This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Minimizing Probing Cost and Achieving Identifiability in Probe Based Network Link Monitoring

Qiang Zheng, *Student Member, IEEE,* and Guohong Cao, *Fellow, IEEE*
Department of Computer Science and Engineering
The Pennsylvania State University
E-mail: {quz103, gcao}@cse.psu.edu

Abstract—Continuously monitoring link performance is important to network diagnosis. In this paper, we address the problem of minimizing the probing cost and achieving identifiability in probe based network link monitoring. Given a set of links to monitor, our objective is to select the minimum number of probing paths that can uniquely determine all identifiable links and cover all unidentifiable links. We propose an algorithm based on a linear system model to find out all irreducible sets of probing paths that can uniquely determine an identifiable link, and we extend the bipartite model to reflect the relationship between a set of probing paths and an identifiable link. Since our optimization problem is NP-hard, we propose a heuristic based algorithm to greedily select probing paths. Our method eliminates two types of redundant probing paths, i.e., those that can be replaced by others and those without any contribution to achieving identifiability. Simulations based on real network topologies show that our approach can achieve identifiability with very low probing cost. Compared with prior work, our method is more general and has better performance.

Index Terms—Internet, network monitoring, end-to-end probe, probing cost, identifiable link.

1 INTRODUCTION

With rapid growth of the Internet, efficiently monitoring the performance and robustness of networks becomes more and more important. Service providers need to keep track of their networks to ensure that their services satisfy the commitments specified in the service level agreements (SLAs). Additionally, applications sensitive to network performance, e.g., online trading, Voice over IP and, IPTV, need to know the network status such as link delay, jitter, and loss rate. These demands motivate recent research on network monitoring and performance inference.

Due to various advantages, tomography-based end-to-end probe has received much attention and has been widely used in network monitoring [1]–[11]. In this approach, some end systems are connected to the network as shown in Fig. 1. An end system sends probing packets to another end system to measure the delay or loss rate of the routing path. Unlike Simple Network Management Protocol (SNMP) based polling [12], end-to-end probe does not need to run agents on routers. In particular, it is suitable for monitoring the performance of network links that belong to a non-cooperative administrative domain, in which directly measuring link performance is usually hard to achieve [13]. Furthermore, compared with reply based probe such as `ping` and `traceroute`, end-to-end probe uses normal data packets, and thus it does not have the problem of being ignored by intermediate routers [14] or blocked by firewalls [15].

Selecting probing paths is the major problem of probe based network link monitoring. Generally, there are two important considerations, i.e., *minimizing probing cost* and *achieving identifiability*. The probing cost is mainly defined as the number of selected probing paths [3], [16]–[20]. It can also be the number of end systems used for probing [9], [16], [17] and the cost specified by network components [3], [7]. Since probing traffic is periodically injected into the network, it consumes network bandwidth and increases workload of the routers. Lower probing cost means less resource consumption, less negative impact on the normal data transmission, and better scalability of the probing scheme. Therefore, many existing works [3], [7], [9], [16]–[18] focus on minimizing the probing cost in network link monitoring.

However, most prior works neglect identifiability and the selected probing paths cannot uniquely infer the performance of the network links. Since a probing path may consist of multiple links, a probe measures the performance of the whole probing path, and it cannot infer the performance of a specific link. Hence, selecting probing paths to cover a link may not be sufficient for monitoring the performance of the link. To uniquely infer the performance of a link, multiple coordinated probes are needed. Unfortunately, in a general network the performance of some links may not be able to be uniquely inferred from probes [21].

Network links can be classified into two types. If the performance of a link can be uniquely inferred by a set of probes, this link is *identifiable* and the set of probes can *uniquely determine* it. Otherwise, the link is *unidentifiable*. If probing paths are not properly selected, the performance of a link cannot be uniquely inferred, even if it is an identifiable link. As shown in Fig. 1, links e_1, e_2, and e_3 are identifiable, but links e_4 and e_5 are unidentifiable because every probe traversing e_4 also traverses e_5. Using probing paths p_1, p_2, and p_4 together can uniquely determine e_1, e_2, and e_3. However, only choosing p_1 and p_2 cannot uniquely determine any link.

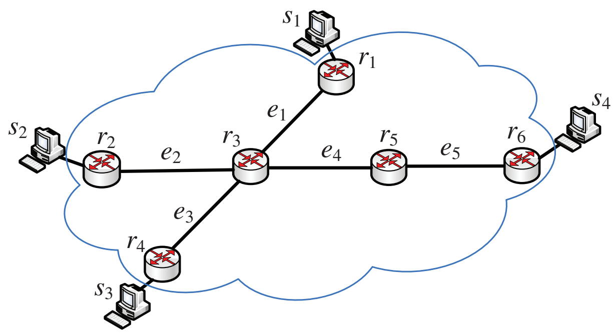


Fig. 1. An example of network link monitoring with six probing paths $p_1(s_1, s_2)$, $p_2(s_1, s_3)$, $p_3(s_1, s_4)$, $p_4(s_2, s_3)$, $p_5(s_2, s_4)$, and $p_6(s_3, s_4)$, where $p_i(s_j, s_k)$ is the probing path between end systems s_j and s_k.

In this paper, we address the problem of minimizing the probing cost and achieving identifiability in probe based network link monitoring. Given a set of links called *target links*[1] to monitor, our objective is to select the minimum number of probing paths that can uniquely determine all identifiable target links and cover all unidentifiable target links. There are three major challenges. First, a target link may be unidentifiable. Using multiple probing paths to monitor an unidentifiable link only wastes network bandwidth. Hence, it is necessary to differentiate an identifiable link from an unidentifiable link. Second, different probing paths have different contributions to achieving our objective, and we need to determine the most useful probing paths. Third, a probing path may be replaced by other probing paths [1], [18], and thus we should avoid selecting redundant probing paths so as to minimize the probing cost.

The basic idea of our approach is to only select probing paths that are the most useful for achieving our objective. More specifically, the proposed approach consists of three parts. First, we adopt a linear system to model the relationship between probing paths and links, and then find out identifiable target links by solving the linear system. Second, we design a method based on matrix decomposition and linear replacement to calculate irreducible sets of probing paths which can uniquely determine each identifiable target link. Third, we extend the bipartite model, which is commonly used for modeling the relationship between a single probing path and a link, to reflect the relationship between a set of probing paths and a link. With this model, we prove that our optimization problem is NP-hard, and then propose a heuristic based algorithm to greedily select probing paths. Simulations based on real ISP networks show that our approach can achieve identifiability with very low probing cost. Compared with prior work, our method is more general and has better performance.

