unknowns. Therefore, the linear inverse problem in (2.1) is often *under-determined*. A lot of solutions have been developed for under-determined linear inverse problems. As we noted in [74], many such proposals solve the regularized least-squares problem

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 J(\mathbf{x}), \quad (2.26)$$

where $\|\cdot\|_2$ denotes the L_2 norm, λ is a regularization parameter, and $J(\mathbf{x})$ is a penalization functional. Proposals of this kind have been used in a wide range of fields, with considerable practical and theoretical success when the data match the assumptions of the method, and the regularization functional matches the properties of the estimand. See [36] for a general description of regularization.

In this work, we compare the accuracy of several widely used inference algorithms (summarized in Table 2.3). The goal is to understand how combinations of different experimental design methods and inference algorithms affect the overall inference accuracy.

2.3.2.1 L_2 Norm Minimization

A common solution to (2.1) is *L_2 norm minimization*, which corresponds to (2.26) with $J(\mathbf{x}) = \|\mathbf{x}\|_2^2$.

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x}\|_2^2. \quad (2.27)$$

If we have prior information that \mathbf{x} is close to an initial solution μ , we can replace $\|\mathbf{x}\|_2$ with $\|\mathbf{x} - \mu\|_2$ in (2.27), resulting in

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x} - \mu\|_2^2. \quad (2.28)$$

(2.28) is also commonly referred to as the Tikhonov regularization [36]. It can be efficiently solved using standard solvers for linear least-squares problems. If desired, one can incorporate the nonnegativity constraints $\mathbf{x} \geq 0$ into (2.28). The resulting nonnegative linear least-squares problem can be solved using PDCO [59], a MATLAB package highly optimized for problems with sparse matrices A .

2.3.2.2 L_1 Norm Minimization

Another common solution to (2.1) is L_1 norm minimization, which corresponds to (2.26) with $J(\mathbf{x}) = \|\mathbf{x}\|_1$ (i.e., the L_1 norm of \mathbf{x}).

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x}\|_1. \quad (2.29)$$

L_1 norm minimization is often used in situations where \mathbf{x} is *sparse*, i.e., \mathbf{x} has only very few large elements and the other elements are all close to 0. This can happen, for instance, in loss inference, where most links have close to 0 loss rate (and thus $\log\{1 - \text{loss rate}\}$ is close to 0). In such scenarios, ideally one would like to find the sparsest solution to $\mathbf{y} = A\mathbf{x}$ by minimizing the L_0 norm $\|\mathbf{x}\|_0$ (i.e., the number of nonzeros in \mathbf{x}). But since the L_0 norm is not convex and is notoriously difficult to minimize, one often approximates L_0 norm with an L_1 norm. As shown in [25], the minimal L_1 norm solution often coincides with the sparsest solution for under-determined linear systems. We have successfully applied L_1 norm minimization to network anomaly inference [72].

As in [72], we solve the following variant of (2.29)

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_1 + \lambda \|\mathbf{x}\|_1. \quad (2.30)$$

An advantage of (2.30) is that it can be cast into an equivalent Linear Programming (LP) problem, for which solutions are available even with large-scale A , owing to modern interior-point LP methods. The LP formulation also allows one to incorporate additional linear constraints, such as the nonnegativity constraints $\mathbf{x} \geq 0$. Finally, if we have prior information that \mathbf{x} is close to an initial solution μ , we can replace $\|\mathbf{x}\|_1$ with $\|\mathbf{x} - \mu\|_1$ in (2.30), yielding

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_1 + \lambda \|\mathbf{x} - \mu\|_1. \quad (2.31)$$

2.3.2.3 Maximum Entropy Estimation

For inference under the nonnegativity constraints $\mathbf{x} \geq 0$, another commonly used solution is *maximum entropy estimation*, which uses the negative entropy function as $J(\mathbf{x})$

in (2.26).

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \sum_i x_i \log x_i, \quad \mathbf{x} \geq 0. \quad (2.32)$$

If we know \mathbf{x} is close to an initial solution $\mu = [\mu_1 \mu_2 \cdots \mu_n]^T$, we can instead minimize the relative entropy [19], resulting in

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \sum_i x_i \log \frac{x_i}{\mu_i}, \quad \mathbf{x} \geq 0. \quad (2.33)$$

(2.33) can be efficiently solved by PDCO [59], which has been highly optimized for sparse matrices A . We have successfully applied (2.33) in the context of traffic matrix estimation [74].

2.3.3 Evaluation

2.3.3.1 Evaluation Methodology

Accuracy Metric

We quantify the inference accuracy using normalized mean absolute error (MAE), which is defined as

$$\text{normalized } MAE = \frac{\sum_i |inferred_i - actual_i|}{\sum_i actual_i}, \quad (2.34)$$

where $inferred_i$ and $actual_i$ are the inferred and actual end-to-end performance for path i . A lower value of normalized MAE indicates better accuracy.

PlanetLab Dataset Description

We evaluate our approach using real traces collected from PlanetLab Internet measurement testbed. Note that the real traces use round-trip performance measurements. As noted in Section 2.1, the problem formulation $\mathbf{y} = A\mathbf{x}$ works for both one-way and round-trip measurements.

Table 2.4 summarizes the RTT and loss traces, where *nodes* include both end hosts and intermediate routers on the end-to-end paths, and *overlay nodes* only include end hosts among which the end-to-end performance needs to be monitored or estimated.

We construct routing matrix, A , using traceroute measurements. We derive the actual RTT based on the mean over 60 probes in every 1-minute interval, and derive the actual loss rates based on the mean over 300 probes in every 90-second interval. We use

	# nodes	# overlay nodes	# paths	# links	rank(A)
PlanetLab-RTT	2514	61	3657	5467	769
PlanetLab-loss	1795	60	3270	4628	690

Table 2.4: Summary of PlanetLab traces used for evaluation.

the following approach to derive the inferred RTT and loss rates: for the paths that are monitored we assume we know the true RTT and loss rates; for the un-monitored paths, we use the inference algorithms described in Section 2.3.2 to infer based on the observed performance of monitored paths. For each interval, we use normalized *MAE* to quantify the inference error.

2.3.3.2 Experimental Parameters

There are several parameters in Bayesian experimental design and network inference. Below we present the values that we use for these parameters in our evaluation.

Prior information for Bayesian experimental design (R): Recall that the design criteria for both Bayesian A - and D -optimality are functions of $D(\eta) = (A_S A_S^T + R)^{-1}$. To estimate R , one needs to estimate both the variance of the measurement noise (σ^2) and the covariance matrix of the link performance ($\Sigma = \sigma^2 R^{-1}$) through network measurement. However, the estimation of such second-order statistics in large-scale network measurement can be both expensive and inaccurate (due to measurement artifacts and non-stationarities in Internet path properties).

