Efficient Identification of Additive Link Metrics via Network Tomography

Liang Ma[†], Ting He[‡], Kin K. Leung[†], Don Towsley[*], and Ananthram Swami[§]
[†]Imperial College, London, UK. Email: {l.ma10, kin.leung}@imperial.ac.uk
[‡]IBM T. J. Watson Research Center, Yorktown, NY, USA. Email: the@us.ibm.com
[*]University of Massachusetts, Amherst, MA, USA. Email: towsley@cs.umass.edu
[§]Army Research Laboratory, Adelphi, MD, USA. Email: ananthram.swami.civ@mail.mil

***Abstract*—We investigate the problem of identifying individual link metrics in a communication network from accumulated end-to-end metrics over selected measurement paths, under the assumption that link metrics are additive and constant during the measurement, and measurement paths cannot contain cycles. According to linear algebra, all link metrics can be uniquely identified when the number of linearly independent measurement paths equals n, the number of links. It is, however, inefficient to collect measurements from all possible paths, whose number can grow exponentially in n, as the number of useful measurements (from linearly independent paths) is at most n. The aim of this paper is to develop efficient algorithms for constructing linearly independent measurement paths and calculating link metrics. We propose one algorithm which can construct n linearly independent, cycle-free paths between monitors without examining all candidate paths, whose complexity is quadratic in n. A further benefit of the proposed algorithm is that the generated paths satisfy a nested structure that allows linear-time computation of link metrics without explicitly inverting the measurement matrix. Our evaluations on both synthetic and real networks verify the superior efficiency of the proposed algorithms, which are orders of magnitude faster than benchmark solutions for large networks.**

I. Introduction

Accurate and efficient monitoring of internal network state (e.g., delays and loss rates on internal links) is essential for various network operations such as route selection, resource allocation, and fault diagnosis. Directly measuring the performance of individual network elements (e.g., nodes/links) is, however, not always feasible due to the overhead caused by measurement traffic and the lack of support at internal network elements for making such measurements [1]. These limitations motivate the need for *external* approaches, where we infer the states of internal network elements by measuring the performance along selected paths from a subset of nodes with monitoring capabilities, hereafter referred to as *monitors*.

Depending on the measurement technique, external approaches can be classified as *hop-by-hop* approaches or *end-to-end* approaches. The former rely on special diagnostic tools such as *traceroute*, *pathchar*, *clink*, and *pipechar* [2] to reveal fine-grained performance metrics of individual links by sending active probes. These tools estimate link-level metrics such as delays, losses, and bandwidths by exchanging Internet Control Message Protocol (ICMP) packets with each intermediate node. Their estimates, however, have known accuracy issues due to asymmetry in routes and different priorities of ICMP and data packets. Some intermediate nodes may even be configured to ignore ICMP packets altogether. Moreover, the measurement process generates a heavy traffic load that competes for valuable network resources with data traffic.

Alternatively, end-to-end approach provides a light-weight solution that only requires the measurement of end-to-end performance metrics (e.g., end-to-end delays) between monitors, and uses network tomography techniques to solve for the corresponding metrics at individual links. Generally, *network tomography*, originated by Vardi [3], refers to the methodology of inferring fine-grained network characteristics from aggregate measurements; in our context, the fine-grained characteristics are individual link metrics, and the aggregate measurements are end-to-end metrics on selected measurement paths[1]. Since only end-to-end measurements are required, network tomography can utilize passive measurements from the transmissions of data packets [4], thus reducing traffic overhead as well as avoiding the need for internal cooperation or equal treatment of control/data packets.

A case of particular interest in network tomography is the inference of *additive* link metrics, i.e., the end-to-end metric over a path of multiple links is the sum of individual link metrics. A typical example of an additive metric is delay, while a multiplicative metric (e.g., packet delivery ratio) can be expressed in an additive form using the $\log(\cdot)$ function. For additive metrics, we can formulate the problem as that of solving a system of linear equations, where the unknown variables are the link metrics, and the known constants are the end-to-end path measurements, each equal to the sum of the corresponding link metrics. The goal of network tomography is essentially to solve this linear system of equations.

Most existing work on network tomography emphasizes inferring as much as possible about link metrics from available path measurements. However, past experience shows that it is frequently impossible to uniquely identify all link metrics from path measurements [5], [6], [7]. In the language of linear algebra, this is because the linear system associated with the measurement paths is noninvertible, i.e., the number of *linearly independent* paths is smaller than the number of links. To make the system invertible, we need to first ensure that there exists a sufficient number of linearly independent paths. To this end, we established in [8] necessary and sufficient conditions on the structure of the network (including the topology and the

The extended version of this paper has been published in *ICDCS'13*.

Research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

[1]The original work [3] addresses a different problem of inferring source-destination traffic matrix from aggregate traffic at relay nodes. Both traffic matrix estimation and link metric identification are representative applications of network tomography [4].

placement of monitors) for it to be *identifiable*, i.e., the number of linearly independent measurement paths equals the number of links, where measurement paths are restricted to cycle-free paths to conform with the requirement of routing protocols.

However, even if a network is known to be identifiable, it is still a challenge to determine a minimum set of paths to measure. Mathematically, this is because many paths are *linearly dependent* in that they can be represented as linear combinations of a subset of paths, and hence their measurements do not provide further information about link metrics. In general, the total number of candidate measurement paths in a network with n links can be exponential in n, but only (up to) n of them are linearly independent and thus useful for link identification. A smart strategy for finding n linearly independent paths will not only save resources in collecting measurements, but also increase the efficiency of calculating link metrics by discarding redundant linear equations.

In this paper, we investigate the following closely-related problems in network tomography: (a) Given an arbitrary identifiable network $\mathcal{G}$ with n links (verified by the condition derived in [8]), how can we efficiently construct n linearly independent, cycle-free paths between monitors? (b) Given measurements on the constructed paths, how can we efficiently calculate individual link metrics? The emphases in both problems are on *efficiency*; although one can enumerate all possible paths until finding n linearly independent ones, such a method will incur a large cost (exponential in n) and is thus unsuitable for large networks. In both problems, we assume that the link metrics are additive and constant. Note that a "constant" link metric refers to one that either changes slowly relative to the measurement process, or that is a statistical characteristic (e.g., mean, variance) of the link that stays constant over time[2].

A. Related Work

Based on the model of link metrics, existing work on network tomography can be broadly classified as algebraic and statistical approaches. Algebraic approaches, as in this paper, model link metrics as unknown constants, and use techniques from linear algebra to compute link metrics from path metrics [5], [6]. Statistical approaches model link metrics as random variables with (partially) unknown probability distributions, and apply various parametric/nonparametric techniques to estimate the distributions from realizations of path metrics [9], [1], [10].

When all nodes are allowed to participate in the measurement process, multicast can be exploited as a measurement method with broad coverage and low overhead [11], [12]. Subtrees and unicast are employed in [4], [13] as alternatives with more flexibility in selecting receivers, but measurements are still conducted along multicast trees.

