Scalpel: Scalable Preferential Link Tomography Based on Graph Trimming

Yi Gao, *Member, IEEE*, Wei Dong, *Member, IEEE*, Wenbin Wu, Chun Chen, *Member, IEEE*, Xiang-Yang Li, *Fellow, IEEE, Member, ACM*, and Jiajun Bu, *Member, IEEE, Member, ACM*

***Abstract*—Inferring per-link metrics through aggregated path measurements, known as *network tomography*, is an effective way to facilitate various network operations, such as network monitoring, load balancing, and fault diagnosis. We study the problem of identifying additive link metrics of a set of *interesting links* from end-to-end cycle-free path measurements among selected monitors, i.e., *preferential link tomography*. Since assigning a node as a monitor usually requires non-negligible operational cost, we focus on assigning the minimum number of monitors (i.e., optimal monitor assignment) to identify all interesting links. By modeling the network as a connected graph, we propose *Scalpel*, a scalable preferential link tomography approach. Scalpel trims the original graph by a two-stage graph trimming algorithm and reuses an existing method to assign monitors in the trimmed graph. We theoretically prove Scalpel has several key properties: 1) the graph trimming algorithm in Scalpel is minimal in the sense that further trimming the graph does not reduce the number of monitors; 2) the obtained assignment is able to identify all interesting links in the original graph; and 3) an optimal monitor assignment in the graph after trimming is also an optimal monitor assignment in the original graph. We implement Scalpel and evaluate it based on both synthetic topologies and real network topologies. Compared with state-of-the-art, Scalpel reduces the number of monitors by 39.0% to 98.6% when 50% to 1% of all links are interesting links.**

***Index Terms*—Graph trimming, network measurement, preferential link tomography.**

I. Introduction

INFERRING fine-grained network characteristics using aggregated measurements, known as *network tomography* [1], is an effective technique to facilitate various network operations [2]–[4], such as network monitoring, load balance, and fault diagnosis. In communication networks, a subset of nodes with monitoring capabilities, i.e., *monitors*, can initiate and collect end-to-end measurements of selected cycle-free paths. Then, by using these *end-to-end* metrics, network tomography techniques can decompose them to *hop-by-hop* link metrics by solving a system of equations. In many cases, these link metrics are additive [2], [5]. For example, delay is a typical additive metric since an end-to-end path delay is the sum of all link delays, while a multiplicative metric like packet delivery ratio can be expressed in an additive form by applying the $\log(\cdot)$ function. In order to identify additive link metrics, we need to solve a linear system, 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 along a path [2].

Since these end-to-end path measurements are conducted among monitors, the monitor assignment (which nodes should be assigned as monitors) becomes a key problem. On one hand, the monitor assignment should comply with certain conditions to enable a sufficient number of linearly independent measurements. For example, assume v is a node with degree two (the degree of a node is the number of links incident to it) in a network and l_1, l_2 are two links incident to v. If node v is not assigned as a monitor, no matter how to conduct simple path (i.e., cycle-free path) measurements among other monitors, we can only calculate the sum of the link metrics of l_1, l_2, instead of their individual link metrics. On the other hand, we usually want to minimize the number of monitors assigned, since assigning a node as a monitor usually needs non-negligible operational cost (e.g., hardware/software, human efforts). Ma *et al.* [2] successfully solved the above problem by proposing the MMP algorithm which identifies *all* link metrics in the network by assigning the *minimum* number of monitors.

However, inferring all link metrics may incur a high overhead which is not necessary in many applications. For example, in a network from the Rocketfuel project [6], 117 out of 182 nodes should be assigned as monitors to identify all link metrics. In practice, network managers are usually interested in a subset of links, instead of all links in the network. For example, in the Internet, some links (e.g., problematic links reported by customers or links located in critical infrastructures such as hospitals and fire departments) are known to be more important than other links. In sensor networks [7]–[9], links near the basestation are more important since they usually carry a large volume of traffic. By identifying these interesting links (i.e., preferential links), we can significantly reduce the monitoring overhead.

Manuscript received August 23, 2014; revised January 27, 2015; accepted February 23, 2015; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor R. Teixeira. Date of publication nulldate; date of current version nulldate. This work was supported in part by the National Science Foundation of China under Grant 61472360, the Fundamental Research Funds for the Central Universities, National Key Technology R&D Program under Grant 2012BAI34B01, and the Demonstration of Digital Medical Service and Technology in Destined Region. The work of X.-Y. Li was supported in part by the National Science Foundation under Grant ECCS-1247944, NSF CMMI 1436786, and the National Natural Science Foundation of China under Grant 61228202. *(Corresponding author: Wei Dong.)*

Y. Gao, W. Dong, W. Wu, C. Chen, and J. Bu are with the College of Computer Science and the Zhejiang Provincial Key Laboratory of Service Robot, Zhejiang University, Zhejiang, China (e-mail: dongw@zju.edu.cn).

X.-Y. Li is with the Department of Computer Science, Illinois Institute of Technology, USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2015.2411691

Given a network topology and a set of *interesting links*, how to assign a minimum number of monitors to identify these interesting link metrics is challenging due to the following three reasons. First, since measurement paths can cross boundaries of different sub-graphs (e.g., bi-connected/tri-connected components [10], [11]), the monitor assignment at these boundaries depends on multiple graph components. As a result, assigning monitors for each graph component separately is not sufficient. Second, it is difficult to obtain the dependency between the identifiability of a link and a particular monitor. Thus, it is difficult to implement a naive method that removes the monitors on which no interesting links depend. Third, a simple improvement of MMP that ignores the graph components without any interesting links cannot solve the preferential link tomography problem effectively. The reason is that MMP needs to identify all links, so that the nodes at the boundaries of multiple graph components can be viewed as monitors. However, in the preferential link tomography problem, these nodes cannot be viewed as monitors, causing the monitor assignment to be a challenging task.

As a first step towards addressing the monitor assignment problem of preferential network tomography, we propose Scalpel (a preliminary version of this work can be found in [12]) which carefully trims the original network graph without sacrificing the optimal solution and uses an existing method to assign monitors. Although it is difficult to obtain the dependency between link identifiability and a particular monitor, we are able to effectively narrow down the search space by safely trimming unrelated network components. After graph trimming, Scalpel reuses the MMP algorithm [2] for monitor assignment. The preferential link tomography problem studied in this paper generalizes the problem studied in MMP in which all links are considered as interesting links. It is more flexible and practical to be able to assign monitors with any given interesting links. Scalpel has several salient features. The graph trimming algorithm in Scalpel is minimal in the sense that further trimming the graph does not reduce the number of monitors that should be assigned to identify all interesting links (formally given by Theorem V.3). As for the correctness of Scalpel, we can prove that Scalpel can identify all interesting links in the original graph, including trimmed links (formally given by Theorem V.1). Scalpel also opens the door for the minimum number of monitor assignment due to the following property: an optimal monitor assignment in the graph after trimming is also an optimal monitor assignment in the original graph (formally given by Theorem V.2).

We implement Scalpel and evaluate it through extensive simulations based on both synthetic topologies and real network topologies. The time complexity of Scalpel is linear in terms of the number of vertices and links, making it able to assign monitors in large scale networks efficiently. Results show that when 50% to 1% of all links are interesting links, Scalpel reduces the number of monitors by 42.7% to 98.6% and 39% to 96% in synthetic topologies and real network topologies, respectively.

The contributions of this paper are summarized as follows.

- We are the first to identify the preferential link tomography problem to efficiently infer the metrics of a set of interesting links.
- We propose Scalpel, a scalable preferential link tomography approach. Scalpel trims the original graph by a two-stage graph trimming algorithm and reuses an existing method to assign monitors in the trimmed graph. We prove that the graph trimming algorithm in Scalpel is minimal in the sense that further trimming the graph cannot reduce the number of monitors that should be assigned to identify all interesting links. We also prove that the obtained assignment is able to identify all interesting links in the original graph.
- We prove that an optimal monitor assignment in the graph after trimming is also an optimal monitor assignment in the original graph. This property opens the door for the minimum number of monitor assignment based on our graph trimming algorithm.
- We implement and evaluate Scalpel by extensive simulations. The time complexity of Scalpel is linear in terms of the number of vertices and links, making it be able to trim large scale networks efficiently. Results shows that Scalpel is able to reduce the number of monitors by 39.0% to 98.6% when 50% to 1% of all links are interesting links.

The remainder of this paper is organized as follows. Section II presents the problem formulation. Section III gives some graph theory fundamentals used in this paper. Section IV describes Scalpel in detail. Section V theoretically analyzes Scalpel and proves its three important properties. Section VI presents the evaluation of Scalpel. Section VII discusses the related work. Finally, Section VIII concludes this paper.

II. Problem Formulation

This section gives the formulation of the minimum monitor assignment problem for identifying a set of interesting links.