1. Target links can be links at critical topological location, and their performance usually affects a large area of the network [22].

The rest of this paper is organized as follows. We present the linear system model and the problem description in Section 2. Our approach is introduced in Section 3, which includes the algorithm for calculating all irreducible sets of probing paths to uniquely determine an identifiable target link, the extended bipartite model, and the heuristic based algorithm for selecting probing paths. Section 4 presents the performance evaluation, and Section 5 reviews related work. Finally, Section 6 concludes the paper.

2 Preliminaries

In this section, we first introduce the system model and define our problem, and then describe the linear system model used in our approach.

2.1 System Model and Problem Description

Similar to prior works on network monitoring [3], [16], [23]–[27], we model the network as a connected undirected graph $G(V, E)$, where V is the set of nodes (routers) and $E = \{e_i | 1 \le i \le m\}$ is the set of edges (communication links between routers). In the rest of this paper, we use edge and link interchangeably. Note that the proposed technique also works for asymmetric links, where the network is modeled as a directed graph.

Some routers in the network can be directly connected by end systems which can send and receive probing packets. The probing packet from end system s_j to end system s_k traverses the routing path from s_j to s_k. Such an end-to-end path is referred to as a *probing path*. A probing path can *cover* a link if it traverses this link. There are two types of links on each probing path. The first is the link between an end system and a router. In the Internet, end systems can only directly connect to edge routers and they are usually close to these routers, e.g., in the same building or campus. Thus, the links between end systems and edge routers are quite short. The second is the link between two routers, which is usually hundreds of miles long in the current Internet. Since the performance of the first type of links is usually quite stable, they are commonly omitted in network link monitoring. Hence, we only count the performance of the second type of links. For example, in Fig. 1 the probe from s_1 to s_3 measures the delay of the probing path p_2. We omit the delay of the link from s_1 to r_1 and the link from r_4 to s_3. Consequently, the measured delay of probing path p_2 is caused by links e_1 and e_3.

In the network, there are n probing paths $P = \{p_i | 1 \le i \le n\}$ which can be used for monitoring a set of m_t target links $E_t \subseteq E$. For simplicity, target links are labeled as $e_1, \cdots, e_{m_t}$. We assume that the network topology is available, which is a widely used assumption in probe based network monitoring. Based on the network topology, we know which links can be covered by a probing path. Network monitoring is very important for Internet Service Providers (ISPs), because they need to keep track of the performance of their networks. ISPs know the complete topology of their networks, and thus this approach has been used in practice.

Probe based network link monitoring has two steps. The first is to select a set of probing paths. Then, end systems on the chosen probing paths periodically issue probing packets and send measurement results to the central Network Operations Center (NOC). In this paper, we focus on the first step; i.e., selecting probing paths. We define the problem of minimizing the probing cost and achieving identifiability as follows, where the probing cost is defined as the number of probing paths selected for network monitoring.

Definition 1 (Problem definition): Given a network G, a set of probing paths P, and a set of target links E_t, the objective is to select the minimum number of probing paths from P, such that all identifiable target links can be uniquely determined and all unidentifiable target links are covered.

Minimizing probing cost is an important objective in existing approaches of probe based network monitoring. Lower probing cost means less resource consumption, less negative impact on the normal data transmission, and better scalability. In most prior works [3], [16]–[20], the probing cost is defined as the number of selected probing paths. Therefore, we aim at choosing a minimum set of probing paths for link monitoring.

The goal of link monitoring is to detect and localize the target links with abnormal performance. Hence, it tries to infer the performance of each target link from probing results as accurately as possible. For identifiable target links, the performance can be uniquely inferred from probes. Hence, we intend to select probing paths to uniquely determine them. If any of them has abnormal performance, we can accurately identify the link. For unidentifiable target links, the performance cannot be uniquely inferred from probes. We can estimate their performance with techniques like second-order statistics [8], [10]. Although the estimated performance may not be accurate, it is still useful for link performance diagnosis. To use the second-order statistics technique, each unidentifiable target link should be covered by at least one probing path. Hence, we also select probing paths to cover all unidentifiable target links.

2.2 Linear System Model

For link performance satisfying the additive metric, the relationship between probing paths and links can be naturally modeled as a linear system $LS = \{ls_i | 1 \le i \le n\}$, in which ls_i is the ith linear equation as shown in Eq. (1). Binary variable a_{ij} is 1 if probing path p_i covers link e_j; otherwise it is 0. Variables x_j and b_i represent the performance of link e_j and probing path p_i. Thus, Eq. (1) means that the performance of p_i is the addition of the performance of all links on p_i.

$$\sum_{j=0}^{m} a_{ij} x_j = b_i \tag{1}$$

The linear system can be written in the matrix computation form as shown in Eq. (2). Variable $\mathbf{x}$ is the vector form of $x_1, \cdots, x_m$ and $\mathbf{b}$ is the vector form of $b_1, \cdots, b_n$. The coefficient matrix $A = (\mathbf{a}_1, \cdots, \mathbf{a}_n)^T$ is also called the *dependency matrix* [17].

$$A\mathbf{x} = \mathbf{b} \tag{2}$$

$$\begin{cases} x_1 + x_2 = b_1 \\ x_1 + x_3 = b_2 \\ x_1 + x_4 + x_5 = b_3 \\ x_2 + x_3 = b_4 \\ x_2 + x_4 + x_5 = b_5 \\ x_3 + x_4 + x_5 = b_6 \end{cases}$$

(a) The linear system model LS

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

(b) The dependency matrix

Fig. 2. The linear system model and the dependency matrix of the example in Fig. 1.

The linear system model is suitable for delay and loss rate which are major link performance metrics in network link monitoring. We use delay as an example to explain the meaning of the linear equations. The network in Fig. 1 has six probing paths and five links. Hence, the linear system has six equations and five variables $x_1, \cdots, x_5$ as shown in Fig. 2(a). The dependency matrix is shown in Fig. 2(b). For probing path p_1 between end systems s_1 and s_2, the overall delay of this path is the sum of the delay on links e_1 and e_2. Therefore, in the first equation in Fig. 2(a), b_1 is equal to the sum of x_1 and x_2.