When only selected nodes (i.e., monitors) can participate in the measurement process, as assumed in this paper, the problem becomes more challenging. When all but k link metrics are *zero*, [14] propose schemes inspired by compressive sensing to construct measurement paths to identify the k non-zero link metrics. For arbitrary valued link metrics, few positive results are known. If the network is directed (links have different metrics in different directions), [10] proves that not all link metrics are identifiable unless every non-isolated node is a monitor, and [5] shows that even if every node is a monitor, unique link identification is still impossible if measurements are restricted to cycles. If the network is undirected (links have equal metrics in both directions), [15] proposes an efficient algorithm to identify link metrics, given that monitors can measure cycles or paths containing cycles. A similar study in [16] characterizes the minimum number of measurements needed to identify a broader set of link metrics (including both additive and nonadditive metrics), under the stronger assumption that measurement paths can contain repeated links. Since routing along cycles is typically prohibited by routing protocols, it remains open what the solution becomes if only cycle-free paths can be measured. Given measurements on n linearly independent paths (n is the number of links), a general solution for computing link metrics using classic linear system solvers (e.g. Gaussian elimination with pivoting) takes $O(n^3)$ time [17], which can be slow for large n. In this paper, we overcome this drawback by developing efficient algorithms for path construction and link identification in arbitrary identifiable networks, employing only cycle-free paths.

B. Summary of Contributions

Our contributions are three-fold:

1) We develop an efficient algorithm to construct n linearly independent, cycle-free paths between monitors in an arbitrary identifiable network in $O(mn)$ time (m is the number of nodes and n the number of links), significantly improving on the exponential complexity of brute-force methods. The algorithm exploits the existence of three independent spanning trees, a special property of identifiable networks derived from the identifiability condition in [8] (see Theorem III.1).

2) Given measurements along the constructed paths, we develop an algorithm to calculate the n link metrics in $O(m+n)$ time, improving on the $O(n^3)$ complexity of the existing solution [17]. The algorithm exploits a nested structure of the constructed paths to avoid the need to explicitly invert the measurement matrix.

3) We compare the proposed algorithms against benchmark solutions through simulations on both synthetic networks and real ISP networks. The results verify the superior efficiency of the proposed algorithms, which can be more than 100 times faster than the benchmarks for large networks.

We note that not all link characteristics can be modeled as additive metrics (e.g., bit error rates). Our goal in this paper is to uniquely and efficiently identify additive link metrics; studies for non-additive metrics are left to future work.

The rest of the paper is organized as follows. Section II formulates the problem. Section III summarizes the theoretical foundations for developing algorithms. Section IV presents our algorithms for path construction and link identification. Evaluations of the proposed algorithms are given in Section V. Finally, Section VI concludes the paper.

II. Problem Formulation

A. Models and Assumptions

We assume that the network topology is known and model it as an undirected graph $\mathcal{G} = (V, L)$, where V and L are the sets of nodes and links, respectively. Without loss of generality, we assume $\mathcal{G}$ is connected, as different connected components have to be monitored separately. Let $m := |V|$ be the number of nodes and $n := |L|$ be the number of links in $\mathcal{G}$. Each link $l_i \in L$ ($i = 1, \ldots, n$) is associated with an *unknown* metric w_{l_i} that represents the average performance (e.g., average delay) of this link, modeled as a constant. In this paper, we assume that link metrics are symmetric in both directions,

[2]In this case, end-to-end measurements are also statistical characteristics, e.g., path mean/variance. In the case of variance, we also need the independence between link qualities to make the metric additive.

TABLE I
MAIN NOTATIONS

Symbol	Meaning
$V(\mathcal{G})$, $L(\mathcal{G})$	set of nodes/links in graph $\mathcal{G}$
m, n	number of nodes/links in $\mathcal{G}$
$\mathcal{G}+l$	add a link: $\mathcal{G}+l = (V(\mathcal{G}), L(\mathcal{G}) \cup \{l\})$, where the end-points of link l are in $V(\mathcal{G})$
$\mathcal{G} \cup \mathcal{G}'$	graph union: $\mathcal{G} \cup \mathcal{G}' = (V(\mathcal{G}) \cup V(\mathcal{G}'), L(\mathcal{G}) \cup L(\mathcal{G}'))$
$\mathcal{P}$	simple path, defined as a graph with $V(\mathcal{P}) = \{v_0, \ldots, v_k\}$ and $L(\mathcal{P}) = \{v_0v_1, v_1v_2, \ldots, v_{k-1}v_k\}$, where $v_0, \ldots, v_k$ are distinct nodes
μ_i	$\mu_i \in V(\mathcal{G})$ is the i-th monitor in $\mathcal{G}$
w_l, $c_{\mathcal{P}}$	metric of link l, sum metric of path $\mathcal{P}$

i.e., if ij denotes the link from node i to node j, then links ij and ji have the same metric. Table I summarizes the main notation used in this paper (following the convention of [18]).

Certain nodes in V are monitors, which can initiate/collecte measurements. We assume source routing at the monitors, i.e., they can control the routing of measurement packets as long as the path begins and ends at distinct monitors and does not contain repeated nodes. In the language of graph theory, we limit measurements to *simple paths* between monitors. Let $\mathbf{w} = (w_{l_1}, \ldots, w_{l_n})^T$ denote the column vector of all link metrics, and $\mathbf{c} = (c_{\mathcal{P}_1}, \ldots, c_{\mathcal{P}_\gamma})^T$ the column vector of all available path measurements, where γ is the number of measurement paths and $c_{\mathcal{P}_i}$ is the sum of link metrics along measurement path $\mathcal{P}_i$. The measurements are linked to the unknown link metrics by the following linear system:

$$\mathbf{R}\mathbf{w} = \mathbf{c}, \tag{1}$$

where $\mathbf{R} = (R_{ij})$ is a $\gamma \times n$ *measurement matrix*, with each entry $R_{ij} \in \{0, 1\}$ indicating whether link j is on path i. The network tomography problem is to invert this linear system to solve for $\mathbf{w}$ given $\mathbf{R}$ and $\mathbf{c}$.

A link is *identifiable* if the associated link metric can be uniquely determined from path measurements; network $\mathcal{G}$ is identifiable if all links in $\mathcal{G}$ are identifiable. The linear system model (1) implies that to uniquely determine $\mathbf{w}$, $\mathbf{R}$ must have full column rank, i.e., rank$(\mathbf{R}) = n$. In other words, we must find n *linearly independent* simple paths between monitors to take measurements. This is highly nontrivial because: (i) there may not exist n linearly independent simple paths in an arbitrary network with arbitrarily placed monitors; (ii) even if there exists a set of n linearly independent simple paths (not necessarily unique), it remains a challenge to efficiently discover such a set, as the total number of paths can grow exponentially with n. The naive solution of taking measurements along n arbitrarily selected paths is unlikely to uniquely identify all link metrics; an alternative solution of enumerating all possible paths until finding n linearly independent ones is unlikely to scale to large networks due to its exponentially growing complexity in n. It is therefore desirable to have a method that achieves both full rank and efficiency.

B. Objective

Given an identifiable network $\mathcal{G}$ with κ ($\kappa \geq 3$) monitors[3], the objective of this paper is to develop efficient algorithms for constructing monitor-to-monitor measurement paths and calculating each link metric in $\mathcal{G}$.

C. Illustrative Example

Fig. 1 displays a sample network with three monitors (μ_1–μ_3) and nine links (link 1–9). To identify all link metrics, nine

[3]Identifiability can be guaranteed by deploying monitors according to the Minimum Monitor Placement algorithm in [8].