In NetQuest, we avoid such difficulties by setting $R = \epsilon I$, where ϵ is a small constant and I is the identity matrix. Our results suggest that this simple choice of R yields designs that consistently out-perform the alternative designs we considered (see Section 2.3.3.3). Moreover, the resulting design is highly insensitive to the choice of ϵ . In our evaluation, we set $\epsilon = 0.001$, which yields good results. Finally, note that a similar approach has been taken in the literature of Bayesian supersaturated design [39].

Prior information for network inference (μ): In this work, unless noted otherwise, we assume no prior information about \mathbf{x} . That is, we set $\mu = \mathbb{0}$ (an all-0 vector) for L_2 and L_1 norm minimization (both with and without nonnegativity constraints), and $\mu = \mathbb{1}$ (an all-1 vector) for maximum entropy estimation. Despite not using any prior information, our results show that NetQuest can achieve high accuracy by probing only a small fraction of paths.

Regularization parameter (λ): Our experience [74, 72] suggests that the inference algorithms are not sensitive to the choice of λ . In our evaluation, we use $\lambda = 0.01$, which gives

satisfactory results.

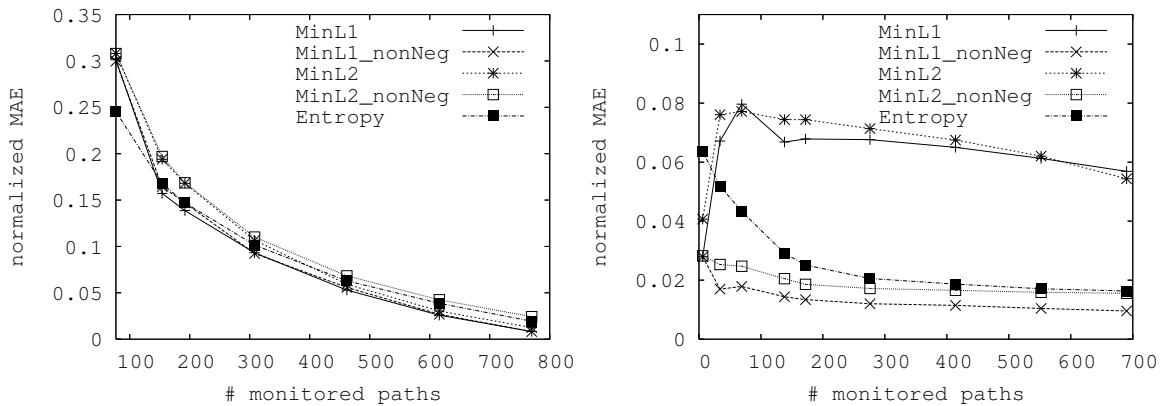
2.3.3.3 Basic Framework Evaluation Results

We first evaluate our basic measurement framework. Then we examine its capability of supporting flexible design requirements. Finally we study the effects of prior information.

In this section, we evaluate the two key components of our framework: (i) inference algorithms, and (ii) design of experiments.

Comparison of Inference Algorithms

First, we compare the accuracy of different inference algorithms. We use the A -optimal design criterion to determine which set of paths to monitor. Figure 2.8 (a) and (b) show the error of inferring end-to-end delay and loss rates as we vary the number of monitored paths. The x-value of the right most point on each curve corresponds to the rank of the routing matrix.



(a) Delay estimation in Planetlab-RTT.

(b) Loss estimation in Planetlab-loss.

Figure 2.8: Comparison of inference algorithms for delay and loss estimation using A -optimal design.

As shown in Figure 2.8 (a), different inference algorithms perform similarly for delay inference. Moreover, as expected, the error decreases with an increasing number of monitored paths.

As shown in Figure 2.8 (b), the performance difference between various inference algorithms is larger for loss rate inference. The inference algorithms that enforce nonnegativity constraints out-perform those that do not enforce such constraints. In addition, the inference error under those algorithms without nonnegativity constraints does not decrease with an

increasing number of monitored paths. Since loss rates take nonnegative values, intuitively enforcing nonnegativity constraints should give better inference. In comparison, the effect of nonnegativity constraints is much smaller for delay inference. This is because all paths have delay larger than 0, so even without enforcing nonnegativity constraints most links are assigned positive delay. Finally, MinL1 consistently out-performs the other inference schemes. As described in Section 2.3.2.2, MinL1 effectively maximizes the sparsity of link loss rate, which matches well with the fact that few links on the Internet are lossy.

From the above results, in the rest of the thesis, unless noted otherwise, we use MinL1_nonNeg for both delay and loss inference.

Comparison of Measurement Designs

Next we evaluate different algorithms for designing measurement experiments. A global view of the performance aggregated over an entire network is useful for a variety of reasons. It can be used for estimating a typical user experience (as in the Internet End-to-end Performance Monitoring Project (IEPM)), detecting anomalies, trouble-shooting, and optimizing performance. The pioneering work in this area is done by Chua et al. [15, 16], which is based on SVD.

We compare the Bayesian experimental designs with the other alternatives for inferring network wide average delay. Here the quantity of interest is $f(\mathbf{x}) = \frac{1}{m} \mathbb{1}A\mathbf{x}$, where $\mathbb{1}$ is an all-1 row vector of length m . We use the PlanetLab traces to evaluate the performance under a realistic scenario, and use simulated topologies to further test the scalability of our measurement design. Figure 2.9 (a) compares different experimental design schemes when MinL2 is used for inference, and Figure 2.9 (a) shows the results when MinL1_nonNeg is used. As noted in Section 2.3.1.1, A -optimal and D -optimal designs are identical for inferring network-wide average, so we only show the results of A -optimal.

As shown Figure 2.9 (a), with the A -optimal design, the inference error decays very fast – the error is within 15% by monitoring only within 2% paths (e.g., 77 out of 3657 paths in PlanetLab-RTT) In comparison, the inference error is 50% or higher (than A -optimal) when the same number of paths are monitored under the other schemes. In addition, we observe that to achieve within 10% inference error, the other schemes require monitoring 60% more paths than A -optimal. Finally, as the number of monitored paths increases, all the schemes converge to close to 0 inference error, since in this case there are sufficient information to reconstruct the global network view. Random selection converges slower because it does not ensure the selected paths are linearly independent. Similar results are observed in Figure 2.9 (b) when the inference algorithm changes to MinL1_nonNeg.

