NetQuest: A Flexible Framework for Large-Scale Network Measurement

Han Hee Song, Lili Qiu, and Yin Zhang
University of Texas at Austin
Austin, TX 78712, USA
{hhsong,lili,yzhang}@cs.utexas.edu

ABSTRACT

In this paper, we present NetQuest, a flexible framework for large-scale network measurement. We apply *Bayesian experimental design* to select active measurements that maximize the amount of information we gain about the network path properties subject to given resource constraints. We then apply *network inference* techniques to reconstruct the properties of interest based on the partial, indirect observations we get through these measurements.

By casting network measurement in a general Bayesian decision theoretic framework, we achieve *flexibility*. Our framework can support a variety of design requirements, including (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 who are interested in different parts of the network. Our framework is also *scalable* and can design measurement experiments that span thousands of routers and end hosts.

We develop a toolkit that realizes the framework on PlanetLab. We conduct extensive evaluation using both real traces and synthetic data. Our results show that the approach can accurately estimate network-wide and individual path properties by only monitoring within 2-10% of paths. We also demonstrate its effectiveness in providing differentiated monitoring, supporting continuous monitoring, and satisfying the requirements of multiple users.

Categories and Subject Descriptors

C.2.3 [**Computer-Communications Networks**]: Network Operations—*network monitoring*; C.2.5 [**Computer-Communications Networks**]: Local and Wide-Area Networks—*Internet*

General Terms

Measurement, Performance

Keywords

Network Measurement, Bayesian Experimental Design, Network Inference, Network Tomography

1. INTRODUCTION

Network measurement is essential to a wide variety of existing and emerging network applications, such as ISP performance management, traffic engineering, content distribution, overlay routing, and peer-to-peer applications. For example, ISPs and enterprise networks put increased focus on network performance, and demand capabilities for detailed performance measurement in networks of hundreds or even thousands of nodes. Performance monitoring also becomes a critical capability that allows overlays and peer-to-peer networks to detect and react to changing network conditions.

While much progress has been made in network measurement, two significant challenges remain. First, large-scale network management applications often require the ability to efficiently monitor the whole network. The quadratic growth in the number of network paths with respect to the number of network nodes makes it impractical to measure every path. Second, existing techniques are often tailored to specific application needs, and thus lack the flexibility to accommodate applications with different requirements.

To address these challenges, in this paper we develop NetQuest, 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. Making the experimental design applicable to such context involves addressing several challenges. First, it is not clear how to formulate the problem of designing network measurement under the Bayesian experimental design framework. Second, the traditional Bayesian experimental design often targets at a single application. In our environment, there can be many applications with different design requirements. How to use Bayesian experimental design to support such diverse application requirements is an interesting open problem.

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.

Based on observations obtained from the measurements, we then use inference techniques to accurately reconstruct the global view

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGMetrics/Performance'06, June 26–30, 2006, Saint Malo, France.
Copyright 2006 ACM 1-59593-320-4/06/0006 ...$5.00.

of the network without requiring complete information. Our results show that our measurement framework can estimate network-wide average path delay within 15% error by monitoring within 2% paths. It achieves a similar degree of accuracy for estimating individual path properties by monitoring 10% paths. In addition, we demonstrate the flexibility of our measurement framework in providing differentiated monitoring, supporting continuous monitoring, and satisfying the requirements of multiple users.

Contributions: This paper makes three main contributions. First, our work brings Bayesian experimental design into large-scale network measurement. While Bayesian experimental design has found many applications in other scientific fields, to the best of our knowledge, this is the first time that it is applied to designing active network measurement experiments.

Second, building on top of Bayesian experimental design and inference techniques, we develop a unified framework within which a large class of network performance inference problems can be modeled, solved, and evaluated. Our framework is flexible, and can accommodate different design requirements. Our framework is also scalable, and can design measurement experiments that span thousands of routers and end hosts.

Third, we develop a toolkit that implements our framework on PlanetLab [23]. Using the toolkit we conduct an extensive evaluation of our framework for efficient monitoring of end-to-end network performance. Our results demonstrate the effectiveness and flexibility of our framework.

Outline: The rest of this paper is organized as follows. We describe the large-scale network measurement problem in Section 2. In Section 3, we introduce our approach to design measurement experiments. In Section 4, we present several network inference algorithms. We describe our toolkit development in Section 5. In Section 6 and Section 7, we present our evaluation methodology and results. Finally, we conclude in Section 8.

2. PROBLEM FORMULATION

In this paper, we focus on monitoring end-to-end performance in large 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*).

Formally, the problem can be specified as follows.

$$\mathbf{y} = A\mathbf{x}, \tag{1}$$

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.*,

$$A_{ij} = \begin{cases} 1 & \text{if path } i \text{ contains link } j, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

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 $\log\{1 - \text{loss rate}\}$. 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 [17, 27, 28]. In this paper, 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 $f(\mathbf{x}) = \frac{1}{m}[1, 1, ..., 1]_{1\times m}A\mathbf{x}$, 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 the linear relationship $\mathbf{y}_S = A_S\mathbf{x}$.

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

$$f(\mathbf{x}) = F\mathbf{x}, \tag{3}$$

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. We will present our approach for designing measurement experiments in Section 3, and describe network inference algorithms in Section 4.

3. 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 in several ways:

- First, the design space is often quite large, involving a number of control variables. Control variables in network measurement include: choosing the sites to launch experiments from, choosing the subset of paths/links to probe, choosing the type of network characteristics to measure, choosing how to randomize, choosing the granularity, frequency and duration of each experiment, etc. These are all relevant aspects in the design.
- Second, the design is often subject to all kinds of constraints imposed by the network and operations, such as the accessibil-

ity of measurement infrastructure, the availability of network resources, and the operational policies and restrictions.

- Third, the design needs to be tailored to accommodate multiple (sometimes conflicting) design objectives. Different users (*e.g.*, VoIP gateway services, versus cable access network providers) often have different notions of performance, and want to monitor different metrics (*e.g.*, delay, loss, or bandwidth).

Given the complexity involved in experimental design, 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 [1, 6] 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.