For loss rate, we need to make a transformation with logarithm. Suppose r_j is the loss rate of link e_j and q_i is the loss rate of link e_j. The loss rate satisfies $\prod_{j=0}^{m}(1 - a_{ij}r_j) = 1 - q_i$. By applying logarithm to both sides, we have $\lg \prod_{j=0}^{m}(1 - a_{ij}r_j) = \lg(1 - q_i)$, which results in $\sum_{j=0}^{m} \lg(1 - a_{ij}r_j) = \lg(1 - q_i)$. There are two cases for the value of $1 - a_{ij}r_j$. If $a_{ij} = 1$, i.e., probing path p_i covers link e_j, $1 - a_{ij}r_j = 1 - r_j$. If $a_{ij} = 0$, we have $1 - a_{ij}r_j = 1$, and thus $\lg(1 - a_{ij}r_j) = 0$. For each probing path p_i, we define $b_i = \lg(1 - q_i)$. Similarly, for each link e_j, we let $x_j = \lg(1 - r_j)$ if e_j is on p_i, or $x_j = 0$ otherwise. Through this transformation, we can apply the linear equation in Eq. (1) to loss rate. For example, the loss rate of p_1 in Fig. 1 is $q_1 = 1 - (1 - r_1)(1 - r_2)$. With the transformation, we have $\lg(1 - r_1) + \lg(1 - r_2) = \lg(1 - q_1)$. By defining $x_1 = \lg(1 - r_1)$, $x_2 = \lg(1 - r_2)$, and $b_1 = \lg(1 - q_1)$, this equation becomes $x_1 + x_2 = b_1$ which is the first equation in Fig. 2(a).

The above transformation is not suitable for the probing paths that traverse failed links. Suppose the probing path p_i that traverses link e_j and e_j fails. The loss rates of p_i and e_j will both be 1, and thus $\lg(1 - q_i)$ and $\lg(1 - r_j)$ are not defined. To solve this problem, we simply ignore the probing result of p_i, if the measured loss rate of p_i is 1. It equals to removing a linear equation from the linear system. In practice, probing paths are very unlikely to traverse failed links due to two reasons. First, the failure probability of links in the Internet is very low and most link failures are transient (about 50% of them last for less than 1 minute and 90% of them are shorter than 10 minutes [28]). Hence, probing paths are free of link failures most of time. Second, when long-lasting link failures occur, the routing protocol can quickly converge to a new network topology, and all routing paths can bypass the failed links. In this case, we need to select probing paths based on the new network topology.

We can use the linear system model to determine if a link is identifiable and if a probing path can be replaced by other probing paths. First, a link e_i is identifiable if and only if the corresponding variable x_i in the linear system is solvable. Hence, we can determine all identifiable links by solving the linear system. Second, a probing path p_i can be replaced by a set of probing paths, if row vector $\mathbf{a}_i$ in the dependency matrix can be linearly expressed by the row vectors corresponding to this set of probing paths. In Fig. 2(b), row vector $\mathbf{a}_5$ can be linearly expressed by row vectors $\mathbf{a}_2$, $\mathbf{a}_3$, and $\mathbf{a}_4$ as $\mathbf{a}_5 = -\mathbf{a}_2 + \mathbf{a}_3 + \mathbf{a}_4$. It means that probing paths p_2, p_3, and p_4 together can replace probing path p_5.

Due to linear dependence among row vectors, there may be many sets of linear equations for a solvable variable. Accordingly, an identifiable link may be uniquely determined by multiple sets of probing paths. We define the solution to an identifiable link as follows.

Definition 2 (Solution to an identifiable link): A *solution* to an identifiable link e_i is an irreducible set of probing paths that can uniquely determine e_i.

Irreducible means that each probing path in the set cannot be replaced by other probing paths in the same set. This definition is consistent with our objective of minimizing the number of selected probing paths. Consider a set of probing paths that can uniquely determine link e_i. If a probing path can be replaced by others in the same set, the set can still uniquely determine e_i after removing this probing path. For uniquely determining e_i, containing such replaceable probing paths only increases the probing cost. Therefore, we require that a solution does not contain redundant probing paths. For example, x_1 in Fig. 2(a) is a solvable variable and $\frac{1}{2}b_1 + \frac{1}{2}b_2 - \frac{1}{2}b_4$ is a solution to x_1. It contains three variables b_1, b_2, and b_4, and thus probing paths p_1, p_2, and p_4 together can uniquely determine link e_1. Since row vectors $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_4$ are linearly independent, each probing path is not replaceable by other two probing paths. Therefore, set $\{p_1, p_2, p_4\}$ is a solution to e_1.

Note that a solution to a solvable variable x_i in the linear system does not necessarily match a solution to an identifiable link e_i. A solution to x_i is a linear combination of variable b_js. The corresponding probing paths may contain replaceable ones. For example, the linear expression $b_1 - \frac{1}{2}b_2 + \frac{1}{2}b_3 - b_5 + \frac{1}{2}b_6$ is a solution to x_1 in Fig. 2(a). However, the set of probing paths $\{p_1, p_2, p_3, p_5, p_6\}$ is not a solution to link e_1, because the corresponding five row vectors are not linearly independent. Table 1 summarizes the symbols used in the linear system model and the symbols that will be used in the path selection algorithm in the next section.

TABLE 1
Table of notations

Symbols	Meaning
G	network under study
m	the number of links in G
e_i	the ith link in G
n	the number of probing paths in G
p_i	the ith probing path in G
E_t	set of target links
m_t	the number of target links
LS	linear system model of G
ls_i	the ith linear equation of LS
b_i	measured performance of the ith probing path
A	dependency matrix
S^i	set of all solutions to the solvable variable x_i in LS
S^i_j	the jth solution to the solvable variable x_i in LS
L	set of linear expressions, $L = \{l_i \| 1 \leq i \leq t\}$
l_i	linear expression for the ith row vector in A_N

3 Path Selection Algorithm

This section introduces our algorithm for selecting probing paths.

3.1 Overview

The basic idea of our algorithm is to determine the contribution of a probing path for achieving our objective, and then choose the most useful probing paths and avoid selecting redundant ones. Probing paths have different contributions to uniquely determine identifiable links and cover unidentifiable links. An identifiable link may have many solutions, and a probing path may be in the solutions to multiple identifiable links. Similarly, an unidentifiable link may be covered by several probing paths, and a probing path can cover multiple links.

The linear system model proposed in Section 2.2 can reflect the contribution of a probing path to covering unidentifiable links. For identifiable links, a natural method is to find out all solutions to each identifiable link, and thus we can know which probing paths are the most useful to achieving identifiability. We propose a method based on matrix decomposition and linear replacement to calculate all solutions to an identifiable link in Section 3.2 and Section 3.3. Moreover, we develop an extended bipartite model to reflect the relationship between probing paths and target links in Section 3.4. Through this model, we introduce a heuristic based algorithm in Section 3.5 which can efficiently select probing paths to achieve our objective.