Summary

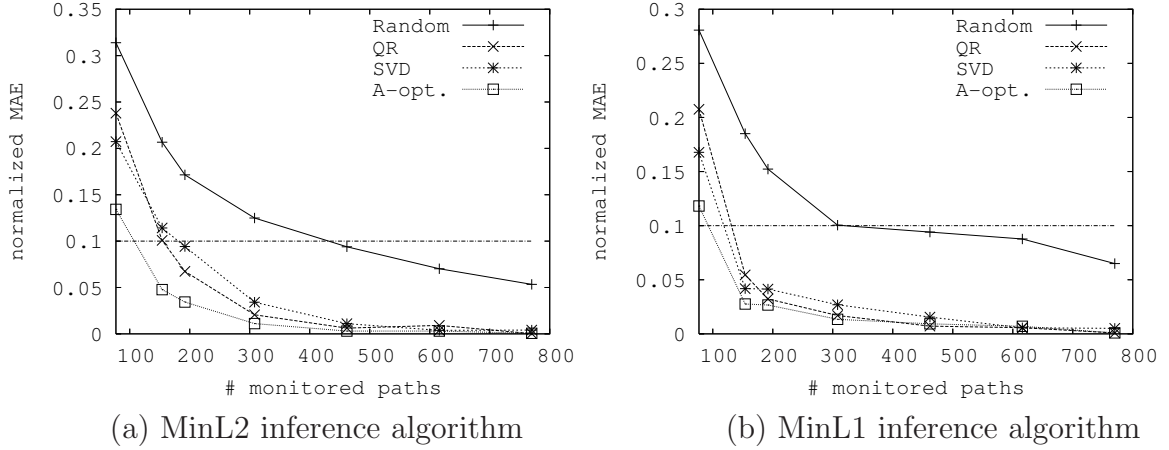


Figure 2.9: Comparison of experimental designs for estimating network-wide delay.

To summarize, in this section we evaluate our measurement framework. Our key findings are:

- Measurement design is crucial for large-scale network monitoring. *A*-optimal is effective in constructing measurement experiments for inferring network-wide average delay. It can achieve 15% inference error by monitoring only 2% paths. Moreover it is also competitive for estimating individual path performance. In addition, it is highly scalable, and can be applied to networks with thousands of routers and end hosts.
- Our results show that the different inference algorithms under study perform similarly for delay inference, whereas the algorithms that enforce nonnegativity constraints perform better for loss inference.

2.4 Summary of IP network measurement and analysis

In this chapter, we develop two path performance measurement and analysis schemes: exact reconstruction of path performance and approximate reconstruction. In the former, we propose a scalable algorithm that, for an overlay of N end hosts, selectively monitors a basis set of only $O(N \log N)$ paths to fully describe all the $O(N^2)$ paths.

In the latter, we build a framework that leverages powerful tools developed in the field of Bayesian experimental design. The framework is highly scalable and can design measurement experiments that span thousands of routers and end hosts. It also is flexible to accommodate a variety of design objectives, such as differentiated, augmented, and multi-user designs.

Chapter 3

Proximity Estimation in Online Social Networks

In the social networks where the relations among nodes are one of the most important aspects in defining the network, many analysis works are based on the proximity measures. As a result, a variety of proximity measures have been proposed. The simplest proximity measures are based on either the shortest graph distance or the maximum information flow between two nodes. One can also define proximity measures based on node neighborhoods (*e.g.*, the number of common neighbors). Finally, several more sophisticated proximity measures involve infinite sums over the ensemble of all paths between two nodes (*e.g.*, Katz measure [40], rooted PageRank [42, 43], and escape probability [67]). Compared with more direct proximity measures such as shortest graph distances and numbers of shared neighbors, path-ensemble based proximity measures incorporate more information about the underlying social structure and have been shown to be more effective in social networks with thousands of nodes [42, 43, 67].

Despite the effectiveness of path-ensemble based proximity measures, it is computationally expensive to summarize the ensemble of all paths between two nodes. The state of the art in estimating path-ensemble based proximity measures (*e.g.*, [67]) typically can only handle social networks with tens of thousands of nodes. As a result, recent works on proximity estimation in large social networks (*e.g.*, [58]) either dismiss path-ensemble based proximity measures due to their prohibitive computational cost or leave it as future work to compare with these proximity measures.

In this chapter, we address the above challenge by developing efficient and accurate techniques to approximate a large family of path-ensemble based proximity measures. We first present proximity embedding, our initial progress on approximating a restricted sub-family of path ensemble based proximity measures on social networks with a few million nodes. Then we go on to outline our ongoing work, clustered graph embedding that aims to provide more generic, scalable, and accurate proximity measurement.

3.1 Problem Formulation

Below we first formally define the classic path-ensemble based proximity measure, *Katz measure*, then show that it can be efficiently estimated by solving a subproblem, the *proximity inversion problem*. In all our discussions below, we model a social network as a graph $G = (V, E)$, where V is the set of nodes, and E is the set of edges. G can be either undirected or directed, depending on whether the social relationship is symmetric.

Katz measure. The Katz measure [40] is a classic path-ensemble based proximity measure. It is designed to capture the following simple intuition: the more paths there are between two nodes and the shorter these paths are the stronger the relationship is (because there are more opportunities for the two nodes to discover and interact with each other in the social network). Given two nodes $x, y \in V$, the Katz measure $\text{Katz}[x, y]$ is a weighted sum of the number of paths from x to y , exponentially damped by length to count short paths more heavily. Formally, we have

$$\text{Katz}[x, y] = \sum_{\ell=1}^{\infty} \beta_{\text{Katz}}^{\ell} \cdot |\text{paths}_{x,y}^{(\ell)}| \quad (3.1)$$

where $\text{paths}_{x,y}^{(\ell)}$ is the set of length- ℓ paths from x to y , and β_{Katz} is a damping factor.

Let A be the adjacency matrix of graph G , where

$$A[x, y] = \begin{cases} 1, & \text{if } \langle x, y \rangle \in E, \\ 0, & \text{otherwise.} \end{cases} \quad (3.2)$$

As shown in [43], the Katz measures between all pairs of nodes (represented as a matrix Katz) can be derived as a function of the adjacency matrix A and the damping factor β_{Katz} as follows.

$$\text{Katz} = \sum_{\ell=1}^{\infty} \beta_{\text{Katz}}^{\ell} A^{\ell} = (I - \beta_{\text{Katz}} A)^{-1} - I \quad (3.3)$$

where I is the identity matrix. Thus, in order to compute Katz , we just need to compute the matrix inverse $(I - \beta_{\text{Katz}} A)^{-1}$.