3.1 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 [15, 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 [1], 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

$$U(\eta) = \int_{\mathcal{Y}} \max_{d\in D} \int_{\mathcal{X}} U(d, \mathbf{x}, \eta, \mathbf{y})\, p(\mathbf{x}|\mathbf{y}, \eta)\, p(\mathbf{y}|\eta) d\mathbf{x}d\mathbf{y}, \quad (4)$$

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 $U(\eta)$:

$$U(\eta^*) = \max_{\eta\in\mathcal{H}} \int_{\mathcal{Y}} \max_{d\in D} \int_{\mathcal{X}} U(d, \mathbf{x}, \eta, \mathbf{y}) p(\mathbf{x}|\mathbf{y}, \eta) p(\mathbf{y}|\eta) d\mathbf{x}d\mathbf{y}. \quad (5)$$

3.1.1 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

$$U_A(\eta) = -\int (F\mathbf{x} - F\hat{\mathbf{x}})^T (F\mathbf{x} - F\hat{\mathbf{x}}) p(\mathbf{y}, \mathbf{x}|\eta) d\mathbf{x}d\mathbf{y}, \quad (6)$$

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

$$U_A(\eta) = -\sigma^2 \operatorname{tr}\{FD(\eta)F^T\}, \quad (7)$$

where $\operatorname{tr}\{M\}$ (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

$$\phi_A(\eta) = \operatorname{tr}\{FD(\eta)F^T\}, \quad (8)$$

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:

$$\int \log \frac{p(F\mathbf{x}|\mathbf{y}, \eta)}{p(F\mathbf{x})} p(\mathbf{y}, \mathbf{x}|\eta) d\mathbf{x}d\mathbf{y}. \quad (9)$$

Since the prior distribution of $F\mathbf{x}$ does not depend on the design η, maximizing (9) is equivalent to maximizing

$$U_D(\eta) = \int \log\{p(F\mathbf{x}|\mathbf{y}, \eta)\} p(\mathbf{y}, \mathbf{x}|\eta) d\mathbf{x}d\mathbf{y}. \quad (10)$$

Under a normal linear model, the Bayesian procedure yields

$$U_D(\eta) = -\frac{n}{2}\log(2\pi) - \frac{n}{2} - \frac{1}{2}\log\det\{FD(\eta)F^T\}, \quad (11)$$

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

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

$$\phi_D(\eta) = \det\{FD(\eta)F^T\}, \quad (12)$$

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

One problem with (12) is that when $\operatorname{rank}(F) < r$ (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

$$\phi_D(\eta) = \Pi_{\operatorname{rank}(F)}\{FD(\eta)F^T\}, \quad (13)$$

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

```
1  S = ∅                          // 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 ∈ {1, 2, ··· , m} − S
4          // compute the new design criterion after adding path i
5          criteria[i] = φ(S ∪ {i})
6      end for
7      // select the path that minimizes the new design criterion
8      S = S ∪ {arg min_i criteria[i]}
9  end for
10 return S
```

Figure 1: Sequential path selection for Bayesian designs.

it full-rank. Therefore, we only need to consider the case when $\text{rank}(F) = \min(r, n)$. In this case, we can simplify (13) into

$$\phi_D(\eta) = \begin{cases} \det\{F^T F\} \det\{D(\eta)\} & \text{if } r \geq n, \\ \det\{F D(\eta) F^T\} & \text{otherwise.} \end{cases} \quad (14)$$

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

3.1.2 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 1.

It is possible to further improve the design obtained by the sequential search algorithm using an *exchange algorithm* (*e.g.*, [10, 19, 18, 21]), 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 [21]. 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.*

3.1.3 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 of the above search algorithm is computing the new design criterion $\phi(S \cup \{i\})$ after adding a path i (line 5 in Figure 1). 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. *Note that understanding of these methods is not required in order to understand the remainder of this paper.*

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

$$(M')^{-1} = (M + \mathbf{a}_i \mathbf{a}_i^T)^{-1} = M^{-1} - \alpha \mathbf{u}\mathbf{u}^T, \quad (15)$$

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

Combine (15) 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

$$\text{tr}\{N'\} = \text{tr}\{N\} - \text{tr}\{\alpha(F\mathbf{u})(F\mathbf{u})^T\} = \text{tr}\{N\} - \alpha \|F\mathbf{u}\|_2^2. \quad (16)$$

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

$$(N')^{-1} = (N - \mathbf{b}\mathbf{b}^T)^{-1} = N^{-1} + \beta \mathbf{v}\mathbf{v}^T, \quad (17)$$

$$\det\{N'\} = \det\{N - \mathbf{b}\mathbf{b}^T\} = \frac{1}{\beta} \det\{N\}, \quad (18)$$

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 1). This takes $O(n^2)$ operations using (15) and (17). 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 [8], 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}.

3.2 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

$$D(\eta) = (A_{S\cup S_0}^T A_{S\cup S_0} + R)^{-1} = (A_S^T A_S + R + A_{S_0}^T A_{S_0})^{-1} \quad (19)$$

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 3.3, 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 $1, 2, \cdots, u$, 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

$$\phi(\eta) = \sum_i w_i \text{tr}\{F_i D(\eta) F_i^T\} = \text{tr}\{F D(\eta) F^T\}, \quad (20)$$

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). Due to space limit, we will not consider such *constrained designs* in this paper. As part of our future work, we plan to further investigate them in the context of specific network monitoring applications.

3.3 Non-Bayesian Designs

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

Rank based solution: Chen *et al.* [2, 3] presented a linear algebraic approach to efficient monitoring of overlay paths. In their context, given k overlay nodes, there are $k(k-1)$ different paths. So A has $k(k-1)$ rows. The quantity of interest is $f(\mathbf{x}) = A\mathbf{x}$ (*i.e.*, the performance on all overlay paths). Let rank(A) denote the rank of matrix A. Their solution is based on the observation that any subset of rank(A) independent rows of A, denoted as A_S, are sufficient to span the space of A: $\{\mathbf{y} \in \mathbb{R}^m | \exists \mathbf{x} \in \mathbb{R}^n \text{ s.t. } \mathbf{y} = A\mathbf{x}\}$. 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 will show in Section 7, rank(A) gives an upper bound on the number of paths that one needs to probe.

SVD based solution: Chua *et al.* [4, 5] 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 paper.