3.2 Decomposition of Linear System

The first step of our approach is to determine if a link is identifiable or not. Solving the linear system is the simplest way to achieve this. Many techniques in linear algebra, such as Gaussian elimination, can solve the linear system. A key observation is that the linear dependence between row vectors of the dependency matrix only affects how many solutions an identifiable link has, but does not affect if a link is identifiable or not. Therefore, we decompose the dependency matrix, based on which we can determine all identifiable target links and calculate one solution for each of them. Then, we use linear replacement to calculate all solutions, which will be introduced in the next subsection.

To decompose the dependency matrix, we divide its row vectors into two groups as shown in Eq. (3). A_R is a maximal independent set of row vectors of matrix A. Suppose A_R contains r row vectors. Matrix A_N contains the other $n-r$ row vectors. According to linear algebra theory, each row vector of A_N can be linearly expressed by row vectors of A_R. For simplicity, we renumber the probing paths such that $A_R = (\mathbf{a}_1, \cdots, \mathbf{a}_r)^T$ and $A_N = (\mathbf{a}_{r+1}, \cdots, \mathbf{a}_n)^T$.

$$A = \begin{pmatrix} A_R \\ A_N \end{pmatrix} \tag{3}$$

We use the algorithm introduced in [18] to decompose the dependency matrix, which is based on standard rank-revealing decomposition techniques [29]. A dependency matrix may have multiple decompositions. Our algorithm for calculating all solutions for each identifiable target link does not have any requirement on the matrix decomposition. Therefore, we can use any tie breaking strategy to choose a decomposition. Based on the matrix decomposition, the linear system in Eq. (2) is also decomposed into two parts as shown in Eq. (4), in which vector $\mathbf{b}$ is partitioned into two vectors $\mathbf{b}_R = (b_1, \cdots, b_r)^T$ and $\mathbf{b}_N = (b_{r+1}, \cdots, b_n)^T$.

$$\begin{pmatrix} A_R \\ A_N \end{pmatrix} \mathbf{x} = \begin{pmatrix} \mathbf{b}_R \\ \mathbf{b}_N \end{pmatrix} \tag{4}$$

Since each row vector of A_N is a linear combination of row vectors of A_R, a solvable/unsovlable variable in linear system $A_R\mathbf{x} = \mathbf{b}_R$ is also solvable/unsolvable in linear system $A\mathbf{x} = \mathbf{b}$. By solving linear system $A_R\mathbf{x} = \mathbf{b}_R$, we can determine whether a target link is identifiable or not. Also, we calculate a solution S_1^i to each solvable variable x_i, which is referred to as the *base solution*. Solving linear system $A_R\mathbf{x} = \mathbf{b}_R$ can obtain only one base solution to each solvable variable x_i, because the row vectors of A_R are linearly independent.

Next, we compute the relationship between variable b_is. Each row vector of A_N can be linearly expressed by row vectors of A_R. By solving the linear system in Eq. (5), we can calculate the linear expression for row vector $\mathbf{a}_{r+i}$ of A_N, where $i = 1, \cdots, n-r$.

$$\sum_{j=1}^{r} c_{ij}\mathbf{a}_j + \mathbf{a}_{r+i} = 0 \tag{5}$$

Multiplying vector $\mathbf{x}$ to both sides of Eq. (5) results in $\sum_{j=1}^{r} c_{ij}\mathbf{a}_j\mathbf{x} + \mathbf{a}_{r+i}\mathbf{x} = 0$. Since $\mathbf{a}_j\mathbf{x} = b_j$ for $1 \le j \le r$ and $\mathbf{a}_{r+i}\mathbf{x} = b_{r+i}$, we have the linear expression in Eq. (6). Through it, we obtain the relationship between b_is, which will be used for calculating all solutions to identifiable target links. We use $L = \{l_i | 1 \le i \le n-r\}$ to represent the set of these linear expressions, where l_i is the expression containing b_{r+i}.

$$\sum_{j=1}^{r} c_{ij}b_j + b_{r+i} = 0 \tag{6}$$

The dependency matrix in Fig. 2(b) shows an example. A decomposition of this matrix is $A_R = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4)^T$ and $A_N = (\mathbf{a}_5, \mathbf{a}_6)^T$. The corresponding partition of vector $\mathbf{b}$ is $\mathbf{b}_R = (b_1, b_2, b_3, b_4)^T$ and $\mathbf{b}_N = (b_5, b_6)^T$. By solving linear system $A_R\mathbf{x} = \mathbf{b}_R$, we discover that links e_1, e_2, and e_3 are identifiable but links e_4 and e_5 are unidentifiable. The basic solution to x_1 is $\frac{1}{2}b_1 + \frac{1}{2}b_2 - \frac{1}{2}b_4$, which means that set $\{p_1, p_2, p_4\}$ is a solution to link e_1. Moreover, set L contains two linear expressions, i.e., l_1 is $b_2 - b_3 - b_4 + b_5 = 0$ and l_2 is $b_1 - b_3 - b_4 + b_6 = 0$.

3.3 Solution Calculation

In this subsection, we propose an algorithm to calculate all solutions to a solvable variable, through which we can obtain all solutions to each identifiable target link. The basic idea of calculating all solutions to a solvable variable x_i is to replace the variables in solutions with the linear expressions in set L. Initially, we have only the base solution S_1^i obtained from linear system $A_R\mathbf{x} = \mathbf{b}_R$. For each linear expression $l_j \in L$, if it has a common variable b_k with S_1^i, replacing b_k in S_1^i with l_j results in a linear expression that can solve x_i. The variables in this linear expression correspond to a set of probing paths which can uniquely determine link e_i. We check if the set contains any replaceable probing paths. If not, this set of probing paths is a solution to e_i. Similar linear replacement is applied to the obtained solutions until no new solution can be found.

The algorithm for calculating all solutions to a solvable variable x_i is shown in Algorithm 1. The input is base solution S_1^i and set L of linear expressions. We use a queue to store the solutions. As shown in line 5–14, each time we take out a solution S_c^i from the queue and apply all possible linear replacements to it. A linear replacement generates a linear expression S_t^i (line 8). Then, in line 9, the algorithm checks if each variable in S_t^i is replaceable by other variables in S_t^i. If not, S_t^i corresponds to a solution to link e_i. We record S_t^i by putting it into queue Q, if it is not in queue Q and set S^i. After applying all linear replacements to the current solution S_c^i, we put it into set S^i. As a result, at any moment set S^i contains all solutions that have been applied linear replacements, and queue Q contains all solutions that will be applied linear replacements. When queue Q becomes empty, the algorithm stops and all solutions to x_i are in set S^i.