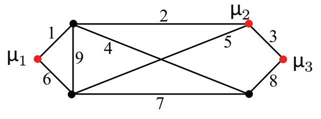


Fig. 1. Sample network with three monitors: μ_1, μ_2, and μ_3.

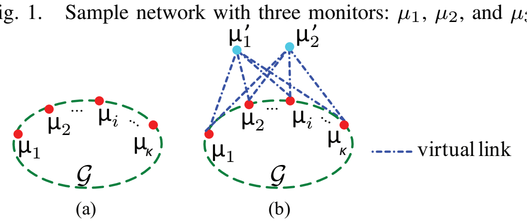


Fig. 2. (a) $\mathcal{G}$ with κ ($\kappa \geq 3$) monitors; (b) $\mathcal{G}_{ex}$ with two virtual monitors.

measurement paths are constructed to form the measurement matrix $\mathbf{R}$:

$$\begin{array}{l} \mu_1 \to \mu_2 : 6\ 5 \\ \qquad\qquad 6\ 9\ 2 \\ \qquad\qquad 1\ 4\ 7\ 5 \\ \mu_1 \to \mu_3 : 1\ 4\ 8 \\ \qquad\qquad 6\ 7\ 8 \\ \qquad\qquad 6\ 9\ 4\ 8 \\ \mu_2 \to \mu_3 : 3 \\ \qquad\qquad 5\ 7\ 8 \\ \qquad\qquad 2\ 4\ 8 \end{array} \Rightarrow \mathbf{R} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

where $R_{ij} = 1$ if and only if link j is on path i. Given the vector $\mathbf{c}$ of sum metrics measured on the constructed paths, we have a linear system $\mathbf{R}\mathbf{w} = \mathbf{c}$. Since $\mathbf{R}$ is *invertible* in this example, we can uniquely identify $\mathbf{w}$ by $\mathbf{w} = \mathbf{R}^{-1}\mathbf{c}$. If path 65 (first row in $\mathbf{R}$) is replaced by path 12, then the new measurement matrix is not invertible, and thus cannot uniquely identify all link metrics. A careful selection of measurement paths is therefore crucial.

III. FUNDAMENTALS ON NETWORK IDENTIFIABILITY

A. Conditions for Identifiability

A prerequisite of any successful link identification method is that the network contain a sufficient number of linearly independent measurement paths. In our previous work [8], we translated this requirement into explicit conditions on the network topology and the placement of monitors as stated below.

Theorem III.1 ([8])**.** Given κ ($\kappa \geq 2$) monitors deployed to measure simple paths between monitors, we have that:

1) if $\kappa = 2$, then no nontrivial $\mathcal{G}$ (with $n > 1$) is identifiable regardless of its topology and the placement of monitors;
2) if $\kappa \geq 3$, then $\mathcal{G}$ is identifiable if and only if the associated *extended graph* $\mathcal{G}_{ex}$ (see Fig. 2) is 3-vertex-connected.

The extended graph $\mathcal{G}_{ex}$ is constructed as follows: as illustrated in Fig. 2, given a network $\mathcal{G}$ with monitors $\mu_1, \cdots, \mu_\kappa$, $\mathcal{G}_{ex}$ is obtained by adding two *virtual monitors* μ'_1 and μ'_2, and 2κ *virtual links* between each pair of virtual-actual monitors. The theorem states that the sufficient and necessary condition for identifying all link metrics in $\mathcal{G}$ is that there are at least 3 monitors, and the extended graph remains connected after removing two arbitrary nodes (i.e., it is 3-vertex-connected).

B. Test and Assurance of Identifiability

Besides being of theoretical value, the above result also has direct application to algorithm design. A straightforward application leads to an algorithm that tests the identifiability of a given network $\mathcal{G}$ under a given monitor placement. This is achieved by applying a classic algorithm [19] to test 3-vertex-connectivity of $\mathcal{G}_{ex}$, with a complexity of only $O(m + n)$.

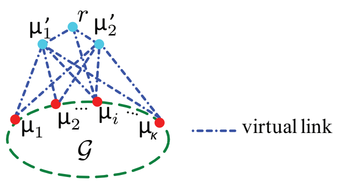


Fig. 3. The r-extended graph $\mathcal{G}^*_{ex}$ of $\mathcal{G}$.

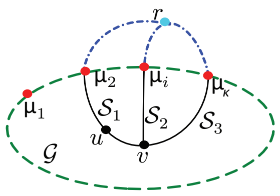


Fig. 4. Three internally vertex disjoint paths between v and r imply three monitor-to-monitor simple paths: $\mathcal{S}_1 \cup \mathcal{S}_2$, $\mathcal{S}_1 \cup \mathcal{S}_3$, and $\mathcal{S}_2 \cup \mathcal{S}_3$.

A more advanced application is in monitor placement, where the objective is to achieve network identifiability with the minimum number of monitors. The algorithm, called *Minimum Monitor Placement (MMP)*, decomposes $\mathcal{G}$ into subgraphs with certain properties (triconnected components) and sequentially places monitors in each subgraph to satisfy the condition in Theorem III.1.2; see [8] for details. MMP is provably optimal in that it minimizes the number of monitors required to guarantee network identifiability. It is also efficient, with a complexity of $O(m + n)$.

IV. Algorithm Design

After verifying that the network is identifiable, i.e., there exist n linearly independent simple paths between monitors, the natural followup questions are: how can we efficiently find n such paths, and how can we efficiently compute link metrics from measurements on these paths? In this section, we first provide the key idea behind our solutions and then formally present the algorithms and analyze their complexity.

The idea of efficient path construction and link identification originates from the identifiability condition in Theorem III.1.2. Since the condition is necessary, any identifiable network $\mathcal{G}$ must have a 3-vertex-connected extended graph $\mathcal{G}_{ex}$. We further extend $\mathcal{G}_{ex}$ by adding another virtual node r and connecting it to the virtual monitors μ'_1, μ'_2 and any one of the real monitors μ_i (for any $i \in \{1, \ldots, \kappa\}$) with three virtual links, as illustrated in Fig. 3; we refer to the new graph as the *r-extended graph*, denoted by $\mathcal{G}^*_{ex}$. It can be shown that $\mathcal{G}^*_{ex}$ is also 3-vertex-connected. By Menger's theorem [18], this implies that there exist at least 3 *internally vertex disjoint* simple paths between any two nodes in $\mathcal{G}^*_{ex}$ (i.e., the paths are disjoint except at end-points). In particular, as illustrated in Fig. 4, each non-monitor node v has (at least) 3 internally vertex disjoint simple paths to the virtual node r. Since any path to r must go through at least one (real) monitor, we can truncate each path at the first monitor on the way to r (i.e., removing the sub-path from this monitor to r), which provides three v-to-monitor paths $\mathcal{S}_1, \mathcal{S}_2$, and $\mathcal{S}_3$ that are disjoint except at v. This allows us to construct three monitor-to-monitor paths $\mathcal{P}_1 := \mathcal{S}_1 \cup \mathcal{S}_2$, $\mathcal{P}_2 := \mathcal{S}_1 \cup \mathcal{S}_3$, and $\mathcal{P}_3 := \mathcal{S}_2 \cup \mathcal{S}_3$, each being a simple path valid for taking measurements. Based on measurements $c_{\mathcal{P}_i}$ $(i = 1, 2, 3)$ from the constructed paths, we can obtain the individual metrics of $\mathcal{S}_i$ by solving the following linear equations:

$$\begin{cases} c_{\mathcal{S}_1} + c_{\mathcal{S}_2} = c_{\mathcal{P}_1}, \\ c_{\mathcal{S}_1} + c_{\mathcal{S}_3} = c_{\mathcal{P}_2}, \\ c_{\mathcal{S}_2} + c_{\mathcal{S}_3} = c_{\mathcal{P}_3}, \end{cases}$$