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: 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: $QR = G[\cdot, E]$ (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 3: QR based path selection algorithm

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^TA 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 [12, Ch 12]). So [4] adapted the method described in Algorithm 12.2.1 of [12, p. 574], which is based on the singular value decomposition (SVD). Subsequently in [5], Chua *et al.* extends their algorithm to incorporate Σ, the covariance 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.

Path selection under general link covariance Σ (*e.g.*, when link performance has spatial dependency) is left as an *open* problem in [5]. 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.* [11] for selecting *numerically* independent rows/columns (the other algorithm is the SVD based solution described above). As noted in [11], the QR based solution is generally more efficient than the SVD based solution and often achieves comparable performance. We extend the algorithm in [11] to incorporate the link covariance matrix Σ when Σ is a diagonal matrix. The resulting algorithm is illustrated in Figure 3.

Summary: Rank based solution requires monitoring 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.

4. 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 (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 (1) is often *under-determined*. A lot of solutions have been developed for under-determined linear inverse problems. As we noted in [28],

Notation	Inference algorithm	Section
MinL2	L_2 norm minimization	§4.1
MinL1	L_1 norm minimization	§4.2
Entropy	Maximum Entropy Estimation	§4.3

Table 1: Inference algorithms. MinL2 and MinL1 can optionally incorporate the nonnegativity constraints: $\mathbf{x} \geq 0$, resulting in MinL2_nonNeg and MinL1_nonNeg, respectively.

many such proposals solve the regularized least-squares problem

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 J(\mathbf{x}), \tag{21}$$

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 [13] for a general description of regularization.

In this paper, we compare the accuracy of several widely used inference algorithms (summarized in Table 1). The goal is to understand how combinations of different experimental design methods and inference algorithms affect the overall inference accuracy.

4.1 L_2 Norm Minimization

A common solution to (1) is L_2 *norm minimization*, which corresponds to (21) with $J(\mathbf{x}) = \|\mathbf{x}\|_2^2$.

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x}\|_2^2. \tag{22}$$

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 (22), resulting in

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x} - \mu\|_2^2. \tag{23}$$

(23) is also commonly referred to as the Tikhonov regularization [13]. 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 (23). The resulting nonnegative linear least-squares problem can be solved using PDCO [24], a MATLAB package highly optimized for problems with sparse matrices A.

4.2 L_1 Norm Minimization

Another common solution to (1) is L_1 *norm minimization*, which corresponds to (21) with $J(\mathbf{x}) = \|\mathbf{x}\|_1$ (*i.e.*, the L_1 norm of $\mathbf{x}$).

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x}\|_1. \tag{24}$$

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 [9], 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 [26].

As in [26], we solve the following variant of (24)

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_1 + \lambda \|\mathbf{x}\|_1. \tag{25}$$

An advantage of (25) 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 (25), yielding

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_1 + \lambda \|\mathbf{x} - \mu\|_1. \tag{26}$$

4.3 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 (21).

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \sum_i x_i \log x_i, \quad \mathbf{x} \geq 0. \tag{27}$$

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 [7], resulting in

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \sum_i x_i \log \frac{x_i}{\mu_i}, \quad \mathbf{x} \geq 0. \tag{28}$$

(28) can be efficiently solved by PDCO [24], which has been highly optimized for sparse matrices A. We have successfully applied (28) in the context of traffic matrix estimation [28].

5. TOOLKIT DEVELOPMENT

We develop a toolkit on the PlanetLab [23] to measure and infer network path properties. Our toolkit is programmed in MATLAB, Perl, and C++, altogether with around 25,000 lines of code. The toolkit design is based on master-slave model. The master accepts measurement requests from users, designs measurement experiments, issues measurement commands to the slaves, and collects and analyzes the results. The slaves accept measurement commands from the master, conduct measurements, gather the results and return them back to the master. While our current implementation is based on one master and multiple slaves, our architecture is extensible to multiple masters and multiple slaves.

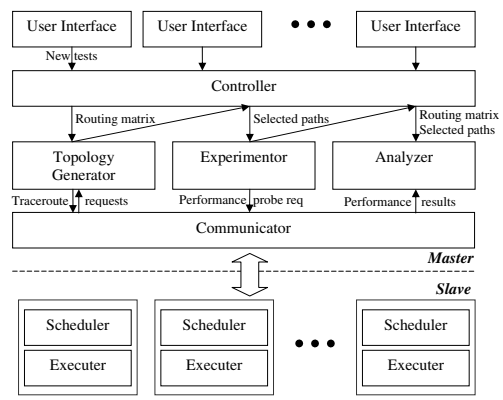


Figure 4: Our toolkit architecture.

As shown in Figure 4, the master consists of the following five modules: (i) controller, which accepts user measurement requests, schedules jobs, and manages the other master modules, (ii) topology manager, which generates and maintains the routing matrix, (iii) experimenter, which applies one of the experimental design algorithms in Section 3 to identify which paths to probe, and issues measurement requests to the corresponding slave nodes, (iv) analyzer, which applies one of the inference algorithms in Section 4 to infer the network performance based on the obtained measurement data, and (v) communication module, which takes care of communication between slaves and masters.

Slave side of the toolkit accepts measurement commands both from topology manager and experimenter. The request is queued at

its scheduler, and executed according to the specified frequency and duration. When a set of measurement experiments has finished, the results are sent back to the communication module of the master. For safety and convenience, we use scriptroute [25] for conducting traceroute. The toolkit runs continuously to measure and infer performance on the paths specified by the users.

6. EVALUATION METHODOLOGY

6.1 Accuracy Metric

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

$$\text{normalized } MAE = \frac{\sum_i |inferred_i - actual_i|}{\sum_i actual_i}, \qquad (29)$$

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.

6.2 Dataset Description

We evaluate our approach using both real traces and synthetic data. *Note that the real traces use round-trip performance measurements, whereas the synthetic data use one-way performance measurements.* As noted in Section 2, the problem formulation $\mathbf{y} = A\mathbf{x}$ works for both one-way and round-trip measurements.

PlanetLab traces: We collected RTT traces among PlanetLab using our toolkit on Oct. 1, 2005 for 10 minutes, with a probing frequency of one probe per second for every monitored path. We also collected loss traces on PlanetLab on Jan. 22, 2006 for 15 minutes, with a probing frequency of one probe per 300 milliseconds for every monitored path. Table 2 summarizes the 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.

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

Table 2: Summary of PlanetLab traces used for evaluation.