The proximity inversion problem. The key of estimating path-ensemble based proximity measures is to efficiently compute elements of the following matrix inverse:

$$P \triangleq (I - \beta M)^{-1} = \sum_{\ell=0}^{\infty} \beta^{\ell} M^{\ell} \quad (3.4)$$

where M is a sparse nonnegative matrix with millions of rows and columns, I is an identity matrix of the same size, and $\beta \geq 0$ is a damping factor. We term this common subproblem the *proximity inversion problem*.

3.2 Proximity Embedding

The key challenge in solving the proximity inversion problem (*i.e.*, computing elements of matrix $P = (I - \beta M)^{-1}$) is that while M is a sparse matrix, P is a dense matrix with millions of rows and columns. It is thus computationally prohibitive to compute and/or store the entire P matrix. To address the challenge, we develop a novel dimensionality reduction technique *proximity embedding*, to approximate elements of $P = (I - \beta M)^{-1}$ based on a static snapshot of M :

3.2.1 Preparation

We first present an algorithm to approximate the sum of a subset of rows or columns of $P = (I - \beta M)^{-1}$ efficiently and accurately. We use this algorithm as a basic building block in both proximity sketch and proximity embedding.

Algorithm. Suppose we want to compute the sum of a subset of columns: $\sum_{i \in S} P[*, i]$, where S is a set of column indices. We first construct an indicator column vector v such that $v[i] = 1$ for $\forall i \in S$ and $v[j] = 0$ for $\forall j \notin S$. The sum of columns $\sum_{i \in S} P[*, i]$ is simply Pv and can be approximated as:

$$Pv = (I - \beta M)^{-1} v = \sum_{\ell=0}^{\infty} \beta^\ell M^\ell v \approx \sum_{\ell=0}^{\ell_{\max}} \beta^\ell M^\ell v \quad (3.5)$$

where ℓ_{\max} bounds the maximum length of the paths over which the summation is performed.

Similarly, in order to compute the sum of a subset of rows $\sum_{i \in S} P[i, *]$, we first construct an indicator row vector u such that $u[i] = 1$ for $\forall i \in S$ and $u[j] = 0$ for $\forall j \notin S$. We then approximate the sum of rows $\sum_{i \in S} P[i, *] = uP$ as:

$$uP = u(I - \beta M)^{-1} = \sum_{\ell=0}^{\infty} \beta^\ell u M^\ell \approx \sum_{\ell=0}^{\ell_{\max}} \beta^\ell u M^\ell \quad (3.6)$$

As a special case when S contains only one element, we can approximate a single row or column of P .

Complexity. Suppose M is an m -by- m matrix with n non-zeros. Computing the product of sparse matrix M and a dense vector v takes $O(n)$ time by exploiting the sparseness of M . Therefore, it takes $O(n \cdot \ell_{\max})$ time to compute $\{M^\ell v \mid \ell = 1, \dots, \ell_{\max}\}$ and approximate Pv . Note that the time complexity is independent of the size of the subset S . The complexity for computing uP is identical.

Note however that the above approximation algorithm is *not* efficient for estimating individual elements of P . In particular, even if we only want a single element $P[x, y]$, we have to compute either a complete row $P[x, *]$ or a complete column $P[* , y]$ in order to obtain an estimate of $P[x, y]$. As a result, we only apply the above technique for preprocessing. We will develop several techniques in the rest of this section to estimate individual elements of P efficiently.

Benefits of truncation. We achieve two key benefits by truncating the infinite expansion $\sum_{\ell=0}^{\infty} \beta^{\ell} M^{\ell}$ to form a finite expansion $\sum_{\ell=0}^{\ell_{\max}} \beta^{\ell} M^{\ell}$. First, we completely eliminate the influence of paths with length above ℓ_{\max} on the resulting sums. This is desirable because as pointed out in [42, 43], proximity measures that are unable to limit the influence of overly lengthy paths tend to perform poorly in other applications. Second, we ensure that $\sum_{\ell=0}^{\ell_{\max}} \beta^{\ell} M^{\ell}$ is always finite, whereas elements of $\sum_{\ell=0}^{\infty} \beta^{\ell} M^{\ell}$ may reach infinity when the damping factor β is not small enough.

3.2.2 Proximity Embedding

Our dimensionality reduction technique, *proximity embedding*, applies matrix factorization to approximate P as the product of two rank- r factor matrices U and V :

$$P_{m \times m} \approx U_{m \times r} \cdot V_{r \times m} \quad (3.7)$$

In this way, with $O(2mr)$ total state for factor matrices U and V , we can approximate any $P[x, y]$ in $O(r)$ time as:

$$\hat{P}[x, y] = \sum_{k=1}^r U[x, k] \cdot V[k, y] \quad (3.8)$$

Our technique is motivated by recent research on embedding network distance (*e.g.*, end-to-end round-trip time) into low-dimensional space (*e.g.*, [54, 44, 65, 47]). Note however that proximity is the opposite of distance — the lower the distance the higher the proximity. As a result, techniques effective for distance embedding do not necessarily work well for proximity embedding.

Algorithm. As shown in Figure 3.1(a), our goal is to derive the two rank- r factor matrices U and V based on only a subset of rows $P[L, *]$ and columns $P[* , L]$, where L is a set of indices (which we term the landmark set). We achieve this goal by taking the following five steps:

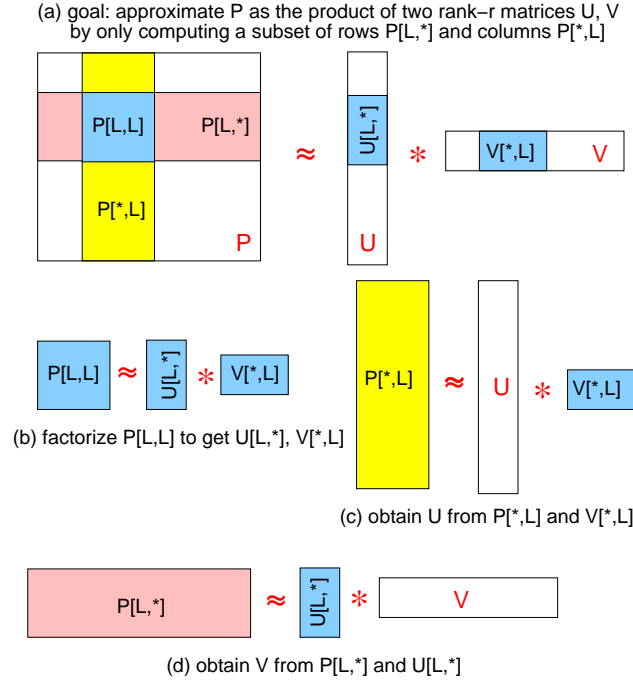


Figure 3.1: Proximity embedding

1. Randomly select a subset of ℓ nodes as the landmark set L . The probability for a node i to be included in L is proportional to the PageRank of node i in the underlying graph¹.
2. Compute sub-matrices $P[L, *]$ and $P[* , L]$ efficiently by computing each row $P[i, *]$ and each column $P[* , i]$ ($i \in L$) separately as described in Section 3.2.1.
3. As shown in Figure 3.1(b), use singular value decomposition (SVD) to obtain the best rank- r approximation of $P[L, L]$:

$$P[L, L] \approx U[L, *] \cdot V[* , L] \quad (3.9)$$

4. Our goal is to find U and V such that $U \cdot V$ is a good approximation of P . As a result, $U \cdot V[* , L]$ should be a good approximation of $P[* , L]$. We can therefore find U such that $U \cdot V[* , L]$ best approximates sub-matrix $P[* , L]$ in least-squares sense (shown in Figure 3.1(c)). With our use of SVD in step 3, the best U is simply

$$U = P[* , L] \cdot V[* , L]^T \quad (3.10)$$

¹We also consider uniform landmark selection, but it yields worse accuracy than PageRank based landmark selection.

5. Similarly, find V such that $U[L, *] \cdot V$ best approximates sub-matrix $P[L, *]$ in least-squares sense (shown in Figure 3.1(d)):

$$V = U[L, *]^T \cdot P[L, *] \quad (3.11)$$

3.2.3 Evaluation

We evaluated our proximity embedding technique with five online social networks: Digg [24], Flickr [31], LiveJournal [46], MySpace [52], and YouTube [70]. We quantify the estimation error using *Relative Error* (defined as $\frac{|est_i - actual_i|}{actual_i}$), where est_i and $actual_i$ denote the estimated and actual values of the proximity measure for node pair i , respectively.

Since it is expensive to compute the actual values over all the data points, we randomly sample 100,000 data points by first randomly selecting 200 rows from the matrix M and then selecting 500 elements from each of these rows. We then compute errors for these 100,000 data points.

We use a damping factor of $\beta = 0.05$, $\ell_{\max} = 6$, and 1600 landmarks unless otherwise specified. For landmark selection, we first compute PageRank for each node and normalize the sum of PageRank of all nodes to 1. Then, we use the normalized PageRank as the probability of assigning a node as a landmark. In this way, nodes with high PageRank are more likely to become landmarks.

Relative errors. Figure 3.2 plots CDF of relative errors using 60 dimensions. We take top 1%, 5%, and 10% of the randomly selected data points and plot each of the selections. In all datasets, we observe that the relative errors are smaller for elements of larger values, which is desirable because larger elements play a more important role in many applications.

Scalability Table 3.1 shows computation time for proximity embedding compared with common neighbor (*e.g.*, # of common friends in the graph), and graph distance (*e.g.*, the shortest path distance). The measurements are taken on an Intel Core 2 Duo 2.33GHz machine with 4GB memory running Ubuntu Linux kernel v2.6.24. First, the query time of both proximity embedding is small, even smaller than common neighbor and graph distance, which are traditionally considered much cheaper operations than computing Katz. Second, the preprocessing of proximity embedding requires computing roughly 50,000 positive and 200,000 negative samples. As the preprocessing can be done in parallel, we use the Condor system [27] to run on 150 machines simultaneously and the actual time taken is only 1/150 of reported time. So even for the largest network, MySpace, it takes within 30 minutes to compute. Furthermore, the pre-processing only needs to be done periodically (*e.g.*, once a few days).

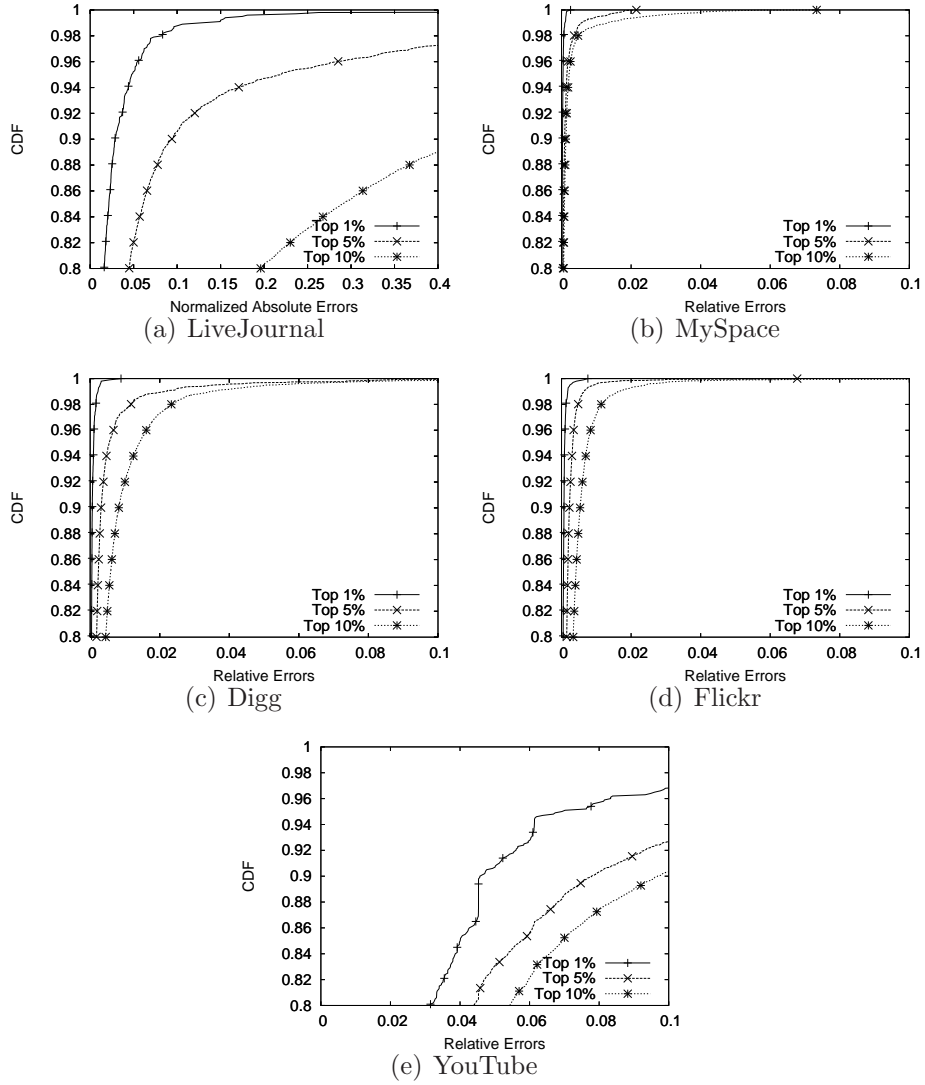


Figure 3.2: Relative errors for top 1%, 5%, and 10% biggest values (Katz metric, $\beta = 0.05$, 1600 landmarks, and 60 dimensions).