Algorithm 1: Spanning Tree-based Path Construction (STPC)

```
input : Network $\mathcal{G}$ with $\kappa$ monitors such that every link in $\mathcal{G}$
        is identifiable
output: Measurement paths in the form of (rows in) a
        measurement matrix $\mathbf{R}$
1  $\mathbf{R} = \emptyset$;
2  Construct $\mathcal{G}^*_{ex}$ from $\mathcal{G}$; //see Fig. 3
3  Find three spanning trees $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$ of $\mathcal{G}^*_{ex}$ that are
   pairwise independent wrt $r$ by the algorithm in [20];
4  foreach node $v$ in $\mathcal{G}$ do
5      if $v$ is a monitor then
6          $\mathcal{P}_{v1} \leftarrow \mathcal{S}_{v1}$; $\mathcal{P}_{v2} \leftarrow \mathcal{S}_{v2}$; $\mathcal{P}_{v3} \leftarrow \mathcal{S}_{v3}$;
7      else
8          $\mathcal{P}_{v1} \leftarrow \mathcal{S}_{v1} \cup \mathcal{S}_{v2}$; $\mathcal{P}_{v2} \leftarrow \mathcal{S}_{v2} \cup \mathcal{S}_{v3}$;
           $\mathcal{P}_{v3} \leftarrow \mathcal{S}_{v3} \cup \mathcal{S}_{v1}$;
9      end
10     Append all non-degenerate $\mathcal{P}_{vi}$ $(i = 1, 2, 3)$ to $\mathbf{R}$;
11 end
12 foreach link $l$ not in $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$ do
13     Find a simple monitor-to-monitor path $\mathcal{P}_l$ traversing $l$ in
       graph $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3 + l$ (see Algorithm 2);
14     Append $\mathcal{P}_l$ to $\mathbf{R}$;
15 end
```

where $c_{\mathcal{S}_i}$ $(i = 1, 2, 3)$ is the path metric on $\mathcal{S}_i$. Repeating this procedure for every node in $\mathcal{G}$ yields the metrics from each node to three monitors. Furthermore, we will show a way to construct these paths such that they are nested, i.e., if a path from v to monitor μ_2 goes through a neighbor u, as illustrated in Fig. 4, then u must use the same path to connect to μ_2. Therefore, we can calculate the metric of link uv by subtracting the $u \rightarrow \mu_2$ path metric from the $v \rightarrow \mu_2$ path metric. We now present the algorithms in detail.

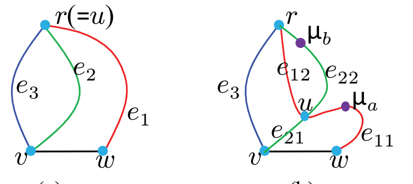
A. Spanning Tree-based Path Construction

Given an arbitrary network $\mathcal{G}$, we propose an algorithm, *Spanning Tree-based Path Construction (STPC)*, to construct linearly independent monitor-to-monitor paths, so that links can be uniquely identified from measurements on these paths. We assume that $\mathcal{G}$ is identifiable. This is guaranteed by deploying monitors according to algorithm MMP in [8], although STPC can also work with other monitor placements as long as network identifiability is guaranteed. In essence, STPC exploits a property of identifiable networks in terms of spanning trees. To this end, we introduce the following definition.

Definition 1. Two spanning trees of an undirected graph $\mathcal{G}(V, L)$ are *independent with respect to (wrt) a vertex $r \in V$* if the paths from v to r along these trees are internally vertex disjoint for every vertex $v \in V$ $(v \neq r)$.

For a 3-vertex-connected graph and any given vertex, Theorem 6 in [20] states that there exist three spanning trees that are pairwise independent wrt this vertex. In particular, since the r-extended graph $\mathcal{G}^*_{ex}$ is guaranteed to be 3-vertex-connected for any identifiable network, we will always be able to find three spanning trees of $\mathcal{G}^*_{ex}$ that are pairwise independent wrt r. These spanning trees provide three internally vertex disjoint paths from each non-monitor node v to r. As previously illustrated in Fig. 4, STPC constructs measurement paths by truncating these node-to-r paths at monitors and then concatenating each pair of truncated paths to form a simple monitor-to-monitor path in the original graph $\mathcal{G}$. See Algorithm 1 for details.

Specifically, STPC has two main steps: (1) constructing measurement paths based on independent spanning trees, and (2) constructing additional paths to measure links not in any



(a) $u = r$ (b) $u \neq r$

Fig. 5. Constructing measurement path traversing a non-tree link vw.

Algorithm 2: Path Construction for Non-Tree Links

```
input : Trees $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$ constructed in Algorithm 1 and a link
        $l = vw$ not in the trees
output: A simple monitor-to-monitor path $\mathcal{P}_l$ traversing $l$ and
        links in the trees
1 From the trees, find two paths $ve_2r$ and $ve_3r$ from $v$ to $r$ that
  do not traverse $w$, and a path $we_1r$ from $w$ to $r$ that does not
  traverse $v$;
2 On path $we_1r$ starting from $w$, find the first intersection node $u$
  with either $ve_2r$ or $ve_3r$;
3 if $u = r$ then
4   | $\mathcal{P}_l \leftarrow \mathcal{S}_{ve_2r} \cup l \cup \mathcal{S}_{we_1r}$;
5 else
6   | $\mathcal{P}_l \leftarrow \mathcal{S}_{ve_3r} \cup l \cup \mathcal{S}_{we_{11}ue_{22}r}$;
7 end
```

of the trees. The first step begins with the application of the algorithm in [20] to find three spanning trees $\mathcal{T}_i$ $(i = 1, 2, 3)$ of $\mathcal{G}^*_{ex}$ that are independent wrt r (line 3). Based on these spanning trees, STPC constructs paths to measure links in the trees (lines 4-11). Let $\mathcal{S}_{vi}$ $(i = 1, 2, 3)$ denote a simple path from node v to the first monitor μ $(\mu \neq v)$ toward r in $\mathcal{T}_i$. If no such μ exists, then $\mathcal{S}_{vi}$ represents a degenerate path containing just a single node v. STPC iterates among all nodes in $\mathcal{G}$: if v is a monitor, then $\mathcal{S}_{vi}$ $(i = 1, 2, 3)$ are already monitor-to-monitor simple paths (line 6); if v is not a monitor, then pairs of $\mathcal{S}_{vi}$'s again form monitor-to-monitor simple paths, as $\mathcal{S}_{v1}$, $\mathcal{S}_{v2}$, and $\mathcal{S}_{v3}$ are disjoint except at v (line 8). Therefore, all the constructed paths $\mathcal{P}_{vi}$ $(i = 1, 2, 3)$ that are non-degenerate (i.e., containing at least one link) are valid measurement paths, and are thus added to the measurement matrix (line 10).