A. Network Model and Assumptions

We assume that the network topology is known and does not change during the measurement period. We model the network topology as an undirected graph $\mathcal{G} = (V(\mathcal{G}), L(\mathcal{G}))$, where $V(\mathcal{G})$ and $L(\mathcal{G})$ are the sets of vertices and links, respectively. Without loss of generality, we can assume that graph $\mathcal{G}$ is connected, since different connected components of the network can be monitored separately. We denote the link that incidents to vertices u and v by uv. We assume the links are symmetric, i.e, the link metrics of uv and vu are the same. We further assume that there is no self-loop link in $L(\mathcal{G})$, and there is at most one link connecting two vertices. A set $\mathcal{I} \subseteq L(\mathcal{G})$ is a set of interesting links. A subset of vertices in $V(\mathcal{G})$ are assigned as monitors and can initiate/collect end-to-end measurements for identifying the metrics of links in $\mathcal{I}$. Since routing along cycles is typically prohibited in real networks [2], cycle-free measurement paths among monitors are preferred.

B. Problem Formulation

We assume that monitors can control the routing of measurement packets. This routing scheme is known as source routing (rfc0791 [13]). Source routing allows a sender of a packet to specify the route the packet takes through the network. Although source routing is not widely used in the current Internet, we believe our work is useful due to the following reasons.

First, source routing is generally supported in networks like overlay networks and single-ISP networks [2]. For example, one can take measurements of a collection of overlay multi-hop paths and use them to calculate the metrics for individual overlay links. Second, recent studies [14] show that source routing can help address problems like convergence and controller placement in networks performing Software-Defined Networking (SDN). When source routing is used in SDN, using network tomography to calculate the link metrics could be a more efficient and light-weight method, compared with each router to directly measure and report link metrics. Third, many studies [15]–[18] show that source routing may possess lower latency or greater available bandwidth, by partially or fully specifying the routing paths taken by packets.

A monitor m_A can initiate a measurement packet to another monitor m_B through a *simple path*. A simple path does not contain repeated vertices. Monitor m_B can obtain the path measurement, which is the sum of all link metrics along the path $\mathcal{P}$. In order to simplify the presentation, we use l (or $\mathcal{P}$) to denote both the link (or path) and its link (or path) metric. γ path measurements are obtained by sending measurement packets among monitors. Then we can build a linear system to identify the interesting links. Let $n = |L|$ be the number of all links in L, $\mathbf{l} = (l_1, \dots, l_n)^T$ be the column vector of all link metrics, and $\mathbf{p} = (\mathcal{P}_1, \dots, \mathcal{P}_\gamma)^T$. The relationship between $\mathbf{l}$ and $\mathbf{p}$ can be formulated as the following linear system:

$$\mathbf{R}\mathbf{l} = \mathbf{p} \tag{1}$$

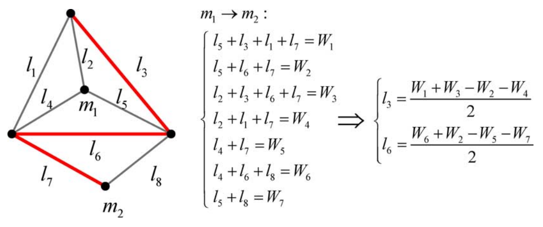
where $\mathbf{R} = (R_{ij})$ is a $\gamma \times n$ *measurement matrix*. $R_{ij} \in \{0, 1\}$ denotes whether link l_j is in measurement path $\mathcal{P}_i$. The minimum monitor assignment problem is how to find the minimum number of monitors such that there exists a measurement matrix $\mathbf{R}$ which makes all interesting link metrics in $\mathbf{l}$ be solvable in the above linear system. If the link metric can be calculated from the linear system obtained by a monitor assignment, this link is *identifiable* by the monitor assignment.

C. A Toy Example

Fig. 1 shows a toy network with five vertices and eight links (l_1 to l_8). Among these links, three of them are interesting links (i.e., $\mathcal{I} = \{l_3, l_6, l_7\}$). In order to identify the interesting link metrics, we conduct seven path measurements from m_1 to m_2. By solving the linear system shown in the figure, we can identify the link metrics of l_3 and l_6. However, the link metric of l_7 cannot be identified in this example, no matter how we conduct path measurements between the two monitors. In fact, two monitors are not sufficient to identify all the three interesting links, no matter how we assign monitors in the network. Given the monitor assignment $\{m_1, m_2\}$, links l_3, l_6 are identifiable while link l_7 is not identifiable. In the following sections, we will describe our method Scalpel to perform efficient preferential link tomography.

III. Fundamentals on Monitor Assignment

We first introduce several concepts in graph theory, which are used in this paper.



$$m_1 \rightarrow m_2 :$$

$$\begin{cases} l_5 + l_3 + l_1 + l_7 = W_1 \\ l_5 + l_6 + l_7 = W_2 \\ l_2 + l_3 + l_6 + l_7 = W_3 \\ l_2 + l_1 + l_7 = W_4 \\ l_4 + l_7 = W_5 \\ l_4 + l_6 + l_8 = W_6 \\ l_5 + l_8 = W_7 \end{cases} \Rightarrow \begin{cases} l_3 = \dfrac{W_1 + W_3 - W_2 - W_4}{2} \\ l_6 = \dfrac{W_6 + W_2 - W_5 - W_7}{2} \end{cases}$$

Fig. 1. Toy example. Two monitors are used to identify link metrics. In this example, the metrics of two interesting links l_3, l_6 can be identified by solving the linear system in the right part. However, the other interesting link l_7 cannot be identified if only these two monitors are assigned.

- A graph is *connected* when there exists at least one path from any vertex to any other vertex in the graph.
- A graph (or component) is *bi-connected* when the graph is still connected after removing one arbitrary vertex and the links incident to it. It includes at least two vertices (i.e., single link). We also call a graph with one link and its two endpoints as a bi-connected graph (or component). Note that partitioning a connected graph into bi-connected components is a classical graph theory problem which can be solved in linear time [10].
- A graph (or component) is *tri-connected* when the graph is still connected after removing any two vertices and the links incident to them. It includes at least three vertices. A triangle is also treated as a tri-connected component in this paper. Note that there exists a classical graph theory algorithm called SPQR-tree[1] which can partition a bi-connected graph (more than two nodes) into a number of tri-connected components and/or circles in linear time. We refer to these tri-connected components and the circles as SPQR components in this paper.
- A *1-cut-v* for a connected graph $\mathcal{G}$ is a vertex whose removal will disconnect the graph $\mathcal{G}$.
- A pair of *2-cut-vs* for a connected graph $\mathcal{G}$ are two vertices that removing one of them does not disconnect $\mathcal{G}$, but removing both disconnects $\mathcal{G}$.
- A *sep-v* is a separating vertex which could be a 1-cut-v, or a 2-cut-v, or both.
- A *cut-l* is a link which connects a pair of 2-cut-vs.
- The degree of a vertex is the number of links incident to it.

Fig. 2 shows an example with two bi-connected components separated by one 1-cut-v. The left bi-connected component A is further partitioned into one tri-connected component and one circle by a pair of 2-cut-vs. The link that connects to the two 2-cut-vs is a cut-l.

It is worth noting that the SPQR-tree algorithm will add some *virtual links* to the components after graph partitioning. In the example shown in Fig. 2, if the cut-l does not exist in the original graph, the SPQR-tree algorithm will add this link to the tri-conneected component and the cycle, as a virtual link. It has

[1]There are four possible components in a typical SPQR-tree [11]. 1) S node, which is a cycle (i.e., polygon). 2) P node, which has multiple links between two vertices. In this paper, we assume that two vertices can only have at most one link. Therefore, there is no P node after graph partition. 3) Q node, which is just a single link. In this paper, a single link is considered as a special case. 4) R node, which is a tri-connected component.

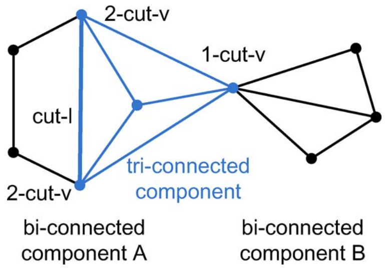


Fig. 2. Sample graph to illustrate some graph theory concepts used in this paper. The 1-cut-v separates the graph into two bi-connected components. The two 2-cut-vs separate one bi-connected components into one tri-connected component and a circle. The link cut-l connects the two 2-cut-vs.

been proved in [19] that the virtual links do not affect link identifiability of tri-connected components (except triangles). For cycles, when at least one of its neighboring SPQR components is a tri-connected component (not triangle), whether the cut-l between them is a virtual link does not affect the link identifiability. For cycles (including triangles) share one virtual link, the virtual link will affect the identifiability of these cycles (or triangles). In order to avoid this, we modify the SPQR-tree algorithm so that these cycles (or triangles) are merged to a larger cycle. As a result, the virtual link will not affect the link identifiability. In the rest of the paper, we will view the virtual links as normal links.

Then we give some important results from existing work [2], [19].

Theorem III.1: If $\mathcal{G}$ is a tri-connected graph, all of its link metrics are identifiable by assigning any three monitors.