Algorithm 1 SolutionCalculation

Input: Base solution S_1^i, and set L of linear expressions
Output: Set S^i containing all solutions to x_i
Procedure:
1: INIT(Q) //Initialize a queue Q
2: $S^i \leftarrow \emptyset$
3: ENQUEUE(Q, S_1^i)
4: **while** $Q \neq \emptyset$ **do**
5: $S_c^i \leftarrow$ DEQUEUE(Q)
6: **for** each linear expression $l_j \in L$ **do**
7: **for** each common variable b_k in S_c^i and l_j **do**
8: $S_t^i \leftarrow$ replace b_k in S_c^i with l_j
9: **if** each variable in S_t^i cannot be replaced by other variables in S_t^i, and $S_t^i \notin Q \wedge S_t^i \notin S^i$ **then**
10: ENQUEUE(Q, S_t^i)
11: **end if**
12: **end for**
13: **end for**
14: $S^i \leftarrow S^i \cup S_c^i$
15: **end while**

Theorem 1: The algorithm `SolutionCalculation` can find all solutions to a solvable variable in the linear system.

Proof: We prove it by contradiction. For a solvable variable x_i, suppose it has a solution S_q^i which is not found by the algorithm. The missed solution S_q^i cannot be linearly expressed by base solution S_1^i and linear expressions in set L. Otherwise, the algorithm `SolutionCalculation` can find it with linear replacement. Since solutions to x_i are linear combinations of variables b_j, S_q^i is in the following form.

$$x_i = \sum_{j=1}^{n} d_{qj} b_j \tag{7}$$

The above equation together with base solution S_1^i and linear expressions in L form a linear system as shown in Eq. (8). The first equation is base solution S_1^i. Since it is computed from linear system $A_R\mathbf{x} = \mathbf{b}_R$, the value of x_i is a linear combination of b_js, where $1 \le j \le r$. The second equation is the same as Eq. (7). For the other $n-r$ equations, each of them is a linear expression in set L, which is shown in Eq. (6).

$$\begin{cases} \sum_{j=1}^{r} d_{1j} b_j - x_i = 0 \\ \sum_{j=1}^{n} d_{qj} b_j - x_i = 0 \\ \sum_{j=1}^{r} c_{1j} b_j + b_{r+1} = 0 \\ \vdots \\ \sum_{j=1}^{r} c_{n-r,j} b_j + b_n = 0 \end{cases} \tag{8}$$

In this linear system, $b_1, \cdots, b_n$ and x_i are unknown variables. Hence, the coefficient matrix is shown in Eq. (9).

$$\begin{array}{c} \begin{array}{cccccc} b_1 & \cdots & b_r & b_{r+1} & \cdots & b_n & x_i \end{array} \\ \begin{pmatrix} d_{11} & \cdots & d_{1r} & 0 & \cdots & 0 & -1 \\ d_{q1} & \cdots & d_{qr} & d_{q,r+1} & \cdots & d_{qn} & -1 \\ c_{11} & \cdots & c_{1r} & 1 & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ c_{n-r,1} & \cdots & c_{n-r,r} & 0 & \cdots & 1 & 0 \end{pmatrix} \end{array} \tag{9}$$

we subtract $d_{q,r+k}$ times of the $(2+k)$th row from the second row, where $k = 1, \cdots, n-r$. For example, the second row is subtracted by $d_{q,r+1}$ times of the third row, $d_{q,r+2}$ times of the fourth row, and so on. The matrix turns into the form as shown in Eq. (10). Compared with the matrix in Eq. (9), the difference is only at the second row. For $j = 1, \cdots, r$, variable d'_{qj} is shown in Eq. (11).

$$\begin{array}{c} \begin{array}{cccccc} b_1 & \cdots & b_r & b_{r+1} & \cdots & b_n & x_i \end{array} \\ \begin{pmatrix} d_{11} & \cdots & d_{1r} & 0 & \cdots & 0 & -1 \\ d'_{q1} & \cdots & d'_{qr} & 0 & \cdots & 0 & -1 \\ c_{11} & \cdots & c_{1r} & 1 & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ c_{n-r,1} & \cdots & c_{n-r,r} & 0 & \cdots & 1 & 0 \end{pmatrix} \end{array} \tag{10}$$

$$d'_{qj} = d_{qj} - \sum_{k=1}^{n-r} c_{kj} d_{q,r+k} \tag{11}$$

Since row vectors in matrix A_R are linear independent, linear system $A_R\mathbf{x} = \mathbf{b}_R$ has only one solution to solvable variable x_i, i.e., base solution S_1^i. Therefore, the first two row vectors in Eq. (10) must be the same. Otherwise, there are two different linear combinations of $b_1, \cdots, b_n$ to express x_i, i.e., $A_R\mathbf{x} = \mathbf{b}_R$ has two different solutions to x_i. It shows that the second row vector of Eq. (9) can be linearly expressed by other row vectors. This contradicts with the assumption that S_q^i cannot be linearly expressed by S_1^i and linear expressions in set L. Therefore, no such missed solution S_q^i exists, which indicates that `SolutionCalculation` can find out all solutions to each solvable variable. □

A solution to variable x_i is a linear combinations of $b_1, \cdots, b_n$. Since b_j corresponds to probing path p_j, we can easily obtain the corresponding solution to identifiable link e_i. Based on Theorem 1, our algorithm can find all solutions to each identifiable link. The computational complexity of this algorithm is determined by the number of linear replacements, which may not be a linear function of the size of dependency matrix. For a large-scale network, an identifiable link may have a large number of solutions, and thus the algorithm needs to run quite long. Actually, the algorithm for selecting probing paths does not need to use all solutions. For an identifiable link, we can choose some of its solutions for probing path selection, and hence the algorithm only needs to calculate some solutions rather than all solutions. More details will be introduced in Section 3.5.