The second step constructs paths for links not in any of the three trees (lines 12-15). For a non-tree link l, i.e., l not in any $\mathcal{T}_i$ $(i = 1, 2, 3)$, STPC invokes an auxiliary algorithm, Algorithm 2, to construct a measurement path $\mathcal{P}_l$ through l such that all the other links on this path belong to the trees. Algorithm 2 utilizes a simple observation as follows. As illustrated in Fig. 5, among the three internally vertex disjoint paths from v to r along the three spanning trees, there exist at least two paths, say ve_2r and ve_3r, that do not traverse w; similarly, there exists at least one path from w to r, say we_1r, that does not traverse v (line 1). Starting from w, we follow we_1r until the first intersection u with either ve_2r or ve_3r (line 2). If $u = r$ as in Fig. 5 (a), then we_1r and ve_2r (or ve_3r) are disjoint except at r. Truncating these paths at the first monitors toward r provides two disjoint paths, $\mathcal{S}_{we_1r}$ and $\mathcal{S}_{ve_2r}$, that connect w and v to monitors. Connecting these paths by link vw gives a simple path between monitors that traverses only vw and links in the trees (line 4). If $u \neq r$, we assume without loss of generality that u is an internal node on ve_2r as illustrated in Fig. 5 (b), which divides path we_1r into sub-paths $we_{11}u$ and $ue_{12}r$, and ve_2r into $ve_{21}u$ and $ue_{22}r$. The new path formed by $we_{11}ue_{22}r$ is disjoint with ve_3r except at r. Truncating paths $we_{11}ue_{22}r$ and ve_3r again provides two disjoint paths $\mathcal{S}_{we_{11}ue_{22}r}$ and $\mathcal{S}_{ve_3r}$ connecting w and v to monitors, which together with link vw form a simple monitor-to-monitor path traversing only vw and links in the trees (line 6). The validity of the algorithm is guaranteed by the following lemma.

Lemma IV.1. Path $\mathcal{P}_l$ constructed by Algorithm 2 is a simple monitor-to-monitor path traversing only l and links in the trees.

Proof: It is easy to see that $\mathcal{P}_l$ contains only one non-tree link, link l. To verify it as a simple path, it suffices to show that the following paths are disjoint except at r: ve_2r and we_1r if $u = r$, or ve_3r and $we_{11}ue_{22}r$ if $u \neq r$. The former is trivially satisfied. For the latter, note that sub-path $we_{11}u$ is disjoint with ve_3r as u is the first intersection (and ve_3r cannot contain u since it is internally vertex disjoint with ve_2r). Moreover, sub-path $ue_{22}r$ is disjoint with ve_3r except at r as ve_2r and ve_3r are internally vertex disjoint. Thus, paths $we_{11}ue_{22}r$ and ve_3r are disjoint except at r, completing the proof. ■

The correctness of STPC, i.e., the constructed measurement matrix $\mathbf{R}$ has rank n, will be clear in Section IV-B when we present an algorithm to explicitly compute all link metrics from measurements on these paths. In general, further processing is needed to find n linearly independent rows in $\mathbf{R}$ that specify the final set of measurement paths, since $\mathbf{R}$ may contain superfluous paths. However, we have a stronger result showing that the final processing is as simple as eliminating duplicate paths because all distinct paths (paths that differ in at least one node) found by STPC are linearly independent.

Theorem IV.2. The number of distinct paths constructed by STPC equals n, the number of links in $\mathcal{G}$.

The proof of Theorem IV.2 is in [21]. Since only distinct paths can be linearly independent, and the number of linearly independent paths constructed by STPC equals n (as these paths can uniquely identify the n links; see Section IV-B), Theorem IV.2 implies that all distinct paths found by STPC are linearly independent. Removing duplicate rows in the constructed $\mathbf{R}$ thus generates an $n \times n$ invertible measurement matrix.

B. Spanning Tree-based Link Identification

Given the paths constructed by STPC, we are now ready to take measurements and compute link metrics. A straightforward approach is to invert the measurement matrix to solve for the link metrics by $\mathbf{w} = \mathbf{R}^{-1}\mathbf{c}$ (assuming $\mathbf{R}$ is invertible)[4], with a complexity of $O(n^3)$, which we seek to avoid. We will show that the $\mathbf{R}$ generated by STPC has a special structure that allows us to directly compute link metrics at a much lower complexity. The algorithm, *Spanning Tree-based Link Identification (STLI)*, consists of three main steps as shown in Algorithm 3: (1) computing node-to-monitor path metrics (lines 1-3), (2) identifying links in the spanning trees (lines 4-12), and (3) identifying the other links (lines 13-15). In the sequel, we use $c_{\mathcal{P}}$ to denote both measured path metric if $\mathcal{P}$ is a monitor-to-monitor path, and calculated path metric if $\mathcal{P}$ is a node-to-monitor path (although the input vector $\mathbf{c}$ only contains measured path metrics).

The first step of STLI aims at computing the node-to-monitor metric $c_{\mathcal{S}_{vi}}$ for every node $v \in V$ and every $i \in \{1, 2, 3\}$ (lines 1-3). From the path construction in steps 6 and 8 of STPC, we see that $c_{\mathcal{S}_{vi}}$ is directly measured if v is a monitor. If v is not a monitor, we can construct three linear equations based on measurements on $\mathcal{P}_{vi}$ $(i = 1, 2, 3)$:

$$\begin{cases} c_{\mathcal{S}_{v1}} + c_{\mathcal{S}_{v2}} = c_{\mathcal{P}_{v1}}, \\ c_{\mathcal{S}_{v2}} + c_{\mathcal{S}_{v3}} = c_{\mathcal{P}_{v2}}, \\ c_{\mathcal{S}_{v3}} + c_{\mathcal{S}_{v1}} = c_{\mathcal{P}_{v3}}, \end{cases} \tag{2}$$

[4]Existing algorithms, e.g., Gaussian elimination [17], can directly solve for $\mathbf{w}$ without computing $\mathbf{R}^{-1}$; their complexity, however, remain $O(n^3)$.

Algorithm 3: Spanning Tree-based Link Identification (STLI)

```
input : Measurement paths and spanning trees $\mathcal{T}_i$ $(i = 1, 2, 3)$ constructed by Algorithm 1, measurements $\mathbf{c}$ on the paths
output: Vector $\mathbf{w}$ of link metrics in $\mathcal{G}$
foreach node $v$ in $\mathcal{G}$ do
    Compute $c_{\mathcal{S}_{vi}}$ from measurements $c_{\mathcal{P}_{vi}}$ $(i = 1, 2, 3)$ by (2);
end
foreach tree $\mathcal{T}_i$ $(i = 1, 2, 3)$ do
    foreach link $vw$ in tree $\mathcal{T}_i$ ($v$ is closer to $r$) do
        if $v$ is a monitor then
            $w_{vw} = c_{\mathcal{S}_{wi}}$;
        else
            $w_{vw} = c_{\mathcal{S}_{wi}} - c_{\mathcal{S}_{vi}}$;
        end
    end
end
foreach link $l$ not in $\mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3$ do
    Compute $w_l$ by subtracting metrics of the other links on $\mathcal{P}_l$ from $c_{\mathcal{P}_l}$;
end
```

from which we can compute $c_{\mathcal{S}_{vi}}$ $(i = 1, 2, 3)$.