Corollary III.2: If $\mathcal{T}$ is a tri-connected component of a graph $\mathcal{G}$, all of its link metrics are identifiable by assigning any three vertices in the original graph $\mathcal{G}$ as monitors, when the three vertices v_1, v_2, v_3 satisfy one of the following conditions. 1) v_1, v_2, v_3 are in $\mathcal{T}$; 2) one or more vertices v_i is not in $\mathcal{T}$, but there are distinct paths (without any repeated vertex) from v_i to $\mathcal{T}$.

*Definition III.1: For a tri-connected graph $\mathcal{G}$ (or component) and two vertices v_1, v_2 in it, its **interior links** are defined as links that do not incident to either of the two vertices; and its **exterior links** are defined as links that incident to only one vertex (v_1 or v_2). Note that a direct link connecting these two vertices is not an exterior link.*

Theorem III.3: For a tri-connected component $\mathcal{T}$ with two monitors assigned in it (or two sep-vs which connect to two monitors through two paths without repeated vertices outside $\mathcal{T}$, or one monitor in $\mathcal{T}$ and one such sep-v), all its interior links are identifiable and all its exterior links are unidentifiable.

Fig. 3 shows a typical tri-connected component $\mathcal{T}$ and two vertices m_1, m_2 which connects to v_1, v_2 through two paths. Theorem III.1 says if we assign three monitors in $\mathcal{T}$ (e.g., v_1, v_2, m_3), all links in $\mathcal{T}$ are identifiable. Corollary III.2 says if we assign m_1, m_2, m_3 as monitors, all links in $\mathcal{T}$ are still identifiable. Link l_2 is an interior link w.r.t. vertices v_2, m_3. Link l_1, l_3, l_4 and l_5 are exterior links w.r.t. vertices v_2, m_3. Theorem III.3 says if we assign two monitors v_2, m_3, the interior links (i.e., l_2) can be identified and the exterior links (i.e., l_1, l_3, l_4, l_5) cannot be identified. Note that l_6 is not an exterior link since it incidents both of the two vertices v_2, m_3. Another example is that if we assign m_2, m_3 as two monitors, the interior links (i.e., l_2) can still be identified and the exterior links (i.e., l_1, l_3, l_4, l_5) cannot be identified.

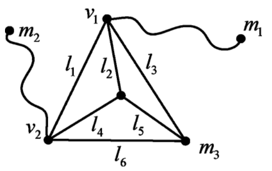


Fig. 3. Typical tri-connected graph to illustrate several results from previous works. l_2 is an interior link w.r.t. vertices v_2, m_3. l_2 can be identified by two monitors $\{v_2, m_3\}$ or $\{m_2, m_3\}$.

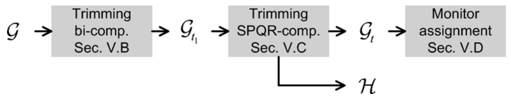

$\mathcal{G}$
Trimming bi-comp. Sec. V.B
$\mathcal{G}_{t_1}$
Trimming SPQR-comp. Sec. V.C
$\mathcal{G}_t$
Monitor assignment Sec. V.D
$\mathcal{H}$

Fig. 4. Overview of Scalpel, which first trims the graph in two stages and assign monitors in the trimmed graph.

IV. Scalpel

Here, we describe the proposed Scalpel algorithm in detail. First, given the original graph and a set of interesting links, Scalpel trims the original graph by a novel graph trimming algorithm. Then Scalpel assigns monitors in the trimmed graph. Fig. 4 shows the overview of Scalpel. Scalpel trims the original graph $\mathcal{G}$ in two stages. The first stage trims some bi-connected components in $\mathcal{G}$ and outputs an intermediate trimmed graph $\mathcal{G}_{t_1}$. The second stage trims some SPQR components (i.e., tri-connected components or circles) in the graph $\mathcal{G}_{t_1}$ and outputs a trimmed graph $\mathcal{G}_t$ and a set of *helper* vertices $\mathcal{H}$. $\mathcal{H}$ is a subset of $V(\mathcal{G} - \mathcal{G}_t)$. Detailed information about choosing these helper vertices will be given in Section IV-C. Based on the trimmed graph $\mathcal{G}_t$, Scalpel assigns monitors to identify interesting links in the original graph $\mathcal{G}$.

Definitions

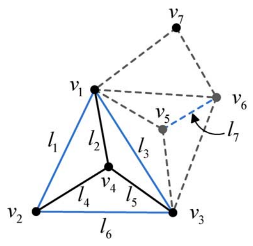
Before describing Scalpel in detail, we first give the definitions of a *monitor assignment* and an *optimal monitor assignment* formally.

Definition IV.1: A monitor assignment $\mathcal{M}(\mathcal{G}_x)$ is a subset of vertex set $V(\mathcal{G}_x)$, in which each vertex is assigned as a monitor. Here, $\mathcal{G}_x$ represents any graph, such as the original graph $\mathcal{G}$ and the trimmed graph $\mathcal{G}_t$.

In particular, $\mathcal{M}(\mathcal{G}_t \cup \mathcal{H})$ is a set of vertices in $V(\mathcal{G}_t) \cup \mathcal{H}$ which are assigned as monitors. Fig. 5 gives an example. The dotted part of the graph is trimmed and the solid part is the trimmed graph $\mathcal{G}_t$. A helper vertex v_7 is in the dotted part. A monitor assignment $\mathcal{M}(\mathcal{G}_t \cup \mathcal{H})$ is $\{v_4, v_7\}$. Since we want to assign minimum number of monitors, we have the following definition about optimal monitor assignment.

*Definition IV.2: An optimal monitor assignment $\mathcal{M}^*(\mathcal{G}_x)$ is a monitor assignment $\mathcal{M}(\mathcal{G}_x)$, in which the number of monitors is **minimum** for identifying all the interesting links in the **original graph** $\mathcal{G}$.*

It is worth noting that an optimal monitor assignment $\mathcal{M}^*(\mathcal{G}_x)$ is able to identify all interesting links in the **original**



$$L(\mathcal{G}_t)=\{l_1,l_2,l_3,l_4,l_5,l_6\}$$
$$V(\mathcal{G}_t)=\{v_1,v_2,v_3,v_4\}$$
$$\mathcal{H}=\{v_7\}$$
$$\mathcal{I}=\{l_1,l_3,l_6,l_7\}$$
$$\mathcal{M}^*(\mathcal{G}_t\cup\mathcal{H})=\{v_4,v_7\}$$
$$\mathcal{M}^*(\mathcal{G})=\{v_4,v_7\}$$

Fig. 5. Example that illustrates a monitor assignment $\mathcal{M}(\mathcal{G}_t\cup\mathcal{H}) = \{v_4, v_7\}$. In order to identify the four interesting links (l_1, l_3, l_6, l_7), assigning v_4 and v_7 as monitors are optimal.

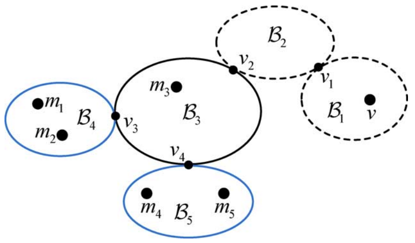


Fig. 6. Example to illustrate the first stage of graph trimming. Bi-connected components $\mathcal{B}_1$ and $\mathcal{B}_2$ without interesting links are trimmed in this stage.

graph $\mathcal{G}$, instead of the graph $\mathcal{G}_x$. According to this definition, we have the following two optimal monitor assignments: 1) optimal monitor assignment $\mathcal{M}^*(\mathcal{G}_t \cup \mathcal{H})$ in $V(\mathcal{G}_t) \cup \mathcal{H}$ (i.e., vertices in the trimmed graph $\mathcal{G}_t$ and the helper vertices set $\mathcal{H}$) and 2) the optimal monitor assignment $\mathcal{M}^*(\mathcal{G})$ in $V(\mathcal{G})$. The difference between $\mathcal{M}^*(\mathcal{G}_t \cup \mathcal{H})$ and $\mathcal{M}^*(\mathcal{G})$ is that the former can only assign vertices in $V(\mathcal{G}_t) \cup \mathcal{H}$ as monitors and the latter can assign vertices in $V(\mathcal{G})$ as monitors. Since $(V(\mathcal{G}_t) \cup \mathcal{H}) \subseteq V(\mathcal{G})$, $\mathcal{M}^*(\mathcal{G})$ has more choices as monitors. In Fig. 5, $V(\mathcal{G}_t) \cup \mathcal{H} = \{v_1, v_2, v_3, v_4, v_7\}$ and $V(\mathcal{G}) = \{v_1, v_2, \ldots, v_7\}$. Assigning v_4 and v_7 as monitors is able to identify all the interesting links in $\mathcal{I}$ $(= \{l_1, l_3, l_6, l_7\})$ in the original graph. In this example, $\{v_4, v_7\}$ is both an optimal monitor assignment $\mathcal{M}^*(\mathcal{G})$ in $V(\mathcal{G})$ and an optimal assignment $\mathcal{M}^*(\mathcal{G}_t \cup \mathcal{H})$ in $V(\mathcal{G}_t) \cup \mathcal{H}$. Note that these two kinds of optimal monitor assignments are not unique, that is, different assignments which are able to identify all interesting links may have the same number of monitors, which are also optimal.