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 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 4 to infer based on the observed performance of monitored paths. For each interval, we use normalized *MAE* to quantify the inference error.

Synthetic data: To further test the scalability of our approach, we generate synthetic network topologies using BRITE topology generator [16]. Table 3 summarizes the topologies we use. BRITE topology generator assigns each link with a delay based on its physical distance. In addition, we further assign a loss rate to each link in the following way as in [3, 22]. A fraction of links were classified as "good", and the rest as "bad". The loss rate for a good link is picked uniformly at random in the 0-1% range and that for a bad link is picked in the 5-10% range. Once each link has been assigned a loss rate, we derive the true loss rate for each path. Then we use Bernoulli or Gilbert loss process at each path to derive the observed loss rate. In the Bernoulli model, each packet is dropped with a fixed probability determined by the loss rate of the path. In the Gilbert case, the path moves between a good state and a bad state, where no packets are dropped at the good state and all packets are dropped at the bad state. Following [20, 22], we use 35% as the probability of remaining in the bad state. The other state-transition probabilities are determined to match the average loss rate with the loss rate assigned to the link. In both cases, the end-to-end loss rate is computed based on the transmission of 10000 packets. Our evaluation shows that the inference accuracy is similar under Bernoulli and Gilbert loss models. So in the interest of brevity, we only show the results from Gilbert loss models.

	# nodes	# overlay nodes	# paths	# links	rank(A)
Brite-n1000-o200	1000	200	39800	2883	2051
Brite-n5000-o600	5000	600	359400	14698	9729

Table 3: Summary of synthetic data used for evaluation.

6.3 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 nonstationarities 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 7). 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 [14].

Prior information for network inference (μ): In this paper, unless noted otherwise, we assume no prior information about $\mathbf{x}$. That is, we set $\mu = \mathbf{0}$ (an all-0 vector) for L_2 and L_1 norm minimization (both with and without nonnegativity constraints), and $\mu = \mathbf{1}$ (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.

In our future work, we plan to develop light-weight techniques to obtain better priors and thus further improving the accuracy. As an initial step, in Section 7.3 we evaluate a simple enhanced prior that does not require generating any extra probing traffic.

Regularization parameter (λ): Our experience [28, 26] 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.

7. 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.

7.1 Basic Framework

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

7.1.1 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

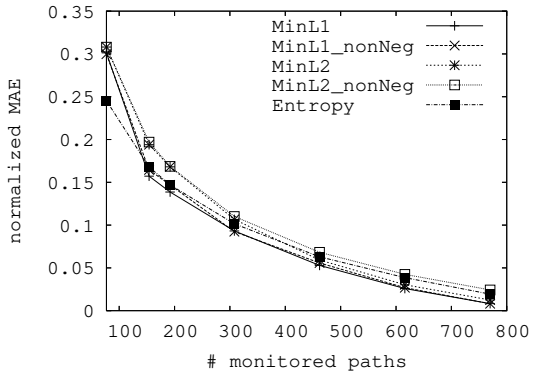


Figure 5: Comparison of inference algorithms for delay estimation in PlanetLab-RTT using A-optimal design.

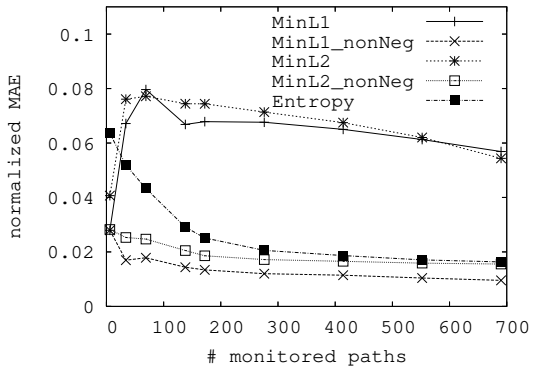


Figure 6: Comparison of inference algorithms for loss estimation in PlanetLab-loss using A-optimal design.

paths to monitor. Figure 5 and Figure 6 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.

As shown in Figure 5, 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 6, 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 4.2, 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 paper, unless noted otherwise, we use MinL1_nonNeg for both delay and loss inference.

7.1.2 Comparison of Measurement Designs

Next we evaluate different algorithms for designing measurement experiments. We consider inferring two types of quantities: network-wide performance and individual path performance.

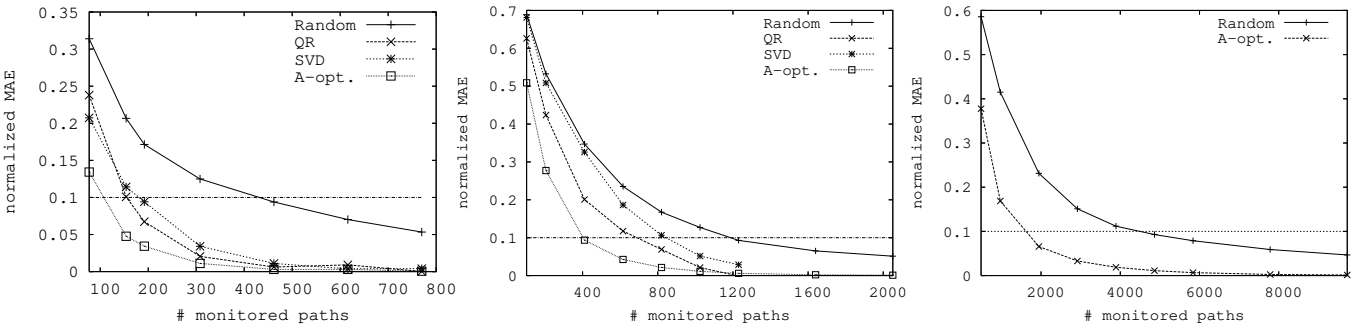
Network-wide performance: A global view of the performance aggregated over an entire network is useful for a variety of reasons. It can be used for estimating a typical user experience (as in the Internet End-to-end Performance Monitoring Project (IEPM)), detecting anomalies, trouble-shooting, and optimizing performance. The pioneering work in this area is done by Chua et al. [4, 5], 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 $f(\mathbf{x}) = \frac{1}{m}\mathbf{1}A\mathbf{x}$, where $\mathbf{1}$ 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 7 compares different experimental design schemes when MinL2 is used for inference, and Figure 8 shows the results when MinL1_nonNeg is used. As noted in Section 3.1, A-optimal and D-optimal designs are identical for inferring network-wide average, so we only show the results of A-optimal. A-optimal is more scalable than SVD and QR, both of which cannot handle Brite-n5000-o600, and fail to run on Brite-n1000-o200 when the number of monitored paths exceeds 1200. Therefore the SVD and QR curves in Figure 7(b) and Figure 8(b) stop at 1200 monitored paths, and Figure 7(c) and Figure 8(c) only show the results for Bayesian experimental designs and random path selection for Brite-n5000-o600.