The second step aims at solving for metrics of all links in the trees (lines 4-12). Consider a link vw in tree $\mathcal{T}_i$ $(i \in \{1, 2, 3\})$, where v is one hop closer to r. If v is a monitor, then the node-to-monitor path $\mathcal{S}_{wi}$ will only contain link vw, and thus its metric is also the metric of vw (line 7). If v is not a monitor, then the node-to-monitor path $\mathcal{S}_{vi}$ must be a sub-path of $\mathcal{S}_{wi}$, shorter by just link vw, and thus the difference in their metrics equals the metric of vw (line 9).

The final step addresses links not in the trees (lines 13-15). Since the measurement path $\mathcal{P}_l$ for each non-tree link l only contains l and links in the trees (Lemma IV.1), we just need to subtract from $c_{\mathcal{P}_l}$ the metrics of these tree links as identified in the second step to compute the metric of l (line 14).

Besides computing link metrics, STLI also serves as a constructive proof that the paths constructed by STPC can uniquely identify all links, i.e., the generated $\mathbf{R}$ has rank n.

C. Complexity Analysis

We now analyze the complexity of the proposed algorithms. STPC has an overall complexity of $O(mn)$. Specifically, the complexity of spanning tree construction in line 3 is $O(m|L(\mathcal{G}^*_{ex})|)$ [20], which is $O(mn)$ since $|L(\mathcal{G}^*_{ex})| = O(|L(\mathcal{G})|) = O(n)$. For lines 4-11, paths $\mathcal{P}_{vi}$ $(i = 1, 2, 3)$ can be constructed in $O(m)$ time for each node v; thus, lines 4-11 take $O(m^2)$ time. Finally, lines 12-15 invoke Algorithm 2 $O(n)$ times, and each invocation takes time $O(m)$. Combining the above yields an overall complexity of $O(nm)$. We point out that it is possible to save some computation by removing redundant links in $\mathcal{G}^*_{ex}$ using an $O(m+n)$-time algorithm in Section 4 of [20], which reduces the number of links to $O(m)$ while maintaining the 3-vertex-connectivity. This step reduces the complexity of spanning tree construction to $O(m^2)$, but the overall complexity remains the same.

Given measurements on the constructed paths, STLI can compute link metrics in $O(m+n)$ time. Specifically, computing $c_{\mathcal{S}_{vi}}$ (lines 1-3) takes $O(m)$ time. Then computing the metric of each link in the spanning trees (lines 6-10) takes only constant time, and there are $O(m)$ links in the trees, making the complexity of lines 4-12 $O(m)$. Finally, computing the metrics of non-tree links (lines 13-15) takes $O(n)$ time as explained below. Thus, the overall complexity is $O(m+n)$.

At first sight, it may seem that line 14 takes $O(m)$ time as there are $O(m)$ link metrics to subtract, making the overall complexity of STLI $O(nm)$. However, we observe that this step can be implemented in constant time using the knowledge of $c_{\mathcal{S}_{vi}}$ as follows. Consider the two cases in constructing $\mathcal{P}_l$ as illustrated in Fig. 5. Suppose that path we_1r belongs to tree $\mathcal{T}_{i_1}$, ve_2r to $\mathcal{T}_{i_2}$, and ve_3r to $\mathcal{T}_{i_3}$ $(i_1, i_2, i_3 \in \{1, 2, 3\}, i_2 \neq i_3)$. In case Fig. 5 (a), the measurement path $\mathcal{P}_l$ is a concatenation of link l, path $\mathcal{S}_{vi_2}$, and path $\mathcal{S}_{wi_1}$, and thus the metric of link l can be computed by $w_l = c_{\mathcal{P}_l} - c_{\mathcal{S}_{vi_2}} - c_{\mathcal{S}_{wi_1}}$. In case Fig. 5 (b), if the first monitor along path $we_{11}ue_{22}r$ appears before or at u (e.g., μ_a), then $\mathcal{P}_l$ consists of l, $\mathcal{S}_{vi_3}$, and $\mathcal{S}_{wi_1}$, and thus $w_l = c_{\mathcal{P}_l} - c_{\mathcal{S}_{vi_3}} - c_{\mathcal{S}_{wi_1}}$; if the first monitor appears after u (e.g., μ_b), then $\mathcal{P}_l$ consists of l, $\mathcal{S}_{vi_3}$, $we_{11}u$, and $\mathcal{S}_{ui_2}$, and thus $w_l = c_{\mathcal{P}_l} - c_{\mathcal{S}_{vi_3}} - (c_{\mathcal{S}_{wi_1}} - c_{\mathcal{S}_{ui_1}}) - c_{\mathcal{S}_{ui_2}}$ (since the metric of $we_{11}u$ equals $c_{\mathcal{S}_{wi_1}} - c_{\mathcal{S}_{ui_1}}$). In all the cases, w_l can be computed in constant time.

The above complexity results are well aligned with needs in practice. Path construction only needs to be performed once for a given topology, and thus can tolerate a higher complexity. In contrast, link identification needs to be performed more frequently to keep monitoring the health of links. In this regard, STLI in conjunction with STPC enables fast identification of link metrics that can scale to large networks.

V. Performance Evaluation

To evaluate the performance of STPC and STLI, we conduct a set of simulations on both randomly-generated and real network topologies. Given a network topology, we first apply the optimal monitor placement algorithm MMP in [8] to select a subset of nodes as monitors so that the network is guaranteed to be identifiable. We then apply the proposed algorithms to construct measurement paths between the placed monitors and compute link metrics from measurements on these paths. The focus of our evaluations is on the *efficiency*, measured by the average running time, of the proposed algorithms in comparison to benchmarks. For path construction, we also evaluate the cost of measuring the constructed paths, in terms of average path length (i.e., number of hops).

As a benchmark for STPC, we use the following algorithm[5], referred to as *Random Walk-based Path Construction (RWPC)*. Given an identifiable network $\mathcal{G}$, RWPC repeats the following steps until the rank of the constructed measurement matrix $\mathbf{R}$ equals n (starting from $\mathbf{R} = \emptyset$):

(i) starting from a randomly selected monitor, follow a random walker until it hits another monitor;
(ii) remove cycles from the path taken by the random walker to generate a simple monitor-to-monitor path;
(iii) if the generated path is linearly independent wrt existing paths in $\mathbf{R}$, append it to $\mathbf{R}$; otherwise, discard the path.

RWPC is essentially a randomized algorithm that examines one path at each iteration until n linearly independent paths are found. In practice, RWPC may iterate indefinitely for large networks. To control its running time, we impose a *maximum number of iterations* I_{MAX}, and force RWPC to terminate after I_{MAX} iterations. Consequently, we also measure its *success rate* r_{succ}, defined as the fraction of Monte Carlo runs during which RWPC successfully finds n linearly independent paths within I_{MAX} iterations (the success rate of STPC is always one). Limiting the number of iterations leads to underestimating the actual running time of RWPC in constructing n linearly

[5]Existing path-construction solutions cannot be used as benchmarks since they are not comparable to STPC for these two reasons: (i) most solutions assume given routing rather than controlled routing as assumed in this paper, (ii) even under controlled routing assumption, existing path construction algorithms may contain cycles, which is prohibited in this paper.

independent paths, but it allows us to apply the algorithm to large networks.