A. Trimming Bi-Connected Components

The first stage graph trimming iteratively trims bi-connected components with no interesting links. Algorithm 1 gives the first stage graph trimming algorithm. The input of this algorithm is a connected graph $\mathcal{G} = \{V(\mathcal{G}), L(\mathcal{G})\}$ and a set $\mathcal{I} \subseteq L(\mathcal{G})$ of interesting links. Scalpel first partitions the graph $\mathcal{G}$ into bi-connected components (line 1). Then Scalpel inserts all bi-connected components with no interesting links into a queue (line 2 to 5). Then Scalpel iteratively trims bi-connected components which only have one 1-cut-v and have no interesting links (line 6 to 13). The first stage graph trimming only needs to traverse the graph once. Further, partitioning a connected graph into bi-connected components can be done in linear time [10]. Therefore, the time complexity of the first stage graph trimming is $O(|V(\mathcal{G})| + |L(\mathcal{G})|)$.

Algorithm 1 First Stage Graph Trimming

```
Input: A connected graph $\mathcal{G}$, a set $\mathcal{I}$ of interesting links
Output: The trimmed graph $\mathcal{G}_{t_1}$
1: partition $\mathcal{G}$ into bi-connected components $\mathcal{B}_1, \mathcal{B}_2, \ldots$
2: Let Queue be a queue for bi-connected components
3: for each bi-connected component $\mathcal{B}_i$ do
4:     if $\{l | l \in \mathcal{B}_i, l \in \mathcal{I}\} = \emptyset$ and $\mathcal{B}_i$ has one 1-cut-v then
5:         Queue.enqueue($\mathcal{B}_i$)
6: while Queue.notEmpty() do
7:     $\mathcal{B} \leftarrow$ Queue.dequeue()
8:     Let $\mathcal{B}_n$ be a neighbor bi-connected component of $\mathcal{B}$
9:     delete $\mathcal{B}$ except the 1-cut-v
10:    if the 1-cut-v is only in $\mathcal{B}_n$ then
11:        mark the 1-cut-v as not a cut vertex
12:    if $\{l | l \in \mathcal{B}_n, l \in \mathcal{I}\} = \emptyset$ and $\mathcal{B}_n$ has one 1-cut-v then
13:        Queue.enqueue($\mathcal{B}_n$)
```

Fig. 6 shows an example. Five bi-connected components ($\mathcal{B}_1$ to $\mathcal{B}_5$) are separated by four 1-cut-vs (v_1 to v_4). We assume that only bi-connected component $\mathcal{B}_4$ and $\mathcal{B}_5$ have interesting links. In this case, Algorithm 1 will first insert $\mathcal{B}_1$ to the queue. Then $\mathcal{B}_1$ is trimmed and $\mathcal{B}_2$ is inserted to the queue. Then $\mathcal{B}_2$ is trimmed. Since $\mathcal{B}_3$ has two 1-cut-vs after v_2 is marked as not a cut vertex, $\mathcal{B}_3$ is not inserted into the queue. Finally, the solid part of the graph is left after the first stage graph trimming.

B. Trimming SPQR Components

The second stage graph trimming trims some tri-connected components or circles (i.e., SPQR components) for each bi-connected component. Different with the first stage graph trimming, the SPQR components trimmed in this second stage may include some special interesting links. These links are *interior links* w.r.t. the two sep-vs s_1, s_2 in a tri-connected component $\mathcal{T}$. As defined by Definition III.1 in Section III, these interior links do not incident to either of the two sep-vs. We use $\text{int}(\mathcal{T}, s_1, s_2)$ to denote the set of interior links in $\mathcal{T}$ w.r.t. the two sep-vs s_1, s_2. Accordingly, we use $\text{ext}(\mathcal{T}, s_1, s_2)$ to denote the set of exterior links which incident one sep-v. Theorem III.3 in Section III says if we assign s_1, s_2 as monitors, all links in $\text{int}(\mathcal{T}, s_1, s_2)$ are identifiable and all links in $\text{ext}(\mathcal{T}, s_1, s_2)$ are not identifiable.

Algorithm 2 gives the second stage graph trimming algorithm. The input is the trimmed graph $\mathcal{G}_{t_1}$ after the first graph trimming stage and the interesting link set $\mathcal{I}$. The output is the trimmed graph $\mathcal{G}_t$, and a helper vertex set $\mathcal{H}$. Each helper vertex v is associated with a link l in $\mathcal{G}_t$. We use $h(l) = v$ (or $l = h^{-1}(v)$ since it is a one-to-one mapping) to denote this association. The reason of reserving the helper vertices is that sometimes assigning one helper vertex as a monitor can achieve the same identifiability as assigning two (or more) monitors in $\mathcal{G}_t$. Therefore, reserving these helper vertices keeps the possibility of finding an optimal solution after graph trimming. Since this algorithm is quite complicated, we use an example shown

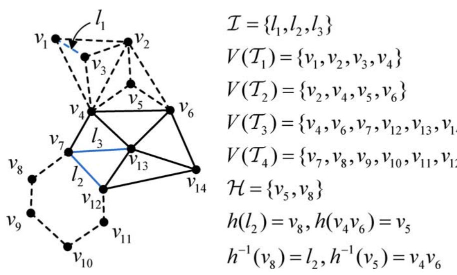

l_1
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8 v_9 v_{10} v_{11} v_{12} v_{13} v_{14}
l_2 l_3
$\mathcal{I} = \{l_1, l_2, l_3\}$
$V(\mathcal{T}_1) = \{v_1, v_2, v_3, v_4\}$
$V(\mathcal{T}_2) = \{v_2, v_4, v_5, v_6\}$
$V(\mathcal{T}_3) = \{v_4, v_6, v_7, v_{12}, v_{13}, v_{14}\}$
$V(\mathcal{T}_4) = \{v_7, v_8, v_9, v_{10}, v_{11}, v_{12}\}$
$\mathcal{H} = \{v_5, v_8\}$
$h(l_2) = v_8, h(v_4v_6) = v_5$
$h^{-1}(v_8) = l_2, h^{-1}(v_5) = v_4v_6$

Fig. 7. Example used to help describe Algorithm 2, the second stage graph trimming. The dotted part in the graph is trimmed in this stage.

in Fig. 7 to help describe the algorithm. The second stage trimming algorithm works as follows.

1) Line 1 to 6: For each bi-connected component in $\mathcal{G}_{t_1}$, Scalpel first partitions it into a number SPQR components, then inserts all SPQR components with only two sep-vs s_1, s_2 into a queue. In Fig. 7, a bi-connected component $\mathcal{G}_{t_1}$ is partitioned into four SPQR components: $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ and $\mathcal{T}_4$. Then $\mathcal{T}_1$ and $\mathcal{T}_4$ are inserted into the queue.

2) Line 7 to 13: For each circle $\mathcal{T}$ in the queue, Scalpel checks whether there are interesting links in $L(\mathcal{T}) - s_1s_2$ (line 10). If there is no such interesting links, Scalpel chooses a vertex (not s_1, s_2) as a helper vertex for link s_1s_2. Then Scalpel deletes $\mathcal{T}$ except s_1, s_2. In Fig. 7, the SPQR component $\mathcal{T}_4$ is deleted since it does not include any interesting link except the cut-l l_2. Then v_8 is chosen as a helper vertex for the cut-l l_2 ($h(l_2) = v_8$).

3) Line 14 to 23: For each tri-connected component $\mathcal{T}$ without any interesting exterior links (line 14), there are three cases before deleting $\mathcal{T}$:1) if there exists a helper vertex $h(l)$ of a link l in $L(\mathcal{T})$ and no interesting link incidents to the helper vertex, Scalpel associates the cut-l s_1s_2 with the helper vertex $h(l)$ (line 14 to 16); 2) if there exists a vertex v in $V(\mathcal{T}) - \{s_1, s_2\}$ and no interesting link incidents to v, add v as a helper vertex and associates the cut-l s_1s_2 with v (line 17 to 19, $L(v)$ in line 17 represents a set of links incident to node v); and 3) if neither of the previous two cases holds, choose a vertex v in $V(\mathcal{T}) - \{s_1, s_2\}$ as a helper vertex and associates the cut-l s_1s_2 with v (line 20 to 22). After adding the helper vertex associated with the cut-l s_1s_2, Scalpel deletes $\mathcal{T}$ except s_1, s_2. In Fig. 7, two tri-connected components $\mathcal{T}_1, \mathcal{T}_2$ are deleted in two iterations since there are no interesting exterior links. In the first iteration, tri-connected component $\mathcal{T}_1$ meets the third case. Then v_3 is chosen as a helper vertex that associates the cut-l v_2v_4. In the second iteration, tri-connected component $\mathcal{T}_2$ meets the first case. Then, v_5 is chosen as a helper vertex that associates the cut-l v_4v_6. Note that the helper vertex v_3 that associates with link v_2v_4 is deleted when v_2v_4 is deleted along with the tri-connected component $\mathcal{T}_2$.