As shown Figure 7(a)-(c), 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, 409 out of 39800 paths in Brite-n1000-o200, 1945 out of 359400 paths in Brite-n5000-o600). 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 8 when the inference algorithm changes to MinL1_nonNeg.

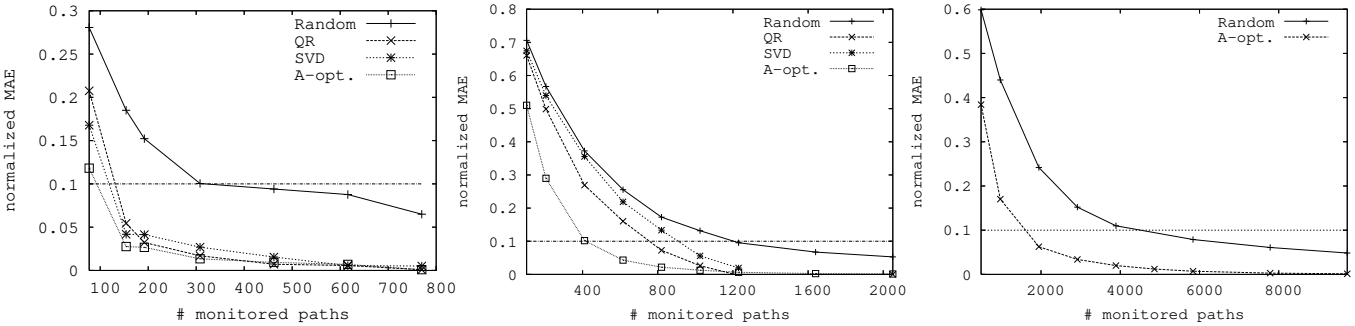
Individual path performance: Figure 9 compares different measurement design schemes for inferring individual path delay. Note that the results for SVD and QR are not available for Brite-n1000-o200 over 1200 monitored paths, nor for Brite-n5000-o600, since they do not scale well. As we can see, the rank-based approach requires monitoring 769 - 9729 paths, which is expensive. In comparison, the other approaches can provide a smooth tradeoff between inference accuracy and measurement overhead. Among them, A-optimal performs the best. To achieve 10% inference error, the other algorithms need to monitor up to 60% more paths than A-optimal. Note that D-optimal performs significantly worse than A-optimal, and sometimes even worse than the other alternatives. This is likely due to the fact that the Kullback-Leibler distance tends to under-penalize estimation errors. Finally, we observe that the performance benefit of A-optimal is largest when the number of monitored path is close to one to two thirds of the rank of the routing matrix. This is because when the number of monitored paths is too small, there is not enough information for accurately reconstructing the global view of network performance, regardless of which design scheme is used. In the other extreme, when the number of monitored paths is close to the rank, any independent row combinations (in the routing matrix) can provide sufficient information for accurate reconstruction of end-to-end path properties.

Figure 10 shows the absolute inference error in loss rate estimation. A-optimal slightly out-performs the other schemes. The performance gap is smaller than that of delay, because most links have close to zero loss rate, and assigning links with zero loss rate (even without extensive network monitoring) can achieve low aver-



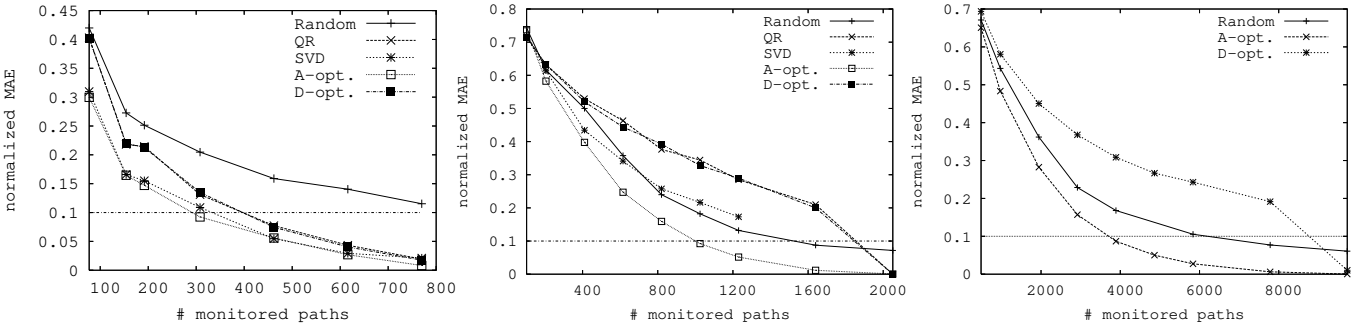
(a) PlanetLab-RTT (b) Brite-n1000-o200 (c) Brite-n5000-o600

Figure 7: Comparison of experimental design schemes for estimating network-wide delay using MinL2 inference algorithm.



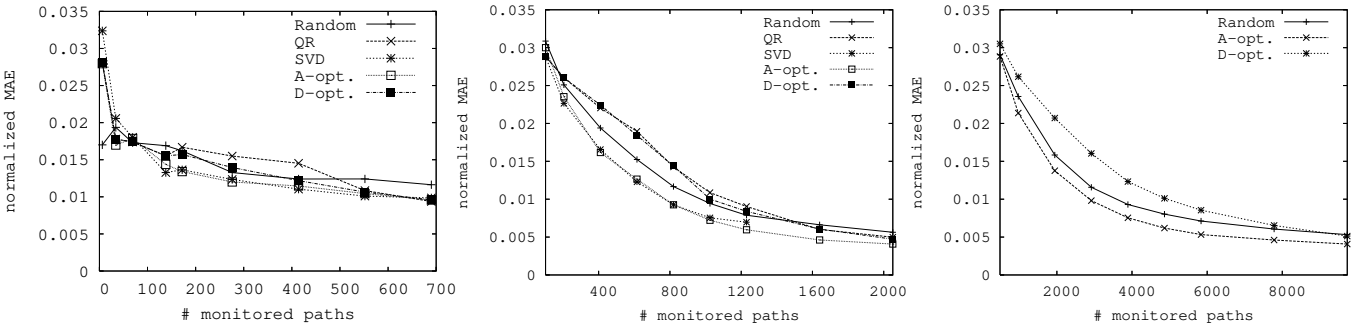
(a) PlanetLab-RTT (b) Brite-n1000-o200 (c) Brite-n5000-o600

Figure 8: Comparison of experimental design schemes for estimating network-wide delay using MinL1_nonNeg inference algorithm.