As a benchmark for STLI, we use the general solution [4] of inverting the measurement matrix: $\mathbf{w} = \mathbf{R}^{-1}\mathbf{c}$, referred to as *Matrix Inversion-based Link Identification (MILI)*. Here $\mathbf{R}$ is an invertible matrix computed by RWPC if it is successful, or STPC otherwise.

Our simulation results include the following metrics:

(a) κ, $\overline{\kappa}$: minimum number of monitors selected by MMP and its average (for randomly generated topologies);
(b) r_{succ}: success rate of RWPC;
(c) Υ: rank$(\mathbf{R})/n$ for RWPC when it is unsuccessful;
(d) t_{STPC}, t_{RWPC}: average running times of STPC and RWPC;
(e) t_{STLI}, t_{MILI}: average running times of STLI and MILI;
(f) h_{STPC}, h_{RWPC}: average lengths of the n paths constructed by STPC and RWPC (when successful).

The simulation is implemented in Matlab R2010a and performed on a computer with Intel Core i5-2540M CPU @ 2.60GHz, 4.00 GB memory, and 64-bit Win7 OS.

A. Random Topologies

We first evaluate the proposed algorithms on synthetic topologies generated according to three different random graph models: Erdös-Rényi (ER) graphs, Random geometric (RG) graphs, and Barabási-Albert (BA) graphs. For each model, we fix the number of nodes to 150, and randomly generate 100 graph realizations[6], which are then fed to the path construction algorithms. For each generated link, we randomly generate a link metric between 0 and 1, which is then used to compute path metrics. Here I_{MAX} is set to $3 \times n$. We now explain the models and the corresponding results separately.

1) Erdös-Rényi (ER) graph: The ER graph is a simple random graph generated by independently connecting each pair of nodes by a link with a fixed probability p. The result is a purely random topology where all graphs with an equal number of links are equally likely to be selected. It is known [22] that $p_0 = \log m/m$ is a sharp threshold for the graph to be connected with high probability, which implies a minimum value of p ($p = 0.0334$ for $m = 150$).

2) Random geometric (RG) graph: The RG graph is frequently used to model the topology of wireless ad hoc networks. It generates a random graph by first randomly distributing nodes in a unit square, and then connecting each pair of nodes by a link if their distance is no larger than a threshold d_c, which denotes node communication range. The resulting topology contains well-connected subgraphs in densely populated areas and poorly-connected subgraphs in sparsely populated areas. It is known that $d_c \geq \sqrt{\log m/(\pi m)}$ ensures a connected graph with high probability [23], which gives a minimum range of $d_c = 0.1031$ for $m = 150$.

3) Barabási-Albert (BA) graphs: The BA graph [24] is used to model many naturally occurring networks, e.g., Internet, citation networks, and social networks. To generate a BA graph, we begin with a small connected graph $\mathcal{G}_0 := (\{v_1, v_2, v_3, v_4\}, \{v_1v_2, v_1v_3, v_1v_4\})$ and add nodes sequentially. For each new node v, we connect v to ϱ existing nodes such that the probability of connecting to node w is proportional to the degree of w. If the number of existing nodes is smaller than ϱ, then v connects to all the existing nodes.

Simulation results are presented in Table II, where each row corresponds to a random graph model, with results averaged over 100 graph realizations. Since the number of links n and the number of monitors κ vary across realizations, we present the average values denoted by $\overline{n}$ and $\overline{\kappa}$. We have tuned parameters of each model to make the number of generated links roughly the same. In Table II, most graphs are 3-vertex-connected, thus requiring only 3 monitors to achieve identifiability (see Theorem III.1). From the results, we see that RWPC finds all linearly independent paths successfully for ER and BA graphs, but fails most of the time for RG graphs. This is because node degrees vary significantly in RG graphs, and once the random walker hits a low-degree node, it has only a few paths to reach monitors, resulting in a high probability of generating duplicate paths. In fact, RWPC can quickly find a majority ($> 90\%$) of the linearly independent paths for RG graphs, but its efficiency drops sharply as the path set grows since most of the newly generated paths are linearly dependent with existing ones. In contrast, STPC only generates paths that are guaranteed to be useful in identifying additional links, thus significantly improving the efficiency. The improvement allows STPC to achieve a significantly smaller running time than RWPC, especially for RG graphs where we see a 45-fold speedup. Note that this is only an underestimate as RWPC often fails to find all the linearly independent paths for RG graphs, and the actual speedup is even bigger. Our link identification algorithm STLI also shows superior efficiency, reducing the running time of MILI by an order of magnitude. A further observation is that t_{STPC}, t_{STLI}, and t_{MILI} are roughly the same for different types of graphs, as their complexity are only determined by the size of the network (measured by n), whereas the running time t_{RWPC} is sensitive to the specific topology. Meanwhile, we notice that STPC tends to generate paths that are longer than those generated by RWPC, especially for BA graphs. This is because STPC restricts paths to the spanning trees, selecting a longer path along spanning trees even if alternative shorter paths exist, while the random walker in RWPC is likely to take shorter paths to monitors. This is an intentional design in STPC to ensure linear independence of the constructed paths; the problem of minimizing path length while guaranteeing linear independence is left for future work. In addition to these results, we have also simulated random graphs with a different number of links and observed similar comparisons; see Section IV in [21] for details.

B. ISP Topologies

We also test these algorithms on real network topologies. We use the *Internet Service Provider (ISP)* topologies from the Rocketfuel project [25], which represent physical connections between backbone/gateway routers of several major ISPs around the globe. The available data do not include link performance metrics; thus, we simulate link metrics by randomly generated numbers between 0 and 1. We point out that the performance of the algorithms is independent of the values of link metrics. Since RWPC is a randomized algorithm, we repeat it for multiple Monte Carlo runs for each ISP topology and report average performance; the number of Monte Carlo runs is 100 unless otherwise stated. In this simulation, we set $I_{\text{MAX}} = 8 \times n$ since we observe that $I_{\text{MAX}} = 3 \times n$ results in zero success rate for RWPC.

Simulation results are presented in Table III, where we sort the networks according to their number of links (i.e., n). We notice that all networks need a significantly higher ratio of monitors compared with the synthetic networks (Table II), ranging from 30% (EBONE, AT&T, Sprintlink) to more than 60% (Abovenet). This is because ISP networks contain a large number of gateway routers to connect to customers or other ISPs, which appear as dangling nodes that have to be selected

[6]All these realizations are checked before use to ensure they are connected.