Data set	proximity embedding		Common neighbor		Graph distance		
Job type	Prep	Query		Query		Query	
Sample type		Pos	Neg	Pos	Neg	Pos	Neg
LiveJournal	32.0hrs	14.6 μ s	13.7 μ s	546.2 μ s	137.5 μ s	9976.8 μ s	2416.7 μ s
Digg	1.2hrs	0.1 μ s	0.1 μ s	15.0 μ s	12.0 μ s	149.2 μ s	132.5 μ s
MySpace	73.3hrs	58.8 μ s	84.1 μ s	1588.1 μ s	841.8 μ s	50273.3 μ s	41473.2 μ s
Flickr	10.6hrs	47.9 μ s	11.8 μ s	478.9 μ s	118.0 μ s	3111.1 μ s	1720.4 μ s
YouTube	16.3hrs	25.1 μ s	20.6 μ s	251.6 μ s	206.4 μ s	3727.5 μ s	1029.9 μ s

Table 3.1: Computation time of proximity embedding, common neighbor and graph distance.

3.3 Clustered Graph Embedding

We are currently developing an embedding technique that significantly improves the generality, accuracy, scalability, and flexibility of social network analysis. In the later sections, (i) we illustrate a novel technique called *clustered graph embedding* and (ii) we enlist the extensions we propose to work on.

3.3.1 Clustered Graph Embedding

Our key idea is to embed the massive original graph $G = (V, E)$ (*e.g.*, with hundreds of millions of nodes) into a much smaller smaller reduced graph (*e.g.*, with tens of thousands of nodes). The reduced graph preserves essential clustering and spectral information of the original graph and can be utilized for a variety of analysis tasks.

Formalization Let A be an $m \times m$ adjacency matrix of the original graph. For simplicity, we first assume that A is symmetric and later explain how to extend our formalization to the case when A is asymmetric. A graph embedding can be mathematically formalized as the following decomposition:

$$A_{m \times m} \approx U_{m \times n} \cdot D_{n \times n} \cdot U_{m \times n}^T, \quad (3.12)$$

where U is an orthonormal matrix, i.e. $U^T U = I_n$, and D is an $n \times n$ matrix which represents the embedded adjacency matrix of the original graph. Equation (3.12) can be applied to approximate any matrix power A^ℓ as follows:

$$A^\ell \approx (U D U^T)^\ell = U D^\ell U^T. \quad (3.13)$$

As a special case, A^2 gives the number of common neighbors between any pair of nodes, which is a frequently used proximity measure. Note that our previous proximity embedding technique (Section 3.2) cannot approximate the number of common neighbors. So [61] computes it directly, which can quickly become too expensive for networks with hundreds of millions of nodes.

Many functions defined on A can be approximated using sum of matrix powers through Taylor Series expansion. Using Equation (3.13) we can approximate these functions with corresponding functions in D . For the Katz measure as an example,

$$\text{Katz} = (I - \beta_{\text{Katz}} A)^{-1} - I = \sum_{\ell=1}^{\infty} \beta_{\text{Katz}}^\ell A^\ell. \quad (3.14)$$

We can approximate the Katz measure Katz using equations (3.12) and (3.13) as follows:

$$\text{Katz} \approx \sum_{\ell=1}^{\infty} U(\beta_{\text{Katz}}^{\ell} D^{\ell})U^T = U \left(\sum_{\ell=1}^{\infty} \beta_{\text{Katz}}^{\ell} D^{\ell} \right) U^T = U \left((I_n - \beta_{\text{Katz}} D)^{-1} - I_n \right) U^T. \quad (3.15)$$

Basic Algorithm When m is not too large, a simple way of choosing U is through eigendecomposition. The best approximation of A is given by $A \approx U\Lambda U^T$, where Λ is a diagonal matrix whose diagonal consists of the n largest in magnitude eigenvalues of A , and columns of U are the corresponding eigenvectors. Unfortunately, the computational complexity and memory requirement of this approximation through the eigendecomposition increases rather quickly as the network size grows, even if we take advantage of the sparsity of A . On the other hand, our experience suggests that with small n , spectral graph embedding cannot capture sufficient social/network structure for very large social networks and therefore yields poor approximation accuracy.

To address the scalability challenge, we propose to combine clustering with spectral graph embedding. Specifically, our new *clustered graph embedding* algorithm involves the following three steps:

1. Graph clustering

Apply a fast graph clustering algorithm to partition the original graph $G = (V, E)$ into a set of C non-overlapping clusters: V_1, V_2, \dots, V_C , such that $\cup_{c=1}^C V_c = V$. Let $m_i = |V_i|$. We use GRACLUS [41] to perform clustering, which directly minimizes the original graph partition objective through a variant of K-means [26] combined with a multilevel approach [23], and therefore avoids the explicit calculation of eigenvectors. The graph clustering we employ is highly scalable to be able to handle networks with hundreds of millions of nodes.

2. Per-cluster graph embedding

For each cluster i , let $A_i = A[V_i, V_i]$ be the submatrix of A with row and column indices given by set V_i . A_i represents the adjacency matrix for cluster i . Perform eigendecomposition on A_i to obtain n_i eigenvectors corresponding to n_i eigenvalues with largest magnitude: $A_i \approx U_i D_i U_i^T$, where columns of U_i are orthogonal, and matrix D_i is diagonal.

3. Global graph embedding

Create a block diagonal matrix U that takes U_i as its i -th diagonal block ($i = 1, \dots, C$). Formally, let $C_i = \{\sum_{j=1}^{i-1} n_j + 1, \dots, \sum_{j=1}^i n_j\}$. Then $U = [V_i, C_i] = U_i (i = 1, \dots, C)$.

All other elements in U are zeros. We then derive D in Equation (3.12) as $D = U^T A U$. The block diagonal structure of U and the orthogonality of U_i 's columns ensure the columns of U are orthogonal.

Advantages Our clustered spectral graph embedding algorithm has several advantages: (i) It is highly scalable. Specifically, intra-cluster eigendecomposition is highly scalable because each cluster is much smaller than the original graph. Meanwhile, the graph clustering step is also highly scalable due to recent development of software such as GRACLUS. In addition, since U is a block diagonal matrix, the total storage required for U is only $\sum_i m_i n_i$, which is much smaller than the storage for a dense matrix of the same size. (ii) Compared with global eigendecomposition, the reduced graph can have much larger size and thus capture much more structural information about the underlying social network. (iii) The embedding preserves both intra-cluster spectral information and the global inter-cluster relationship and is thus more accurate than clustering alone. (iv) Unlike our own proximity embedding in [61], it is general and flexible and can be used to approximate different functions defined on A without having to construct different U .