4) Line 24 to 29: If an SPQR component $\mathcal{T}$ is deleted, its neighboring SPQR components (i.e., $\mathcal{T}_n$ in line 25) need some additional operations for the next iteration. Specifically, it is possible that some SPQR components have two 2-cut-vs (i.e., meets the condition in line 28) after the deletion of the SPQR component $\mathcal{T}$. Therefore, these SPQR components need to be inserted to the queue. In Fig. 7, before the deletion of $\mathcal{T}_1$, $\mathcal{T}_2$ is not in the queue since it has three 2-cut-vs v_2, v_4, v_6. After $\mathcal{T}_1$ is deleted, $\mathcal{T}_2$ is inserted into the queue, since it has two 2-cut-vs v_4, v_6.

The second stage graph trimming needs to partition each bi-connected component into SPQR components and traverse a number of SPQR components (line 5, 28 in Algorithm 2). Although there is a "for" loop inside a "while" loop, each tri-connected component is only inserted into the queue at most once. Since the SPQR partition can be done in linear time [11], the time complexity of the second stage graph trimming is $O(|V(\mathcal{G}_{t_1})| + |L(\mathcal{G}_{t_1})|)$, where $\mathcal{G}_{t_1}$ is the graph after the first stage graph trimming. Since the time complexity of the first stage graph trimming is also linear (Section IV-B), the time complexity of the two-stage graph trimming is linear in terms of the number of vertices and links.

Algorithm 2 Second Stage Graph Trimming

```
Input: A graph $\mathcal{G}_{t_1}$, a set $\mathcal{I}$ of interesting links
Output: Trimmed graph $\mathcal{G}_{t_2}$ (i.e., $\mathcal{G}_t$), helper vertex set $\mathcal{H}$
1: for each bi-connected component $\mathcal{B}_i$ do
2:     partition $\mathcal{B}_i$ into SPQR components $\mathcal{T}_1, \mathcal{T}_2, \dots$
3:     Let Queue be a queue for SPQR components
4:     for each SPQR components $\mathcal{T}_j$ do
5:         if $\mathcal{T}_j$ has only two sep-v $s_1, s_2$ then
6:             Queue.enqueue($\mathcal{T}_j$)
7:     while Queue.notEmpty() do
8:         $\mathcal{T} \leftarrow$ Queue.dequeue()
9:         if $\mathcal{T}$ is a circle then
10:            if $(L(\mathcal{T}) - s_1s_2) \cap \mathcal{I} = \emptyset$ then
11:                $\mathcal{H} = \mathcal{H} \cup \{v\},\ v \in (V(\mathcal{T}) - \{s_1, s_2\})$
12:                $h(s_1s_2) = v$
13:                delete $\mathcal{T}$ except $s_1s_2$
14:        else if $\mathrm{ext}(\mathcal{T}, s_1, s_2) \cap \mathcal{I} = \emptyset$ then
15:            if $\exists h(l) \in \mathcal{H}, l \in L(\mathcal{T}), h(l) \cap V(\mathcal{I}) = \emptyset$ then
16:                $h(s_1s_2) = h(l)$
17:            else if $\exists v \in (V(\mathcal{T}) - \{s_1, s_2\}), L(v) \cap \mathcal{I} = \emptyset$ then
18:                $\mathcal{H} = \mathcal{H} \cup \{v\}$
19:                $h(s_1s_2) = v$
20:            else
21:                $\mathcal{H} = \mathcal{H} \cup \{v\},\ v \in (V(\mathcal{T}) - \{s_1, s_2\})$
22:                $h(s_1s_2) = v$
23:            delete $\mathcal{T}$ except $s_1s_2$
24:        if $\mathcal{T}$ is deleted then
25:            for each $\mathcal{T}_n, |V(\mathcal{T}) \cap V(\mathcal{T}_n)| > 0$ do
26:                if $s \in \{s_1, s_2\}$ are only in $\mathcal{T}_n$ then
27:                    mark s as not 2-cut-v
28:                if $\mathcal{T}_n$ has two sep-vs then
29:                    Queue.enqueue($\mathcal{T}_n$)
```

C. Monitor Assignment

After the two-stage graph trimming, Scalpel has a trimmed graph $\mathcal{G}_t$ and a set of helper vertices $\mathcal{H}$. Based on the trimmed graph $\mathcal{G}_t$, Scalpel assign monitors in $\mathcal{G}_t$ to identify interesting links in $\mathcal{G}_t$. We will prove that if a monitor assignment $\mathcal{M}(\mathcal{G}_t)$ can identify all links in $\mathcal{G}_t$, it can also identify interesting links in $\mathcal{G}$ (Theorem V.1). This property of Scalpel enables us to assign monitors in the trimmed graph, which is usually much smaller than the original graph when there are a small number of interesting links. In the current implementation, Scalpel reuses an existing method called MMP [2] to assign monitors in $\mathcal{G}_t$. MMP is able to assign the minimum number of monitors in a graph to identify *all* links. According to the above property, this assignment is able to identify the interesting links in the original graph $\mathcal{G}$.

Based on the output of graph trimming (the trimmed graph $\mathcal{G}_t$ and the helper vertices set $\mathcal{H}$), we will prove that an optimal monitor assignment $\mathcal{M}^*(\mathcal{G}_t \cup \mathcal{H})$ in $V(\mathcal{G}_t) \cup \mathcal{H}$ has the same number of monitors as an optimal monitor assignment $\mathcal{M}^*(\mathcal{G})$ in $V(\mathcal{G})$ (Theorem V.2). In other words, an optimal monitor assignment $\mathcal{M}^*(\mathcal{G}_t \cup \mathcal{H})$ is also an optimal monitor assignment $\mathcal{M}^*(\mathcal{G})$. This property opens a door for designing optimal monitor assignment algorithm based on the trimmed graph $\mathcal{G}_t$ and the helper vertices set $\mathcal{H}$. Based on the graph trimming algorithm proposed in this paper, an optimal monitor assignment for the preferential link tomography problem should carefully assign monitors in each graph component (i.e., bi-connected component/SPQR component) and the boundaries of multiple graph components. We consider designing an optimal monitor assignment as future work.

The measurement load is directly related to the probes needed to be sent to identify the interesting links. It is shown in [20] that the number of probes could be the same with the number of links in the network, when using MMP [2] to assign monitors. Since Scalpel uses MMP to assign monitors in the trimmed graph $\mathcal{G}_t$, the number of probes could be at most the number of links in the trimmed graph. For the interesting links not in $\mathcal{G}_t$, at most four probes are required to identify each interesting link [20]. Therefore, the number of probes can be linearly bounded by the number of links in the network..

V. Theoretical Analysis

Here, we give the following three theorems which describe the three important properties of Scalpel formally. The formal proofs of these theorems can be found in [21].

Theorem V.1: If a monitor assignment $\mathcal{M}(\mathcal{G}_t)$ in $V(\mathcal{G}_t)$ is able to identify all links in $\mathcal{G}_t$, it is also able to identify all interesting links in the original graph $\mathcal{G}$, including the interesting links been trimmed.

Since Scalpel reuses an existing method to assign monitors in $\mathcal{G}_t$ to identify all links in it. According to this theorem, the obtained assignment by Scalpel is able to identify all interesting links in the original graph, including those trimmed links.

Theorem V.2: An optimal monitor assignment $\mathcal{M}^*(\mathcal{G}_t \cup \mathcal{H})$ and an optimal monitor assignment $\mathcal{M}^*(\mathcal{G})$ have the same number of monitors.

Theorem V.2 says that an optimal monitor assignment in the graph after trimming is also an optimal monitor assignment in the original graph, or optimality preserving feature of the graph trimming algorithm.

Theorem V.3: The graph trimming algorithm in Scalpel is minimal in the sense that if we further trim one SPQR component $\mathcal{T}$ from $\mathcal{G}_t$ and use MMP to assign minimum monitors in $\mathcal{G}_t - \mathcal{T}$ to identify all interesting links, the number of monitors cannot be reduced.

VI. Evaluation

We evaluate Scalpel through extensive simulations on both synthetic network topologies and real network topologies. In this section, we will first give the evaluation methodology, including the evaluation metrics and detailed description of the topologies used. Then, we will show the graph trimming results and monitor assignment results of Scalpel.

A. Methodology

The first metric is the effectiveness of graph trimming. The first stage graph trimming of Scalpel trims a number of bi-connected components and the second stage graph trimming of Scalpel trims a number of SPQR components. Therefore, we report the number of bi-connected components and SPQR components which are trimmed when there are different number of interesting links. Then, we compare the monitor assignment results of Scalpel with MMP. The metrics used are the number of monitor assigned and the execution time. Note that MMP is an optimal monitor assignment algorithm to identify all links in a graph. The evaluation of Scalpel is mainly used to show reduction in the number of monitors when there are different numbers of interesting links, instead of showing that Scalpel performs better than MMP.

For each network topology, we randomly choose different number of links as interesting links: 1%, 5%, 10%, 50% and 100% of all links in the original network. We ran each simulation ten times and report the average values. We use both synthetic topologies and real network topologies.