TABLE II
RANDOM GRAPHS (ER: $p = 0.0656$, RG: $d_c = 0.15554$, BA: $\varrho = 5$, $I_{\text{MAX}} = 3 \times n$)

graph	$\overline{n}$	m	$\overline{\kappa}$	r_{succ}	Υ	t_{STPC} (s)	t_{RWPC} (s)	t_{STLI} (ms)	t_{MILI} (ms)	h_{STPC}	h_{RWPC}
ER	736.46	150	3	100.00%	NA	20.2	372.61	7.39	74.58	22.16	14.35
RG	739.57	150	3.57	28.00%	91.43%	20.49	918.42	8.09	74.16	29.16	21.44
BA	732	150	3	100.00%	NA	19.61	395.71	7.70	70.82	21.65	9.41

TABLE III
ISP TOPOLOGIES ($I_{\text{MAX}} = 8 \times n$ FOR THE FIRST 5 NETWORKS, AND $I_{\text{MAX}} = \infty$ FOR THE LAST 3)

ISP	n	m	κ	r_{succ}	Υ	t_{STPC} (s)	t_{RWPC} (s)	t_{STLI} (ms)	t_{MILI} (ms)	h_{STPC}	h_{RWPC}
Abovenet	294	182	117	80.00%	99.61%	10.12	58.20	2.46	5.08	5.68	4.03
EBONE	381	172	55	75.00%	99.69%	13.65	139.37	3.78	11.06	9.61	7.00
Tiscali	404	240	138	70.00%	99.67%	28.07	171.58	3.81	10.71	7.05	4.89
Exodus	434	201	85	67.00%	99.76%	21.13	226.15	4.13	14.49	8.26	6.13
Telstra	758	318	164	24.00%	99.76%	80.38	2999.96	6.70	118.17	7.86	6.22
AT&T	2078	631	208	NA	NA	685.46	131.1 hrs	19.50	1302.85	23.48	11.33
Sprintlink	2268	604	163	NA	NA	608.18	46.8 hrs	20.52	1560.55	15.03	11.06
Verio	2821	960	408	NA	NA	697.86	170.3 hrs	29.15	3366.79	13.22	8.97

as monitors; see MMP in [8] for detailed explanations. STPC again significantly outperforms RWPC with a speedup ranging from 6 fold (Abovenet, Tiscali) to 879 fold (Verio). In fact, RWPC becomes so slow for the largest three networks (AT&T, Sprintlink and Verio) that it is unable to complete a successful Monte Carlo run ($r_{\text{succ}} = 0\%$) even after 40 hours. To find out the time RWPC takes to find n linearly independent paths, we remove the limitation on the number of iterations ($I_{\text{MAX}} = \infty$) and let it run until success. RWPC takes up to 7 days (Verio) to complete a single Monte Carlo run[7], which is in sharp contrast with STPC that finds n linearly independent paths in 10 minutes. For link identification, STLI also outperforms MILI with a speedup ranging from 2 fold (Abovenet) to 115 fold (Verio). Over all, we observe that the running-time advantages of STPC and STLI both increase with the size of the network, while the success rate and the efficiency of RWPC decay. As in the synthetic simulations, we again observe a relatively larger path length for STPC. However, the increase in path length is only moderate compared with the decrease in running time, and this is likely the cost needed to ensure linear independence of the paths.

VI. CONCLUSION

We studied the problem of network tomography from an algorithmic perspective, proposing efficient algorithms for constructing measurement paths and uniquely identifying link metrics from path measurements. The proposed algorithms utilize a special structure of identifiable networks in the form of independent spanning trees to strategically construct linearly independent measurement paths and compute link metrics without explicitly inverting the measurement matrix. Extensive simulations on both synthetic and real networks show that the proposed algorithms can guarantee unique identification of link metrics while being orders of magnitude faster than existing solutions for large networks.

REFERENCES

[1] F. Lo Presti, N. Duffield, J. Horowitz, and D. Towsley, "Multicast-based inference of network-internal delay distributions," *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, pp. 761–775, Dec. 2002.

[2] M. Coates, A. O. Hero, R. Nowak, and B. Yu, "Internet tomography," *IEEE Signal Processing Magzine*, vol. 19, pp. 47–65, 2002.

[3] Y. Vardi, "Estimating source-destination traffic intensities from link data," *Journal of the American Statistical Assoc.*, pp. 365–377, 1996.

[4] E. Lawrence and G. Michailidis, "Network tomography: A review and recent developments," *Frontiers in Statistics*, vol. 54, 2006.

[5] O. Gurewitz and M. Sidi, "Estimating one-way delays from cyclic-path delay measurements," in *IEEE INFOCOM*, 2001.

[6] Y. Chen, D. Bindel, and R. H. Katz, "An algebraic approach to practical and scalable overlay network monitoring," in *ACM SIGCOMM*, 2004.

[7] A. Chen, J. Cao, and T. Bu, "Network Tomography: Identifiability and Fourier domain estimation," in *IEEE INFOCOM*, 2007.

[8] L. Ma, T. He, K. K. Leung, D. Towsley, and A. Swami, "Topological conditions for identifying additive link metrics via end-to-end path measurements," Technical Report, Imperial College, London, UK, Jul. 2012. [Online]. Available: http://www.commsp.ee.ic.ac.uk/%7elm110/pdf/MaTR1Jul12.pdf

[9] N. Duffield and F. Lo Presti, "Multicast inference of packet delay variance at interior network links," in *IEEE INFOCOM*, 2000.

[10] Y. Xia and D. Tse, "Inference of link delay in communication networks," *IEEE Journal of Selected Areas in Communications*, 2006.

[11] A. Adams, T. Bu, T. Friedman, J. Horowitz, D. Towsley, R. Caceres, N. Duffield, F. Presti, and V. Paxson, "The use of end-to-end multicast measurements for characterizing internal network behavior," *IEEE Communications Magazine*, vol. 38, no. 5, pp. 152–159, May 2000.

[12] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: recent developments," *Statistical Science*, 2004.

[13] M.-F. Shih and A. Hero, "Unicast inference of network link delay distributions from edge measurements," in *IEEE ICASSP*, 2001.

[14] W. Xu, E. Mallada, and A. Tang, "Compressive sensing over graphs," in *IEEE INFOCOM*, 2011.

[15] A. Gopalan and S. Ramasubramanian, "On identifying additive link metrics using linearly independent cycles and paths," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, 2011.

[16] N. Alon, Y. Emek, M. Feldman, and M. Tennenholtz, "Economical graph discovery," in *Symposium on Innovations in Computer Science*, 2011.

[17] G. H. Golub and C. F. Van-Loan, *Matrix Computations*. The Johns Hopkins University Press, Baltimore and London, 1996.

[18] R. Diestel, *Graph theory*. Springer-Verlag Heidelberg, New York, 2005.

[19] J. E. Hopcroft and R. E. Tarjan, "Dividing a graph into triconnected components," *SIAM Journal on Computing*, vol. 2, pp. 135–158, 1973.

[20] J. Cheriyan and S. N. Maheshwari, "Finding nonseparating induced cycles and independent spanning trees in 3-connected graphs," *Journal of Algorithms*, vol. 9, pp. 507–537, 1988.

[21] L. Ma, T. He, K. K. Leung, D. Towsley, and A. Swami, "Efficient identification of additive link metrics: Theorem proof and evaluations," Technical Report, Imperial College, London, UK, Nov. 2012. [Online]. Available: http://www.commsp.ee.ic.ac.uk/%7elm110/pdf/MaTechreportNov12.pdf

[22] P. Erdös and A. Rényi, "On the evolution of random graphs," *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, vol. 5, pp. 17–61, 1960.

[23] P. Gupta and P. Kumar, "Critical power for asymptotic connectivity in wireless networks," *Stochastic Analysis, Control, Optimization and Applications*, pp. 547–566, 1999.

[24] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, pp. 47–97, Jan. 2002.

[25] "Rocketfuel: An ISP topology mapping engine," University of Washington, 2002. [Online]. Available: http://www.cs.washington.edu/research/networking/rocketfuel/interactive/

[7]For this reason, we only conduct one Monte Carlo run for each of these three networks.