To make it more clear, we use an example to show how to calculate all solutions to identifiable link e_1 in Fig. 1. Initially, we have base solution $S_1^1 : \frac{1}{2}b_1 + \frac{1}{2}b_2 - \frac{1}{2}b_4$. Set L contains two linear equations $l_1 : b_2 - b_3 - b_4 + b_5 = 0$ and $l_2 : b_1 - b_3 - b_4 + b_6 = 0$. The base solution has two common variables b_2 and b_4 with linear expression l_1. Replacing b_2 in S_1^1 with $b_3 + b_4 - b_5$ results in $\frac{1}{2}b_1 + \frac{1}{2}b_3 - \frac{1}{2}b_5$, which is a new solution to x_1. We name it as S_2^1. Similarly, replacing b_4 in S_1^1 with $b_2 - b_3 + b_5$ produces $\frac{1}{2}b_1 + \frac{1}{2}b_3 - \frac{1}{2}b_5$. Then we use l_2 to replace b_1 and b_4 in S_1^1. Both of them result in $\frac{1}{2}b_2 + \frac{1}{2}b_3 - \frac{1}{2}b_6$, which is named as S_3^1. Until now, we have already applied all possible linear replacements to S_1^1. Next, we apply linear replacements to S_2^1 and S_3^1. The generated new solutions are put into the queue, and linear replacements continue until the queue is empty. When applying l_1 to solution $b_1 - \frac{1}{2}b_4 - \frac{1}{2}b_5 + \frac{1}{2}b_6$ to replace b_4, we get linear expression $b_1 - \frac{1}{2}b_2 + \frac{1}{2}b_3 - b_5 + \frac{1}{2}b_6$. Since the corresponding row vectors $\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_5, \mathbf{a}_6\}$ are not linearly independent, we do not take it as a solution to x_i. The algorithm stops after finding out six solutions listed in the first column of Table 2. Accordingly, identifiable link e_1 has six solutions listed in the second column of Table 2.

TABLE 2
Solutions to variable x_1 in linear system and corresponding solutions to link e_1

Solution to x_1	**Solution to e_1**
$\frac{1}{2}b_1 + \frac{1}{2}b_2 - \frac{1}{2}b_4$	$\{p_1, p_2, p_4\}$
$\frac{1}{2}b_1 + \frac{1}{2}b_3 - \frac{1}{2}b_5$	$\{p_1, p_3, p_5\}$
$\frac{1}{2}b_2 + \frac{1}{2}b_3 - \frac{1}{2}b_6$	$\{p_2, p_3, p_6\}$
$b_3 + \frac{1}{2}b_4 - \frac{1}{2}b_5 - \frac{1}{2}b_6$	$\{p_3, p_4, p_5, p_6\}$
$b_1 - \frac{1}{2}b_4 - \frac{1}{2}b_5 + \frac{1}{2}b_6$	$\{p_1, p_4, p_5, p_6\}$
$b_2 - \frac{1}{2}b_4 + \frac{1}{2}b_5 - \frac{1}{2}b_6$	$\{p_2, p_4, p_5, p_6\}$

3.4 Extended Bipartite Model

As in the literature [6], the relationship between probing paths and target links is usually modeled as a bipartite graph $G_B =$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

(U, V, E), where U and V are two sets of vertices and E is a set of edges. Vertex $u_i \in U$ is probing path p_i, and $v_j \in V$ is target link e_j. If probing path p_i traverses target link e_j, the bipartite graph has an edge between u_i and v_j.

The traditional bipartite model reflects the coverage relation between a probing path and a target link. However, it cannot reflect which probing paths together can uniquely determine an identifiable target link. We propose an extended bipartite model to address this problem. Our extended bipartite model $G'_B = (U, V, E)$ has vertex sets U and V and edge set E. Vertex set U has two subsets U_1 and U_2, and vertex set V also has two subsets V_1 and V_2 as shown in Fig. 3. A vertex in set V_1 represents an identifiable target link, and a vertex in V_2 represents an unidentifiable target link. Each vertex in U_1 represents a solution to an identifiable target link, and hence it corresponds to a set of probing paths. Also, we use a vertex in U_2 to represent a probing path that can cover unidentifiable target links. Therefore, set E contains three kinds of edges as follows.

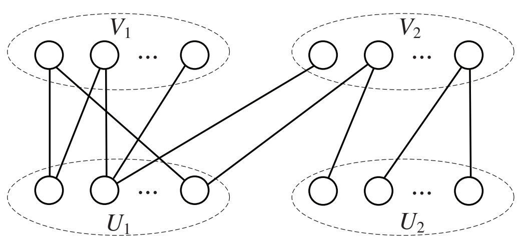


Fig. 3. An illustration of the extended bipartite model.

1) The edge between V_1 and U_1. For an identifiable target link represented by vertex v_i^1 in V_1, if vertex u_j^1 in U_1 represents a solution to this link, there is an edge connecting v_i^1 and u_j^1. The edge reflects the identifiability relationship between a solution and an identifiable target link.
2) The edge between V_2 and U_1. For an unidentifiable target link represented by vertex v_i^2 in V_2, if the probing paths represented by vertex u_j^1 in U_1 can cover this link, there is an edge between v_i^2 and u_j^1. The edge reflects the coverage relation between a solution and an unidentifiable target link.
3) The edge between V_2 and U_2. For an unidentifiable target link represented by vertex v_i^2 in V_2, if the probing path represented by vertex u_j^2 in U_2 can cover this link, there is an edge between v_i^2 and u_j^2. The edge reflects the coverage relation between a single probing path and an unidentifiable target link.

There is no edge between sets V_1 and U_2. For identifiable target links, we need to select probing paths to uniquely determine them, which is achieved by choosing their solutions from set U_1. Edges between V_1 and U_1 are sufficient for this task. Thus, we do not add any edge between an identifiable target link and a single probing path.

The commonly used bipartite model is a special case of our extended bipartite model. If we do not consider the identifiability, we can intentionally mark all target links as unidentifiable to make set V_1 empty. Accordingly, set U_1 becomes empty because there is no solution. Hence, the extended bipartite model turns into the commonly used bipartite model.

3.5 Path Selection Algorithm

Through the extended bipartite model, our path selection problem equals to selecting a set of vertices from $U_1 \cup U_2$, so that all vertices in $V_1 \cup V_2$ are their neighbors. We have the following theorem.

Theorem 2: The path selection problem is NP-hard.

Proof: The path selection problem can be proved to be NP-hard via proving that the set cover problem is a special case of this problem. In the extended bipartite model, vertex $u_i^1 \in U_1$ represents a set of probing paths and vertex $u_j^2 \in U_2$ represents a single probing paths. Consider the number of probing paths represented by a vertex in set U_1 and U_2, we have $|u_i^1| \geq 1$ and $|u_j^2| = 1$. By setting $|u_i^1|$ to 1 for each vertex $u_i^1 \in U_1$, it is equivalent to minimizing the number of vertices selected from $U_1 \cup U_2$, which is the same as the set cover problem. Therefore, the set cover problem is a special case of our path selection problem. Since the set cover problem is NP-hard, our problem is also NP-hard. □

Since this problem is NP-hard, we propose a heuristic based algorithm to solve it based on the extended bipartite model. The basic idea is to greedily select probing paths until all identifiable target links can be uniquely determined and all unidentifiable target links are covered. The idea of greedy selection is widely used for probing path selection, because it has low computational complexity and usually achieves good performance. A large-scale network can have thousands of probing paths, and thus an identifiable target link may have a large number of solutions. To enhance the scalability of our approach, we introduce a parameter α to control how many solutions are used for an identifiable target link. The computational complexity of the algorithm directly relates to α. A smaller α makes the algorithm faster. On the other hand, a larger α is helpful for reducing the probing cost as shown in our evaluation in Section 4.3.