(a) PlanetLab-RTT (b) Brite-n1000-o200 (c) Brite-n5000-o600

Figure 9: Comparison of experimental design schemes for per-path delay inference using MinL1_nonNeg inference algorithm.



(a) PlanetLab-loss (b) Brite-n1000-o200 (c) Brite-n5000-o600

Figure 10: Comparison of experimental design schemes for per-path loss inference using MinL1_nonNeg inference algorithm (simulation uses the Gilbert loss model).

age inference error. Note that when the number of monitored paths is equal to the rank of the routing matrix, the error is non-zero due to the sampling errors we introduce when assigning true loss rates in synthetic topologies. Finally, we observe that D-optimal is not competitive, and performs considerably worse than A-optimal. We observe similar results for Bernoulli loss model and other topologies. So hereafter we will focus on Bayesian A-optimal designs.

7.1.3 Summary

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.

7.2 Flexibility of Measurement Design

In this section, we evaluate the flexibility of our measurement framework by estimating delay on individual network paths in the PlanetLab-RTT topology.

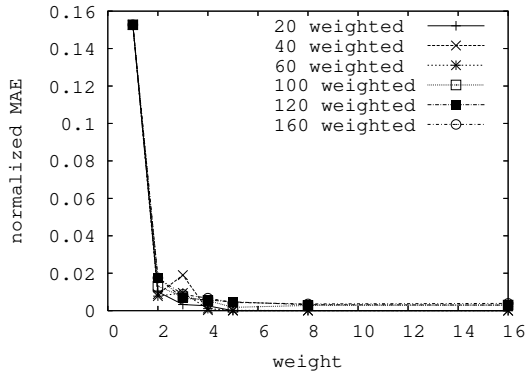
7.2.1 Differentiated Design

First we examine the effectiveness of experimental designs for providing differentiated treatment to a subset of paths. We apply the technique described in Section 3.2 to achieve this goal. In our evaluation, we randomly assign a subset of preferred paths with a weight varying from 1 to 16, while fixing the weight on the remaining paths to 1. Figure 11 shows the inference error on both the preferred and the remaining paths when we monitor 200 paths in PlanetLab-RTT topology and vary the number of preferred paths from 20 to 160. We make the following observations. First, as we would expect, the inference error on the preferred paths decreases with an increasing weight. When the weight is 4 and higher, the inference error is close to 0. This is because when the weight is high enough, the performance of many preferred paths is either directly monitored or exactly reconstructed from the monitored paths. Second, we observe that the inference error on the remaining paths increases slightly, since as we pay more attention to the preferred paths, the remaining paths are monitored less extensively. For a similar reason, the inference error of the remaining paths tends to increase slightly with an increasing number of preferred paths.

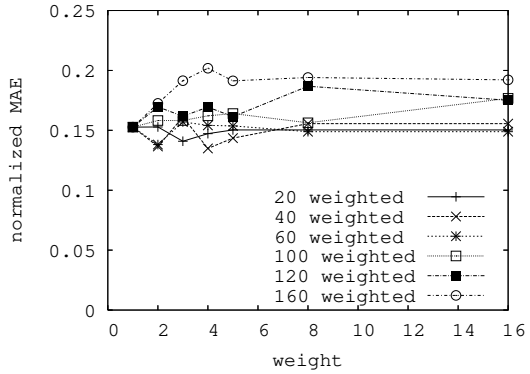
7.2.2 Augmented Design

Next we consider augmented design for supporting continuous monitoring. Our evaluation is based on the following scenario. Suppose we identify a set of paths to monitor, and some of them fail to provide us measurement data (e.g., due to software or hardware failures at monitor sites or at their incoming/outgoing links). In this case, we need to identify the additional measurements to conduct given that we have already obtained the measurement results from the unfailed paths.

In our evaluation, we first use A-optimal to identify 100 paths to monitor in PlanetLab-RTT. Then we vary the number of failed paths from 10 to 80, and apply different augmented design algorithms to determine the additional paths to monitor. Figure 12 shows the average inference error under different schemes. As we can see, A-optimal yields the lowest error. Moreover, its inference error is similar for a varying number of failed paths. In comparison,



(a) Inference error on the preferred paths



(b) Inference error on the remaining paths

Figure 11: Use differentiated design based on A-optimal design to provide a higher resolution in estimating a selected set of preferred paths in PlanetLab-RTT.

the inference error of the other schemes tends to increase with an increasing number of failed paths. This suggests that the A-optimal design is the most effective in augmenting existing designs.

7.2.3 Multi-user Design

Now we study the multi-user scenarios, where each user is interested in a certain part of network. We compare the following design schemes:

- *Separate design and separate inference (sep./sep.)*: Each user individually determines the set of paths to monitor, and makes inference based solely on his/her own observations.
- *Separate design and joint inference (sep./joint)*: This is an enhancement of the previous version. Users still individually decide which paths to monitor, but they make inference based on the observations made from all users.
- *Augmented design and joint inference (aug./joint)*: In the augmented design, we first design measurement experiments for user 1, and then apply the the augmented design (in Section 7.2.2) to construct measurement experiments for user 2. We continue the process for all the other users.
- *Union design and joint inference (union/joint)*: In the union design, we take a union of all the paths that are interesting to at least one user, and then apply the (basic) measurement design.
- *Joint design and joint inference (joint/joint)*: Unlike in the union design, where all interesting paths are treated equally, in joint design we set a path's weight to be the square root of the number of users who are interested in the path (see Section 3.2).