1) Synthetic Topologies: We consider four widely used synthetic topologies. *Barabási-Albert (BA) graph* [22]. A Barabási–Albert (BA) graph is generated by the following steps. We begin with a small connected graph $(\{v_1, v_2, v_3, v_4\}, \{v_1v_2, v_1v_3, v_1v_4\})$ and add new nodes sequentially. For each new node, we connect it to two existing nodes. The probability of connecting a node w is proportional to the node degree of w. *Erdös–Rényi (ER) graph* [23]. An Erdös–Rényi (ER) graph is a simple random graph generated by connecting each pair of nodes with a fixed probability. In our simulations, this probability is set to be 0.015. *Random Geometric (RG) graph* [24]. A Random Geometric (RG) graph is generated by first randomly deploying nodes in a unit square area and then connecting nodes when their distance is no larger than a threshold. In our simulations, we set the threshold to 0.08. *dK*-graph [25]. *dK*-graph models the degree distribution of a network and generate topologies with similar degree distribution. Then *dK*-graph generates network topologies with similar degree distribution but different scales. We use a real network topology as input and use *dK*-graph tool to generate larger graphs.

2) Autonomous System Topologies: We also use real network topologies from the Rocketfuel [6] project to evaluate Scalpel. The topologies used are the autonomous system (AS) topologies, which represent IP-level connections between backbone/gateway routers of ASes from major Internet Service Providers (ISPs) (i.e., AT&T) around the world [2]. Each AS in Rocketfuel corresponds to an ISP.

TABLE I
MAIN RESULTS OF SCALPEL IN DIFFERENT NETWORKS WITH DIFFERENT NUMBER OF INTERESTING LINKS

Network	|V(G)|	|L(G)|	1%		5%		10%		50%	
			|trim-B|	|trim-T|	|trim-B|	|trim-T|	|trim-B|	|trim-T|	|trim-B|	|trim-T|
BA	300	596	0/1	139/142	0/1	125.8/142	0/1	110.4/142	0/1	34/142
ER	297	668	18.8/20	28/29	18.8/20	24.8/29	16.8/20	22.8/29	10/20	6/29
RG	296	840	13.2/17	25/31.4	11/17	20.6/34	9.8/17	13.6/34.6	4/17	2.4/35.6
dK-1009	1009	3773	167/170	97.8/101	160.6/170	89.3/101	152.6/170	81.6/101	80.1/170	23.5/101
dK-2485	2485	9402	599/607	190.4/195	574.8/607	172.5/195	547.4/607	154.5/195	299.9/607	48/195
dK-3018	3018	11275	492.3/499	289.3/298	474.3/499	261.6/298	449.3/499	243.3/298	253.3/499	69.6/298
dK-5072	5072	19367	1267.6/1282	381.3/391	1213/1282	349.3/391	1151/1282	316/391	627/1282	88/391
AS1221	318	758	138.5/142	42.3/46.6	130.2/142	40.9/52.9	122/142	35.4/53	58.8/142	9.7/53
AS1239	604	2268	101.7/104	58.7/61.2	96.9/104	53.5/62.3	92.3/104	47.6/61.75	49.9/104	13.7/63.8
AS1755	172	381	26.7/28	34/35.2	24.8/28	30.7/35.6	23.4/28	26.1/35.55	12.4/28	7.8/36.8
AS3257	240	404	104.1/107	42.1/46	98.8/107	31.2/46.6	92.5/107	27.2/47.25	45.4/107	6.8/48.85
AS3356	624	5299	28.8/30	55.9/58	28/30	50.9/58	26.1/30	45.7/58	14.5/30	14.4/58
AS3967	201	434	36.6/38	45.2/47.7	34.7/38	41.2/48.5	32.7/38	34.6/48.45	16.9/38	11.1/51
AS6461	182	296	84.3/87	36.1/37.4	81.3/87	30.8/38.2	74.3/87	28.4/39.25	40.1/87	8/41
AS7018	631	2078	56.3/58	152.7/158	54.2/58	139.3/158	50.8/58	121.8/158	27.1/58	36.4/157.95

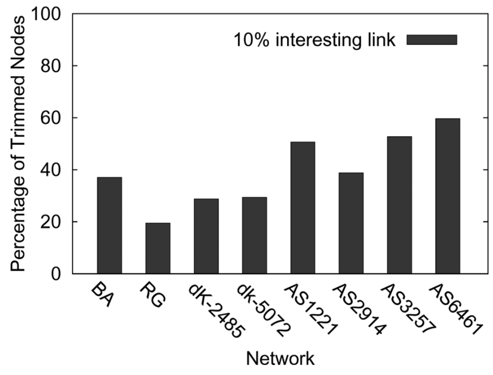


Fig. 8. Percentage of nodes trimmed by Scalpel in different networks.

B. Results

1) Effectiveness of Graph Trimming: Table I reports the graph trimming results of all topologies, including seven synthetic topologies and eight AS topologies. The first column is the topologies name. The second column and third column are the number of vertices and links in each topology. The other columns are the results in four different settings with different percentages of interesting links. $|\text{trim-}\mathcal{B}|$ denotes the number of bi-connected components which are trimmed in the first graph trimming stage. $|\text{trim-}\mathcal{T}|$ denotes the number of SPQR components which are trimmed in the second stage graph trimming. We also show the number of bi-connected components in the original graph, as well as the number of SPQR components in the graph after the first stage graph trimming. For example, the 125.8/142 entry of the first row means that when 5% of the links are interesting links, there are 142 SPQR components (on average) after the first trimming stage and 125.8 (on average) of them are trimmed by the second trimming stage.

From Table I, we first observe that when there are only a small number of interesting links (e.g., 1%), most of the bi-connected components and the SPQR components are trimmed by Scalpel. For example, when 1% links are interesting links in the *dK*-5072 topology, 1267.6 out of 1282 (on average) bi-connected components in the original graph are trimmed by the first trimming stage. Then, 381.3 out of 391 (on average) SPQR components in the graph are trimmed by the second trimming stage. Second, when the number of interesting links increases, the number of trimmed components decreases. There is also a special topology in the first row, which is a bi-connected component. In this case, the first trimming stage cannot trim any bi-connected component but the second trimming stage can still trim a large number of SPQR components.

Fig. 8 shows the percentage of nodes trimmed by Scalpel in eight different networks, including four synthetic topologies and four AS topologies. We can see that when 10% of the links are interesting links, about 20% to 60% of nodes will be trimmed by Scalpel. Compared with Table I, the percentage of trimmed nodes are smaller than the percentage of trimmed graph components. The reason is that many trimmed graph component only includes a small number of nodes and the ones left usually include more nodes.

2) Monitor Assignment: Then we show the monitor assignment results. Fig. 9 shows the results in the eight AS topologies. The x-axis is the percentage of interesting links in all links, and the y-axis is the number of monitors assigned. The blue horizontal line indicates the number of vertices in each topology. We set the percentage of interesting links to $1\%, 5\%, 10\%, 20\%, \ldots, 100\%$. From Fig. 9, we can see that when only 1% of links are interesting links, only a small number of vertices are assigned as monitors. When more links are interesting links, more monitors should be assigned. Note that using MMP directly in the original graph to assign monitors is equivalent to the case when all links are interesting links. Therefore, using Scalpel assign monitors in the trimmed graph reduces the number of monitors compared to using MMP directly in the original graph. Table II shows the monitor reduction ratios in different networks with different settings.

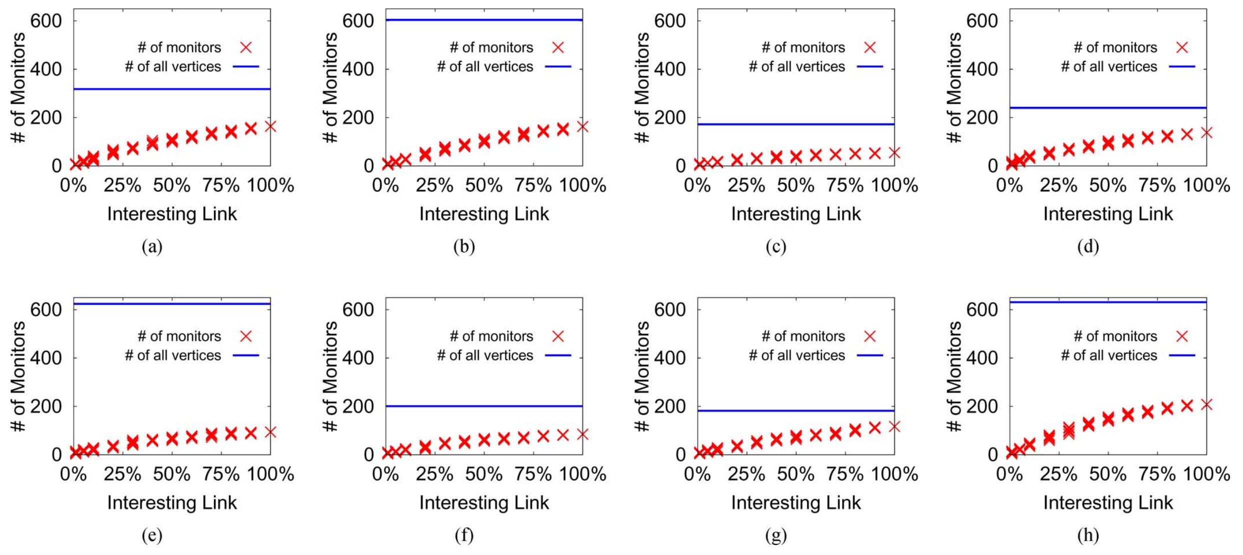


Fig. 9. Monitor assignment results of Scalpel. For each topology, we repeat each simulation 10 times and plot the results. (a) AS1221. (b) AS1239. (c) AS1755. (d) AS3257. (e) AS3356. (f) AS3967. (g) AS6461. (h) AS7018.