The path selection algorithm is shown in Algorithm 2. It takes linear system LS and parameter α as input, and returns a set of probing paths that can uniquely determine all identifiable target links and cover all unidentifiable target links. The algorithm first calculates solutions to each identifiable target link with the algorithm `SolutionCalculation`, and then selects α solutions for each of them. Next, it builds the extended bipartite graph. Then, the algorithm starts to select probing paths in following three steps.

First, the algorithm deals with identifiable target links (line 8–16), because uniquely determining an identifiable link is more complex and needs more probing paths than covering an unidentifiable link. Consider an identifiable target link represented by vertex $v_i^1 \in V_1$. We choose one of its solutions to uniquely determine this link. In the extended bipartite graph, it is equivalent to choosing a vertex u_j^1 from U_1 such that v_i^1 and u_j^1 are neighbors. There are three considerations when selecting such a vertex v_i^1 from U_1: 1) v_i^1 may have multiple neighbors in V_1, i.e., this set of probing paths can uniquely determine multiple identifiable target links; 2) v_i^1 represents a set of probing paths, and our objective is to minimize the number of selected probing paths; 3) v_i^1 may have neighbors in set V_2, i.e., this set of probing paths cover some unidentifiable target links. We intend to add as few probing paths as possible to uniquely determine as many identifiable target links as possible. Hence, the algorithm selects a vertex from U_1 with the maximal value of $\frac{numNewVertices1}{numNewPaths}$ in each round as shown in line 9. If there are multiple candidates in U_1, we further consider how many unidentifiable target links can be covered. This process continues until all identifiable target links can be uniquely determined.

Next, the algorithm selects probing paths for unidentifiable target links (line 17–25). When all identifiable target links are handled, unselected probing paths are contained in sets U_1 and U_2. Hence, the algorithm selects vertices from $U_1 \cup U_2$ until all vertices in V_2 have a neighbor in U_1 or U_2. We intend to add as few probing paths as possible to cover as many unidentifiable target links as possible. Accordingly, the algorithm chooses a vertex with the maximal value of $\frac{numNewVertices2}{numNewPaths}$ in each round as shown in line 18.

Finally, the algorithm removes all replaceable ones from the selected probing paths (line 26). In the first two steps, the probing paths selected in each round do not contain replaceable ones. However, this cannot ensure that the final set of selected probing paths does not contain replaceable probing paths. Therefore, after finishing probing path selection, the algorithm removes all replaceable ones from the selected probing paths. This is achieved by the linear system based method introduced in Section 2.2.

Theorem 3 (Correctness): The set of probing paths returned by the algorithm `PathSelection` can uniquely determine all identifiable target links and cover all unidentifiable target links.

Proof: Before removing redundancy, probing paths in set P_s can uniquely determine all identifiable target links and cover all unidentifiable target links. Thus, we need to prove that removing replaceable probing paths from P_s does not affect the correctness.

Suppose set P_s contains s probing paths $p_1, \cdots, p_s$, in which $p_{r+1}, \cdots, p_s$ are replaceable and removed from set P_s. Consider a removed probing path p_k. In the dependency matrix, row vector $\mathbf{a}_k$ can be linearly expressed by row vectors $\mathbf{a}_1, \cdots, \mathbf{a}_r$. Accordingly, variable b_k is a linear combination of $b_1, \cdots, b_r$ as shown in Eq. (12).

$$b_k = \sum_{i=1}^{r} c_i b_i \tag{12}$$

We first prove that removing redundancy does not affect uniquely determining identifiable target links. Given an identifiable target link e_j, it can be uniquely determined by probing paths $p_1, \cdots, p_s$. Hence, the linear system variable x_j can be linearly expressed by variables $b_1, \cdots, b_s$ as shown in Eq. (13).

Algorithm 2 PathSelection

Input: The linear system LS and parameter α
Output: A set of probing paths P_s
Procedure:
1: **for** each identifiable target link **do**
2:     Calculate solutions to it with the algorithm `SolutionCalculation`, and then select α solutions.
3: **end for**
4: Construct the extended bipartite graph $G'_B = (U, V, E)$ with the selected solutions.
5: $P_s = \emptyset$
6: Mark all vertices in V_1, V_2, U_1, and U_2 as uncovered.
7: Mark all probing paths as unused
8: **while** V_1 has uncovered vertices **do**
9:     Select an uncovered vertex u_m^1 from U_1 with the largest $\frac{numNewVertices1}{numNewPaths}$, where $numNewVertices1$ is the number of uncovered vertex in V_1 connected to u_m^1, and $numNewPaths$ is the number of unused probing path contained in u_m^1. If there are multiple candidates, select the one that connects to the maximal uncovered vertices in V_2.
10:     Mark u_m^1 as covered.
11:     Mark all uncovered vertices in $V_1 \cup V_2$ connected to u_m^1 as covered.
12:     **for** each unused probing path p_i in u_m^1 **do**
13:         $P_s = P_s \cup p_i$
14:         Mark p_i as used.
15:     **end for**
16: **end while**
17: **while** V_2 has uncovered vertices **do**
18:     Select an uncovered vertex u_m^2 from $U_1 \cup U_2$ that has the largest $\frac{numNewVertices2}{numNewPaths}$, where $numNewVertices2$ is the number of uncovered vertex in V_2 connected to u_m^2, and $numNewPaths$ is the number of unused probing path contained in u_m^1.
19:     Mark u_m^2 as covered.
20:     Mark all uncovered vertices in V_2 connected to u_m^2 as covered.
21:     **for** each unused probing path p_j in u_m^2 **do**
22:         $P_s = P_s \cup p_j$
23:         Mark p_j as used.
24:     **end for**
25: **end while**
26: Remove all replaceable probing paths from set P_s.

$$\begin{aligned} x_j &= \sum_{i=1}^{s} d_i b_i \\ &= \sum_{i=1}^{k-1} d_i b_i + d_k b_k + \sum_{i=k+1}^{s} d_i b_i \end{aligned} \tag{13}$$

Replacing b_k in Eq. (13) with Eq. (12) results in a new linear expression in Eq. (14).