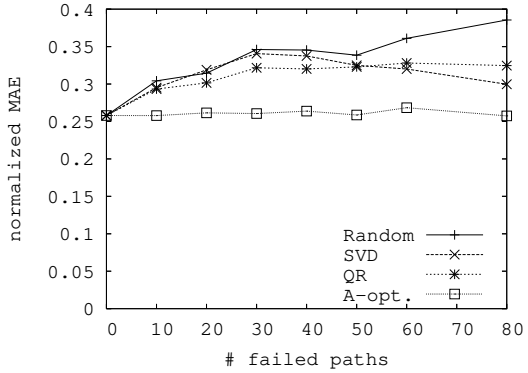


Figure 12: Comparison of various augmented design schemes in PlanetLab-RTT.

All the design algorithms can work in separate, augmented, and union modes. In addition, A-optimal can also support joint design.

In our evaluation, we have 16 users, each interested in 50 paths. There are a common set of paths that are interesting to all users. Figure 13(a) compares the A-optimal design in its various design modes for inferring individual path delay as the number of common paths is varied from 0 to 40. The remaining paths interesting to a user are randomly selected from all the non-common paths. In the order of accuracy ranking, joint/joint $>$ union/joint $\approx$ aug./joint $>$ sep./joint $>$ sep./sep. In particular, sep./sep. incurs significantly higher error than the others, because in this case different users do not share their observation. Enabling information sharing in sep./joint reduces the normalized *MAE* by 0.2 or higher. A further reduction is achieved by incorporating users' interest into measurement design. Interestingly, augmented design performs similarly to union design, even though the former is an online version of the latter (i.e., the i-th user determines its measurement experiments without considering the j-th user's interest for $j > i$). The performance of joint/joint is even better, and its benefit over separate design grows as the number of common paths increases. This is attributed to the fact that it not only incorporates all users' interest in designing measurement, but also it biases measurement towards paths that interest more users. Figure 13(b) compares the inference error as we vary the number of monitored paths. As before, joint/joint yields the best performance among all the schemes. The performance gap is largest with a small number of monitored paths. This suggests that experimental design is especially important under tight measurement resource constraints.

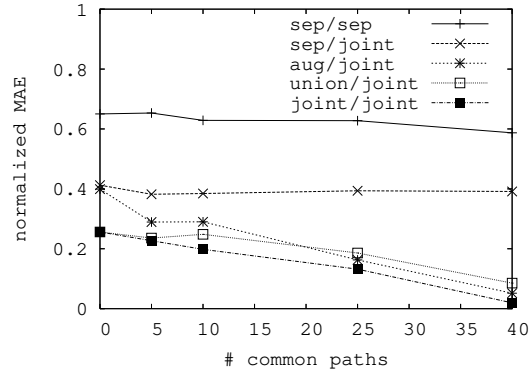
Next we compare the performance across different design schemes. Figure 14 (a) and (b) show the inference error of various design schemes under their best design modes. They all use joint inference. As they show, A-optimal yields the lowest error, up to 80% lower than the alternatives.

7.2.4 Summary

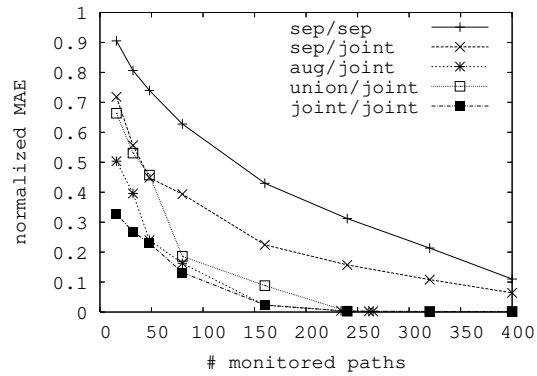
To summarize, we demonstrate the flexibility of our measurement framework using the real trace. Our results show that it can effectively support differentiated design, augmented design, and joint design. Such capabilities are useful for a variety of network monitoring applications.

7.3 Effects of Prior Information

So far we assume no prior information about $\mathbf{x}$. In our future work, we plan to develop light-weight techniques to obtain better priors and thus further improving the inference accuracy. As a first step, below we develop a simple method for obtaining an enhanced



(a) vary # common paths, while fixing the monitored paths to be 80



(b) vary # monitored paths, while fixing the common paths to be 25

Figure 13: Comparison of different design modes in handling multi-user scenarios using PlanetLab-RTT, where all modes use the A-optimal design.

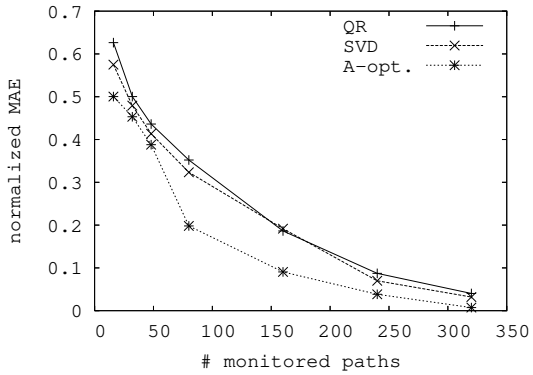
prior. Specifically, we consider a special form of prior whose elements are all equal: $\mu = z \cdot \mathbf{1}$, where z is an unknown, and $\mathbf{1}$ is an all-1 column vector of length n. We can estimate z by solving an over-determined least-squares problem: $\mathbf{y} = A\mu = (A\mathbf{1})z$, yielding $z = \frac{(A\mathbf{1})^T\mathbf{y}}{\|A\mathbf{1}\|_2^2}$. Intuitively, the resulting prior $\mu = \frac{(A\mathbf{1})^T\mathbf{y}}{\|A\mathbf{1}\|_2^2} \cdot \mathbf{1}$ estimates the average link performance. An advantage of this method is that it is extremely simple and requires no extra measurement.

Figure 15 shows the accuracy of delay inference for different measurement design schemes using the above enhanced prior. Compared with Figure 9(a), we make the following observations. First, the enhanced prior improves the inference accuracy for all measurement design schemes. It reduces the normalized *MAE* by up to 0.07 (or about 25%). The improvement is largest when the number of monitored paths is small. This is because the enhanced prior information is most helpful to compensate for incomplete monitoring information. In comparison, with extensive monitoring we can accurately estimate performance even without prior. Second, the relative ranking of different design schemes remains the same as before. A-optimal design continues to yield the highest accuracy.