TABLE II
OF MONITORS AND EXEC TIME COMPARED WITH MMP

Network	1%		5%		10%		50%	
	|M|	time	|M|	time	|M|	time	|M|	time
AS1221	-0.96	+0.045	-0.88	+0.083	-0.82	+0.085	-0.34	+0.072
AS1239	-0.95	+0.033	-0.89	+0.06	-0.83	+0.047	-0.37	+0.078
AS1755	-0.89	+0.062	-0.77	+0.062	-0.69	+0.057	-0.29	+0.028
AS3257	-0.93	+0.09	-0.81	+0.054	-0.74	+0.05	-0.31	+0.045
AS3356	-0.9	+0.004	-0.82	+0.017	-0.75	+0.003	-0.29	+0.009
AS3967	-0.93	+0.041	-0.84	+0.037	-0.75	+0.053	-0.3	+0.045
AS6461	-0.94	+0.063	-0.87	+0.081	-0.8	+0.063	-0.39	+0.046
AS7018	-0.95	+0.045	-0.88	+0.051	-0.79	+0.047	-0.29	+0.053

In the AS topologies, Scalpel is able to reduce up to 96% of the monitors compared with MMP. The table also shows that Scalpel needs similar execution time in various networks compared with MMP.

Fig. 10(a) shows the number of monitors assigned in the three random topologies (i.e., BA, ER, RG) when there are different number of interesting links. As expected, fewer monitors are assigned when there are fewer interesting links. Although these random graphs have similar number of vertices (i.e., 300, 297, 296), they require different number of monitors. The main reason is that the BA graph has fewer links than the other two random graphs. Since a dense network can enable more measurement paths to identify interesting links, it usually requires fewer monitors than a sparse network. Fig. 10(b) shows the number of monitors assigned in the four topologies generated by the *dK*-graph tool [25]. When there are more interesting links in the network, more monitors are required to be assigned.

We further evaluate the monitor assignment performance of Scalpel under different settings of interesting link location and network topology type. Fig. 11(a) shows the results when the interesting links are edge links, core links and randomly chosen links. We can see that when the interesting links are edge links, more monitors are assigned to identify them. The reason is that fewer measurement paths can be used to identify edge links compared with core links. Fig. 11(b) shows the results under different network types. In these experiments, 10% of links are interesting links. We can see that the bus topology and the ring topology need more monitors. The mesh topology needs the least monitors. The reason is that there are more links in the mesh topology. Since more links can enable more measurement paths among monitors, the mesh topology requires fewer monitors.

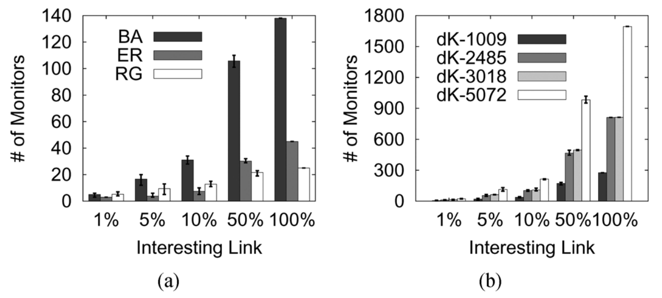


Fig. 10. Monitor assignment results of synthetic topologies. (a) Random topologies. (b) *dK*-graph topologies.

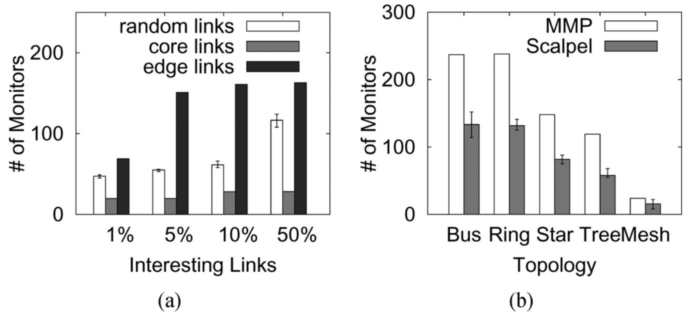


Fig. 11. Monitor assignment results under different network settings. (a) Location of interesting links. (b) Network topology type.

TABLE III
IDENTIFIABILITY IN CASE OF TOPOLOGY CHANGE

Modified	AS1221	AS1755	AS2914	AS7018	AS3257	AS3967
1%	96.6%	100.0%	99.8%	100.0%	99.2%	99.2%
5%	95.7%	100.0%	99.6%	100.0%	100.0%	97.7%
10%	95.3%	100.0%	99.2%	99.8%	100.0%	97.0%

In order to evaluate the robustness of Scalpel, we also conduct the following experiments. We use Scalpel to assign monitors and then modify the topology by changing a number of links (i.e., removing some links and adding the same number of other links). Then, in the modified network, we use the percentage of interesting links which can be identified by the monitors assigned as an evaluation metric. Table III shows the results. When 1% to 10% of links are modified, 95.3% to 100% of the interesting links can still be identified by the monitors assigned by Scalpel.

VII. RELATED WORK

Knowledge of the internal state of a network (e.g., link delays) is essential for network monitoring, fault diagnosis, load balancing and other network operations. In order to measure the network metrics, different approaches have been proposed in the literature. The first category includes *hop-by-hop* approaches, which use diagnostic tools such as *traceroute*, *pathchar* [26], and *Network Characterization Service (NCS)* [27] to measure hop-by-hop link metrics directly. By sending multiple probes with different time-to-live (TTL) fields, *traceroute* can measure the delay of each hop on the probed path. *Pathchar* uses a similar approach to measuring hop-by-hop delays, capacities and loss rates. *NCS* also reports available capacities of each hop. These approaches require that the Internet Control Message Protocol (ICMP) be supported at all nodes. Further, the above tools send a relatively large number of probes, introducing non-negligible overhead.

The other category includes end-to-end approaches, which use end-to-end metrics to infer internal link metrics [2]–[4], [28]–[31], also known as network tomography.

A key problem is how to assign the minimum number of monitors so that the operational cost can be reduced. The basic idea is to build a linear system from the path measurements and use linear algebraic techniques to calculate the unknown link metrics [32], [33]. These approaches utilize path information to calculate link metrics, reducing the number of probes significantly. Some approaches use uncontrollable probes to obtain path measurements. In [4], [28], the problem of minimum monitor assignment under uncontrollable routing is proved to be NP-hard. Other approaches assume the network is controllable, i.e., a monitor is able to send measurement packets with pre-determined paths. This controllable network assumption is reasonable since it is generally supported for networks under common administration [2]. As shown in [32], the problem is challenging due to the existence of linearly dependent paths. Several approaches [2], [33], [34] have been used to calculate the link metrics. When the link metrics are binary variables (e.g., normal or failed), Chen *et al.* give a topology requirement to identify all failed links using only one monitor measuring cycles. Many approaches focus on the usual case when the link metrics are arbitrary valued. When cyclic measurement paths are allowed, Gopalan *et al.* [5] give the necessary and sufficient conditions on the network topology. Since routing along cycles is typically prohibited in real networks, cycle-free measurement path are preferred. Ma *et al.* [2] give the necessary and sufficient conditions on the network topology when only cycle-free measurement paths are used. In order to identify all links in a connected network, Ma *et al.* also propose an efficient algorithm called MMP [2] to assign the minimum number of monitors as well as an efficient path construction algorithm [20]. Different with [2], we consider a more practical and general case when we are only interested in a subset of links. Assigning monitors to identify interesting links faces non-trivial challenges due to the complex dependency between the link identifiability and each monitor. A recent work [31] proposes a robust network tomography technique to deal with node failures. In the preferential link tomography problem, this robustness is also worth studying and we consider it as future work.

VIII. CONCLUSION

In this paper, we propose Scalpel, an efficient preferential link tomography approach. We theoretically prove that the graph trimming algorithm in Scalpel is minimal and the obtained assignment by Scalpel is able to identify all interesting links in the original graph. Extensive simulations based on both synthetic topologies and real network topologies show the effectiveness of Scalpel. Compared with state-of-the-art, our approach reduces the number of monitors by 39.0% to 98.6% when 50% to 1% of all links are interesting links.