3.3.2 Extensions

The basic algorithm will be further extended to support (i) incremental update of graph embedding, (ii) support for asymmetric adjacency matrix, and (iii) Supervised link prediction.

- **Incremental update**

As massive online social networks are highly dynamic, with hundreds of thousands of new users and millions of new links added daily. It is thus desirable to be able to inexpensively update the graph embedding equation $A \approx U D U^T$ under the new adjacency matrix $A' = A + \Delta A$ when ΔA has only few non-zero elements (*i.e.*, when the interval between updates is short). A simple strategy is to keep U relatively stable while updating D frequently with $D' = D + \Delta D$, where $\Delta D = U^T \Delta A U$ can be efficiently computed due to the sparsity of ΔA . In fact, D can be updated continuously near real-time whenever A changes. U only needs to be recomputed when the cumulative changes to A become too significant.

- **Asymmetric adjacency matrix A**

We can perform eigendecomposition on an asymmetric A . However, when A is asymmetric, its eigenvectors are no longer orthogonal. In this case, we need to replace

Equation (3.12) with the following asymmetric graph embedding equation:

$$A_{m \times m} \approx U_{m \times n} \cdot D_{n \times n} \cdot U_{m \times n}^T, \quad (3.16)$$

where U is a block diagonal matrix containing column eigenvectors of each A_i , V is a block diagonal matrix containing row eigenvectors of each A_i (i.e., column eigenvectors of A_i^T). U and V are further scaled properly so that $V^T U = I_n$. We can prove that the asymmetric graph embedding has properties very similar to the symmetric graph embedding.

- **Supervised link prediction**

The task of link prediction is to predict the network edges that will be added to the network in the future. We can potentially leverage the graph embedding to improve the accuracy of link prediction. Specifically, we apply the supervision on past snapshots of networks using clustered graph embedding and learn the optimal parameters of proximity metric for links created in the future. For this we first decompose D as $D = W_{m \times r} \cdot W \Lambda_{r \times r} \cdot W_{m \times r}^T$, where $r \ll n$ and Λ is an $r \times r$ diagonal matrix containing D 's eigenvalues and W is an $n \times r$ matrix containing eigenvectors. The Katz measure in Equation (3.15) can be approximated as

$$\text{Katz}_{m \times m} \approx U_{m \times n} \cdot W_{m \times r} \cdot \Sigma_{r \times r} \cdot W_{m \times r}^T \cdot U_{m \times n}^T, \quad (3.17)$$

where Σ is a $r \times r$ diagonal matrix. We propose to apply the supervised learning in the determination of this Σ to automatically find its optimal diagonal components.

Chapter 4

Assessment of service quality in IPTV networks

As ongoing work, we are engaged in the design and implementation of a novel service network management scheme. Our target application is a large commercial IPTV system. The goal of our system is to accurately and scalably assess the quality of audio and video data streamed to the end users, and be able to predict possible future occurrences of video quality problems.

As a first step towards the goal, we conducted an analysis of set-top box (STB) crashes to gain insight into service problems in IPTV system. STBs are computer boxes that sits next to TVs and interface to both the user and the network to deliver TV programs IPTV offers. STB crash is an extreme case of video quality problem in that a crash incurs minutes of noticeable service interruptions to the customers. In the analysis, we applied data mining techniques to correlate STB crash events with various parts of IPTV system including the network events, user activities, and video stream contents. Our key findings are: (i) Network events and STB crashes have no significant correlation. (ii) Frequent user activities (*e.g.*, repeated video stream controls) are not closely related to the crashes. But Correlating STB crashes with the extensive usage over a long period of time, we find the STB crash rate can increase over the duration of power cycle. (iii) The correlation between the crashes and video stream content suggests a strong possibility that impaired video streams may cause STB crashes.

We believe that the data-driven analysis that we demonstrated in the preliminary work is applicable beyond the specific context of troubleshooting STB crashes. In the proposed work, we plan to design and implement a macroscopic data-driven service management that provides the followings in the context of commercial IPTV service:

- **Scalable aggregation of performance indicators**

We plan to first identify the events that reflects the subscribers of IPTV experiencing video quality problems (*i.e.*, the service level impairments). Such events include customer trouble tickets, STB crash log, STB diagnosis logs, and user activity information. We then determine a set of key performance indicators of video quality from a vast set of measurements from diverse parts of IPTV system. The data set we collect

to diagnose the user-perceived video quality begins from the system logs from set-top boxes at residential homes, network device logs from various level of the multicast hierarchy, and the video source information from video encoders, encrypters, and buffering servers. We plan to provide a method that effectively aggregates the key performance indicators across various service levels and different regions by considering the specifics of the topological structure.

- **Accurate prediction of future service performance degradation**

With the integrated metric that summarizes various performance indicators, we further extend our management scheme to predict possible outbreak of video quality problems in near future. Learning from the historical trend of the data, we plan to apply various class of linear and non-linear regression methods and achieve to present a highly accurate Quality of Service (QoS) predictor.

Bibliography

- [1] <http://www.bittorrent.com>.
- [2] CISCO Telepresence. <http://www.cisco.com/telepresence>.
- [3] Monitoring and Managing Avaya IP Telephony Traffic. <http://www.avaya-apac.com/downloads/ipt/lb2554dev.pdf>.
- [4] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *18th ACM SOSP*, 2001.
- [5] D. G. Andersen et al. Resilient overlay networks. In *Proc. of ACM SOSP*, 2001.
- [6] E. Anderson et al. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.
- [7] At&t uverse. <http://www.att.com/u-verse/>.
- [8] A. L. Barabasi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaboration. *Physica A: Statistical Mechanics and its Application*, 2002.
- [9] R. M. Bell, Y. Koren, and C. Volinsky. Chasing \$1,000,000: How we won the Netflix Progress Prize. *Statistical Computing and Statistical Graphics Newsletter*, 18(2):4–12, 2007.
- [10] T. Bu, N. Duffield, F. Presti, and D. Towsley. Network tomography on general topologies. In *ACM SIGMETRICS*, 2002.
- [11] Consumer electronic show - next big thing supersession, 2010. <http://ces.cnet.com/next-big-thing/>.
- [12] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10:273–304, 1995. <http://citeseer.ist.psu.edu/chaloner95bayesian.html>.
- [13] Y. Chen, D. Bindel, and R. H. Katz. Tomography-based overlay network monitoring. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, 2003.
- [14] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 55–66, New York, NY, USA, 2004. ACM Press.