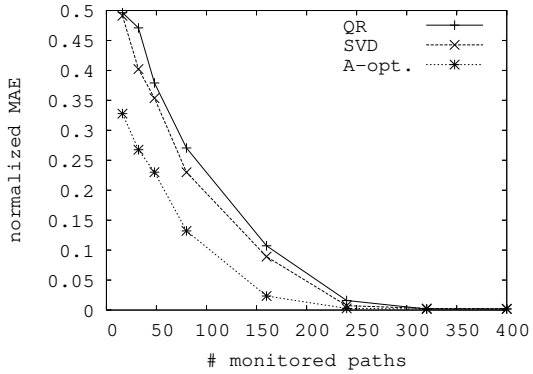
On the other hand, the enhanced prior only yields very little accuracy improvement for loss inference (results omitted in the interest of brevity). This is not surprising, because most links have very low loss rate, making $\mu = \mathbf{0}$ a good prior.

8. CONCLUSION

In this paper, we develop NetQuest, a flexible framework for large-scale network measurement and inference. It consists of two major components: the design of measurement experiments, and



(a) 10 common paths



(b) 25 common paths

Figure 14: Comparison of different design schemes using their best modes on PlanetLab-RTT.

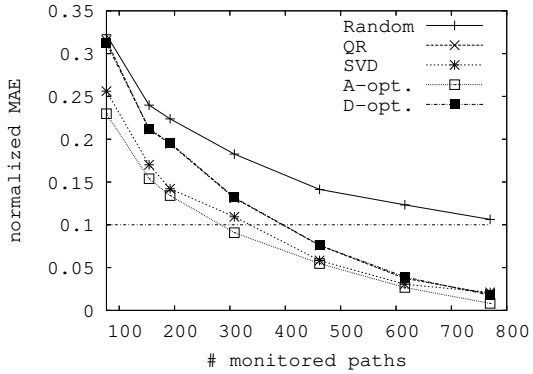


Figure 15: Effects of the enhanced prior on PlanetLab-RTT.

network inference. For the former, we leverage powerful tools developed in the field of Bayesian experimental design. For the latter, we build on top of a number of existing network tomography techniques to infer network properties based on partial, indirect observations. Our framework is flexible, and can accommodate a variety of design objectives, such as differentiated, augmented, and multi-user designs. It is also highly scalable and can design measurement experiments that span thousands of routers and end hosts.

There are several avenues for future work. First, we plan to further enhance the robustness of our experimental design and inference by making the design *fault tolerant*. Specifically, we want a design that minimizes the *worst-case* design criterion in the presence of multiple faulty paths (*i.e.*, paths that experience either failures or major routing changes). Second, we are interested in applying our techniques to estimating other network properties, such as traffic matrix estimation. In particular, we may use Bayesian experimental design to identify strategic locations to place additional measurement capabilities to enhance the accuracy of traffic matrix estimation. Third, we would like to extend our framework to incorporate additional design constraints and to handle non-linear metrics. Note that there is a rich literature on non-linear Bayesian experimental design [1]. Finally, we are interested in developing light-weight techniques to obtain better prior information.

9. REFERENCES

[1] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10:273–304, 1995. `http://citeseer.ist.psu.edu/chaloner95bayesian.html`.

[2] Y. Chen, D. Bindel, and R. H. Katz. Tomography based overlay network monitoring. In *Proc. of ACM/USENIX Internet Measurement Conference*, 2003.

[3] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *Proc. of ACM SIGCOMM*, 2004.

[4] D. B. Chua, E. D. Kolaczyk, and M. Crovella. Efficient monitoring of end-to-end network properties. In *Proc. of IEEE INFOCOM*, Jul. 2004.

[5] D. B. Chua, E. D. Kolaczyk, and M. Crovella. A statistical framework for efficient monitoring of end-to-end network properties. In *Proc. of ACM SIGMETRICS*, 2005.

[6] M. A. Clyde. Experimental design: A Bayesian perspective. *International Encyclopia Social and Behavioral Sciences*, Apr. 2001.

[7] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. New York: Wiley, 1991.

[8] T. Davis. LDL: A sparse LDL' factorization and solve package (version 1.2), 2005. `http://www.cise.ufl.edu/research/sparse/ldl/`.

[9] D. Donoho. For most large underdetermined systems of linear equations, the minimal l1-norm near-solution approximates the sparsest near-solution. `http://www-stat.stanford.edu/~donoho/Reports/2004/l1l0approx.pdf`.

[10] V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.

[11] 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`.

[12] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, 2nd edition, 1989.

[13] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, 1997.

[14] B. Jones, D. Lin, and C. Nachtsheim. Bayesian D-optimal supersaturated designs, 2004. Submitted for publication.

[15] D. V. Lindley. *Bayesian Statistics – A Review*. SIAM, Philadelphia, PA, 1972.

[16] A. Medina, A. Lakhina, I. Matta, and J. Byers. Boston university representative Internet topology generator (BRITE). `http://www.cs.bu.edu/brite/`.

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

[18] A. J. Miller and N. K. Nguyen. A Fedorov exchange algorithm for D-optimal design. *Applied Statistics*, 1994.

[19] T. J. Mitchell. An algorithm for the construction of D-optimal designs. *Technometrics*, 1974.

[20] E. M. Nahum, C. C. Rosu, S. Seshan, and J. Almeida. The effects of wide-area conditions on WWW server performance. In *Proc. of SIGMETRICS*, Jun. 2001.

[21] N. K. Nguyen and A. J. Miller. A review of exchange algorithms for constructing discrete D-optimal designs. *Computational Statistics and Data Analysis*, 1992.

[22] V. N. Padmanabhan, L. Qiu, and H. Wang. Server-based inference of Internet performance. In *Proc. of IEEE INFOCOM*, Mar. 2003.

[23] Planetlab. `http://www.planet-lab.org`.

[24] M. Saunders. PDCO: Primal-Dual interior method for Convex Objectives, 2003. `http://www.stanford.edu/group/SOL/software/pdco.html`.

[25] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A facility for distributed internet measurement. In *Proc. of USITS*, Mar. 2003.

[26] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *Proc. Internet Measurement Conference*, Oct. 2005.

[27] 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.

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