There are multiple dimensions to explore in the future work. First, we would like to investigate how to design an optimal monitor assignment algorithm so that the number of assigned monitors can be minimized. It is possible since our graph trimming algorithm guarantees that an optimal monitor assignment in the graph after trimming is also an optimal monitor assignment in the original graph. Second, we would like to investigate how to select measurement paths to facilitate the identification of link metrics.

ACKNOWLEDGMENT

The authors would like to thank the editors and the anonymous reviewers for their insightful comments.

REFERENCES

[1] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *J. Amer. Stat. Assoc.*, vol. 91, no. 433, pp. 365–377, 1996.
[2] L. Ma, T. He, K. Leung, A. Swami, and D. Towsley, "Inferring link metrics from end-to-end path measurements: Identifiability and monitor placement," *IEEE/ACM Trans. Netw.*, vol. 22, no. 4, pp. 1351–1368, Aug. 2014.
[3] E. Lawrence, G. Michailidis, V. N. Nair, and B. Xi, "Network tomography: A review and recent developments," in *Frontiers in Statistics*, 2006, pp. 345–364.
[4] R. Kumar and J. Kaur, "Practical beacon placement for link monitoring using network tomography," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 12, pp. 2196–2209, Dec. 2006.

[5] A. Gopalan and S. Ramasubramanian, "On identifying additive link metrics using linearly independent cycles and paths," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 906–916, Jun. 2012.
[6] N. T. Spring, R. Mahajan, D. Wetherall, and T. E. Anderson, "Measuring ISP topologies with Rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, Feb. 2004.
[7] L. Mo, Y. He, Y. Liu, J. Zhao, S. Tang, X. Li, and G. Dai, "Canopy closure estimates with GreenOrbs: Sustainable sensing in the forest," in *Proc. ACM SenSys*, 2009, pp. 99–112.
[8] X. Mao, X. Miao, Y. He, T. Zhu, J. Wang, W. Dong, X. Yang Li, and Y. Liu, "Citysee: Urban CO2 monitoring with sensors," in *Proc. IEEE INFOCOM*, 2012, pp. 1611–1619.
[9] W. Dong, Y. Liu, Y. He, and T. Zhu, "Measurement and analysis on the packet delivery performance in a large scale sensor network," in *Proc. IEEE INFOCOM*, 2013, pp. 2679–2687.
[10] R. E. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, 1972.
[11] J. E. Hopcroft and R. E. Tarjan, "Dividing a graph into triconnected components," *SIAM J. Comput.*, vol. 2, no. 3, pp. 135–158, 1973.
[12] Y. Gao, W. Wu, W. Dong, C. Chen, X.-Y. Li, and J. Bu, "Preferential link tomography: Monitor assignment for inferring interesting link metrics," in *Proc. IEEE ICNP*, 2014, pp. 167–178.
[13] *RCF0791 (Internet Protocol)*, [Online]. Available: https://tools.ietf.org/html/rfc791
[14] M. Soliman, B. Nandy, I. Lambadaris, and P. Ashwood-Smith, "Source routed forwarding with software defined control, considerations and implications," in *Proc. CoNEXT*, 2012, pp. 43–44.
[15] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. ACM SOSP*, 2001, pp. 131–145.
[16] X. Yang and D. Wetherall, "Source selectable path diversity via routing deflections," in *Proc ACM SIGCOMM*, 2006, pp. 159–170.
[17] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala, "Path splicing," in *Proc. ACM SIGCOMM*, 2008, pp. 27–38.
[18] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, and A. Donnelly, "Symbiotic routing in future data centers," in *Proc ACM SIGCOMM*, 2010, pp. 51–62.
[19] L. Ma, T. He, K. K. Leung, A. Swami, and D. Towsley, "Monitor placement for maximal identifiability in network tomography," in *Proc. IEEE INFOCOM*, 2014, pp. 1447–1455.
[20] L. Ma, T. He, K. K. Leung, D. Towsley, and A. Swami, "Efficient identification of additive link metrics via network tomography," in *Proc. IEEE ICDCS*, 2013, pp. 581–590.
[21] Y. Gao, W. Dong, W. Wu, C. Chen, X.-Y. Li, and J. Bu, "Scalpel: Scalable preferential link tomography based on graph trimming [techRep]," [Online]. Available: http://www.emnets.org/pub/gaoyi/scalpel-techrep.pdf
[22] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Phys.*, vol. 74, no. 1, p. 47, 2002.
[23] P. Erdös and A. Rényi, "On random graphs I," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.
[24] M. Penrose*, Random Geometric Graphs*. : Oxford University Press Oxford, 2003, vol. 5.
[25] P. Mahadevan, C. Hubble, D. V. Krioukov, B. Huffaker, and A. Vahdat, "Orbis: Rescaling degree correlations to generate annotated Internet topologies," in *Proc. ACM SIGCOMM*, 2007, pp. 325–336.
[26] A. B. Downey, "Using pathchar to estimate Internet link characteristics," in *Proc. ACM SIGCOMM*, 1999, pp. 241–250.
[27] G. Jin, G. Yang, B. R. Crowley, and D. A. Agarwal, "Network characterization service (NCS)," in *Proc. IEEE HPDC*, 2001, pp. 289–299.
[28] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in IP networks," *IEEE/ACM Trans. Netw.*, vol. 14, no. 5, pp. 1092–1103, Oct. 2006.
[29] J. D. Horton and A. Lopez-Ortiz, "On the number of distributed measurement points for network tomography," in *Proc. ACM IMC*, 2003, pp. 204–209.
[30] H. H. Song, L. Qiu, and Y. Zhang, "Netquest: A flexible framework for large-scale network measurement," *IEEE/ACM Trans. Netw.*, vol. 17, no. 1, pp. 106–119, Feb. 2009.
[31] S. Tati, S. Silvestri, T. He, and T. L. Porta, "Robust network tomography in the presence of failures," in *Proc. IEEE ICDCS*, 2014, pp. 481–492.
[32] Y. Chen, D. Bindel, H. Song, and R. H. Katz, "An algebraic approach to practical and scalable overlay network monitoring," in *Proc. ACM SIGCOMM*, 2004, pp. 55–66.
[33] O. Gurewitz and M. Sidi, "Estimating one-way delays from cyclic-path delay measurements," in *Proc. IEEE INFOCOM*, 2001, pp. 1038–1044.
[34] S. Ahuja, S. Ramasubramanian, and M. Krunz, "SRLG failure localization in all-optical networks using monitoring cycles and paths," in *Proc. IEEE INFOCOM*, 2008, pp. 181–185.

Yi Gao (M'15) received the B.S. and Ph.D. degrees from Zhejiang University, Zhejiang, China, in 2009 and 2014, respectively.

He is currently a Research Assistant Professor with Zhejiang University, Zhejiang, China. From December 2008 to April 2009, he was with the Information System College of Singapore Management University as an exchange student. From October 2011 to October 2012, he was with McGill University as a Joint Training Research Student. His research interests include protocols design and measurement in networks.

Wei Dong (S'08–M'14) received the B.S. and Ph.D. degrees from the College of Computer Science at Zhejiang University, Zhejiang, China, in 2005 and 2010, respectively.

He was a Postdoctoral Fellow with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, from December 2010 to December 2011. He is currently an Associate Professor with the College of Computer Science in Zhejiang University, Zhejiang, China. His research interests include networked embedded systems and wireless sensor networks.

Wenbin Wu received the B.S. degree from Zhejiang University of Technology, Zhejiang, China, where he is currently working toward the M.S. degree in computer science.

His research interests include fine-grained measurement in wireless sensor networks and network tomography techniques in modern communication networks.

Chun Chen (M'08) received the B.S. degree in mathematics from Xiamen University, China, in 1981, and the M.S. and Ph.D. degrees in computer science from Zhejiang University, Zhejiang, China, in 1984 and 1990, respectively.

He is a Professor with the College of Computer Science and the Director of Institute of Computer Software at Zhejiang University. His research interests include embedded system, image processing, computer vision, and CAD/CAM.

Xiang-Yang Li (M'00–SM'08–F'15) received the B.S. degrees in computer science and business management from Tsinghua University, Beijing, China, both in 1995, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2000 and 2001, respectively.

He is a Professor with the Illinois Institute of Technology, Chicago, IL, USA. He published a monograph *Wireless Ad Hoc and Sensor Networks: Theory and Applications* (Cambridge University, 2008). His research interests include wireless networking, mobile computing, security and privacy, cyber physical systems, smart grid, social networking, and algorithms.

Prof. Li is an ACM Distinguished Scientist. He holds EMC-Endowed Visiting Chair Professorship at Tsinghua University. He was a recipient of the China NSF Outstanding Overseas Young Researcher (B). He and his students won four Best Paper Awards and one Best Demo Award and was nominated for Best Paper Awards twice (ACM MobiCom 2008 and ACM MobiCom 2005).

Jiajun Bu (M'06) received the B.S. and Ph.D. degrees in computer science from Zhejiang University, Zhejiang, China, in 1995 and 2000, respectively.

He is a Professor with the College of Computer Science and the Deputy Director of the Institute of Computer Software, Zhejiang University, Zhejiang, China. His research interests include embedded system, mobile multimedia, and data mining.

Prof. Bu is a member of the ACM.