- [15] D. B. Chua, E. D. Kolaczyk, and M. Crovella. Efficient monitoring of end-to-end network properties. In *Proc. of IEEE INFOCOM*, Jul. 2004.
- [16] D. B. Chua, E. D. Kolaczyk, and M. Crovella. Efficient monitoring of end-to-end network properties. In *IEEE INFOCOM*, 2005.
- [17] M. A. Clyde. Experimental design: A Bayesian perspective. *International Encyclopedia Social and Behavioral Sciences*, Apr. 2001.
- [18] C. Cortes, D. Pregibon, and C. T. Volinsky. Communities of interest. *Intelligent Data Analysis*, 2002.
- [19] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. New York: Wiley, 1991.
- [20] J. Davidsen, H. Ebel, and S. Bornholdt. Emergence of a small world from local interactions: Modeling acquaintance networks. *Physical Review Letters*, 2002.
- [21] T. Davis. LDL: A sparse LDL' factorization and solve package (version 1.2), 2005. <http://www.cise.ufl.edu/research/sparse/ldl/>.
- [22] J. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.
- [23] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):1944–1957, 2007.
- [24] Digg. <http://www.digg.com>.
- [25] D. Donoho. For most large underdetermined systems of linear equations, the minimal l_1 -norm near-solution approximates the sparsest near-solution. <http://www-stat.stanford.edu/~donoho/Reports/2004/l1l0approx.pdf>.
- [26] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, New York, second edition, 2001.
- [27] D. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. A worldwide flock of Condors: Load sharing among workstation clusters. *Future Generation Computer Systems*, 1996.
- [28] Facebook. <http://www.facebook.com>.
- [29] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationship of the Internet topology. In *ACM SIGCOMM*, 1999.
- [30] V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.
- [31] Flickr. <http://www.flickr.com>.
- [32] G. Golub and C. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, 1989.

- [33] G. H. Golub, V. Klema, and G. W. Stewart. Rank degeneracy and least squares problems. Technical Report CS-TR-76-559, Stanford University, Aug. 1976. <http://www-db.stanford.edu/TR/CS-TR-76-559.html>.
- [34] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, 2nd edition, 1989.
- [35] R. Govindan and H. Tangmunarunkit. Heuristics for Internet map discovery. In *IEEE INFOCOM*, 2000.
- [36] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, 1997.
- [37] S. Hill, F. Provost, and C. Volinsky. Network-based marketing: Identifying likely adopters via consumer networks. *Statistical Science*, 2006.
- [38] E. M. Jin, M. Girvan, and M. E. J. Newman. The structure of growing social networks. *Physical Review Letters*, 2001.
- [39] B. Jones, D. Lin, and C. Nachtsheim. Bayesian D-optimal supersaturated designs, 2004. Submitted for publication.
- [40] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 1953.
- [41] B. Kulis and Y. Guan. Graclus—Efficient graph clustering software for normalized cut and ratio association on undirected graphs, 2008. 2010/01/26 (version 1.2).
- [42] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proc. of Conference on Information and Knowledge Management (CIKM)*, 2003.
- [43] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 2007.
- [44] H. Lim, J. Hou, and C. H. Choi. Constructing Internet coordinate system based on delay measurement. In *Proc. of IMC*, 2003.
- [45] D. V. Lindley. *Bayesian Statistics – A Review*. SIAM, Philadelphia, PA, 1972.
- [46] LiveJournal. <http://www.livejournal.com>.
- [47] Y. Mao and L. K. Saul. Modeling distances in large-scale networks by matrix factorization. In *Proc. of IMC*, New York, NY, USA, 2004.
- [48] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proc. of ACM SIGCOMM*, Aug. 2002.

- [49] A. J. Miller and N. K. Nguyen. A Fedorov exchange algorithm for D-optimal design. *Applied Statistics*, 1994.
- [50] A. Mislove, K. P. Gummadi, and P. Druschel. Exploiting social networks for Internet search. In *Proc. of HotNets-V*, 2006.
- [51] T. J. Mitchell. An algorithm for the construction of D-optimal designs. *Technometrics*, 1974.
- [52] MySpace. <http://www.myspace.com>.
- [53] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 2003.
- [54] T. E. Ng and H. Zhang. Predicting internet network distance with coordinate-based approaches. In *Proc. of INFOCOMM*, 2002.
- [55] N. K. Nguyen and A. J. Miller. A review of exchange algorithms for constructing discrete D-optimal designs. *Computational Statistics and Data Analysis*, 1992.
- [56] I. Research. Global market analysis, 2008. <http://www.imsresearch.com>.
- [57] S. Rhea, B. Godfrey, B. Karp, J. Kubiawicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. OpenDHT: A Public DHT Service and Its Uses. In *ACM SIGCOMM 2005*.
- [58] P. Sarkar, A. W. Moore, and A. Prakash. Fast incremental proximity search in large graphs. In *Proc. of ICML*, 2008.
- [59] M. Saunders. PDCO: Primal-Dual interior method for Convex Objectives, 2003. <http://www.stanford.edu/group/SOL/software/pdco.html>.
- [60] Skype. <http://www.skype.com>.
- [61] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu. Scalable proximity estimation and link prediction in online social networks. In *IMC*, 2009.
- [62] H. H. Song, L. Qiu, and Y. Zhang. NetQuest: A flexible framework for large-scale internet measurement. In *ACM SIGMETRICS*, Saint-Malo, France, 2006.
- [63] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. In *ACM SIGCOMM*, 2002.
- [64] L. Subrmanian, S. Agarwal, J. Rexford, and R. H.Katz. Characterizing the Internet hierarchy from multiple vantage points. In *IEEE INFOCOM*, 2002.
- [65] L. Tang and M. Crovella. Virtual landmarks for the Internet. In *Proc. of IMC*, 2003.
- [66] H. Tangmunarunkit et al. Network topology generators: Degree-based vs structural. In *ACM SIGCOMM*, 2002.

- [67] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. In *Proc. of KDD*, 2007.
- [68] L. Wang, K. Park, R. Pang, V. S. Pai, and L. Peterson. Reliability and security in the codeen content distribution network. In *USENIX*, 2004.
- [69] Wikipedia. Social network. http://en.wikipedia.org/wiki/Social_network.
- [70] YouTube. <http://www.youtube.com>.
- [71] Y. Zhang et al. On the constancy of Internet path properties. In *Proc. of SIGCOMM IMW*, 2001.
- [72] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *Proc. Internet Measurement Conference*, Oct. 2005.
- [73] 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*, Jun. 2003.
- [74] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proc. of ACM SIGCOMM*, Aug. 2003.