$$x_j = \sum_{i=1}^{k-1} d_i b_i + d_k \sum_{i=1}^{r} c_i b_i + \sum_{i=k+1}^{s} d_i b_i \tag{14}$$

Since $r+1 \le k \le s$, the above equation is equivalent to Eq. (15). It shows that x_j is a linear combination of $b_1, \cdots, b_{k-1}, b_{k+1}, \cdots, b_s$, which means that link e_k can be uniquely determined after removing the probing path p_k.

$$\begin{aligned} x_j &= \sum_{i=1}^{r} d_i b_i + \sum_{i=r+1}^{k-1} d_i b_i + d_k \sum_{i=1}^{r} c_i b_i + \sum_{i=k+1}^{s} d_i b_i \\ &= \sum_{i=1}^{r} (d_i + d_k c_i) b_i + \sum_{i=r+1}^{k-1} d_i b_i + \sum_{i=k+1}^{s} d_i b_i \end{aligned} \tag{15}$$

When removing probing path p_k, variable b_k in Eq. (14) is replaced by a linear combination of $b_1, \cdots, b_r$. We can remove all replaceable probing paths from set P_s with the same method. Finally, x_j becomes a linear combination of $b_1, \cdots, b_r$. It shows that e_j can be uniquely determined by $p_1, \cdots, p_r$.

Next, we prove that removing p_k does not affect covering unidentifiable target links. Suppose unidentifiable target link e_u is covered by p_k. Then, element $a_{k,u}$ in the dependency matrix is 1. Since row vector $\mathbf{a}_k$ can be linearly expressed by row vectors $\mathbf{a}_1, \cdots, \mathbf{a}_r$, elements $a_{1,u}, \cdots, a_{r,u}$ cannot be all 0, which means that at least one of $a_{1,u}, \cdots, a_{r,u}$ is 1. That is, at least one probing path among $p_1, \cdots, p_r$ can cover link e_u. Therefore, removing p_k does not affect covering e_u.

In conclusion, after removing redundancy, probing paths in set P_s can still uniquely determine all identifiable target links and cover all unidentifiable target links. □

Next, we present the performance bound of the path selection algorithm. An example in Section 4.5 will show that the number of selected probing path can be smaller than the row rank of the dependency matrix.

Theorem 4 (Performance bound): The upper bound of the number of probing paths selected by the algorithm `PathSelection` is the row rank of the dependency matrix.

Proof: Consider set P_s before removing its replaceable probing paths, we have $P_s \subseteq P$. Similar to P, set P_s corresponds to a linear system LS_s in the form of Eq. (2). The dependency matrix A_s of LS_s is formed by some row vectors of the original dependency matrix A. Therefore, the row rank of A_s is no larger than that of A.

After removing all replaceable probing paths, the probing paths left in P_s correspond to another linear system LS'_s. Its dependency matrix A'_s is formed by the maximal independent set of row vectors of A_s. Therefore, the rank of A'_s is the same as that of A_s, which is no larger than the rank of matrix A.

In conclusion, the number of probing paths returned by the algorithm is no larger than the row rank of the dependency matrix. □

Finally, we use an example to show our algorithm selects probing paths for target links e_1 and e_4 in Fig. 1. In the example, we use all solutions for path selection. The corresponding bipartite graph is shown in Fig. 4. Set U_1 has six vertices, each of which represents a solution to identifiable target link e_1. Set U_2 has three vertices, because there are three probing paths p_3, p_5, and p_6 that traverse unidentifiable target link e_4. Vertex e_4 in V_2 has no edge to vertex $\{p_1, p_2, p_4\}$ in U_1, because these three probing paths do not traverse e_4. The algorithm first chooses a set of probing paths for e_1. Among all six vertices in U_1, vertices $\{p_1, p_2, p_4\}$, $\{p_1, p_3, p_5\}$, and $\{p_2, p_3, p_6\}$ have the same value of $\frac{numNewVertices1}{numNewPaths}$. Since $\{p_1, p_3, p_5\}$ and $\{p_2, p_3, p_6\}$ can also cover one unidentifiable link, the algorithm selects one of them, e.g., $\{p_1, p_3, p_5\}$. After choosing it, all vertices in $V_1 \cup V_2$ are marked as covered. Since the selected set does not contain any replaceable probing path, the algorithm stops and returns a set of probing paths $\{p_1, p_3, p_5\}$.

4 PERFORMANCE EVALUATIONS

This section evaluates the performance of our algorithm. We study how the performance is affected by network topologies, the percentage of nodes for probing, and the percentage of target links. Moreover, we compare the performance of our method with prior work [18].

4.1 Simulation Setup

The simulation is based on nine ISP topologies derived by the Rocketfuel project [30], which are widely used in evaluation of related works. Table 3 summarizes the number of nodes and links in each topology. All topologies adopt the shortest path routing calculated based on link cost. Since probes traverse routing paths, some links cannot be covered by probes. Hence, we select target links only from the links that can be covered by probing paths.

We consider three parameters in the evaluation. The first is the percentage of target links. We randomly select 10% to 100% of the links that can be covered by probing paths as target links. The second is the percentage of *probers*, i.e., routers that are directly connected by end systems. We randomly select 20%, 40%, and 60% of routers as probers. The third is parameter α in the algorithm `PathSelection`. We set it to 1, 10, 100, and 1000. The path selection algorithm does not specify how to select α solutions; thus we randomly select α solutions in the evaluation. Different strategies for selecting α to build the extended bipartite graph are left for future work. For each simulation, we run it 100 times and report the average result.

We define the following two metrics to quantify the performance.

1) *Overall probing cost*: It is the number of the selected probing paths.
2) *Cost per identifiable target link*: The algorithm `PathSelection` first selects probing paths to uniquely determine identifiable target links, and then to cover unidentifiable target links. We count the first part of probing cost and amortize it to identifiable target links. In linear system, n linear equations can solve at most n variables. Therefore, the lower bound of the cost per identifiable target link is 1.

4.2 Percentage of Identifiable Links

In this subsection, we investigate how many links are identifiable in a network consisting of symmetrical and asymmetrical links. In the simulation, the cost of an asymmetrical link in each direction is set to a random number. Compared to networks of symmetrical links, the number of linear equations and variables are doubled in networks of asymmetrical links. For each topology, we randomly choose 10% to 100% of routers as probers and compute the percentage of identifiable links. We run each simulation 10,000 times and take the average.

The results on networks with symmetrical links are shown in Fig. 5. Using more probers results in a higher percentage of identifiable links. When every node is a prober, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

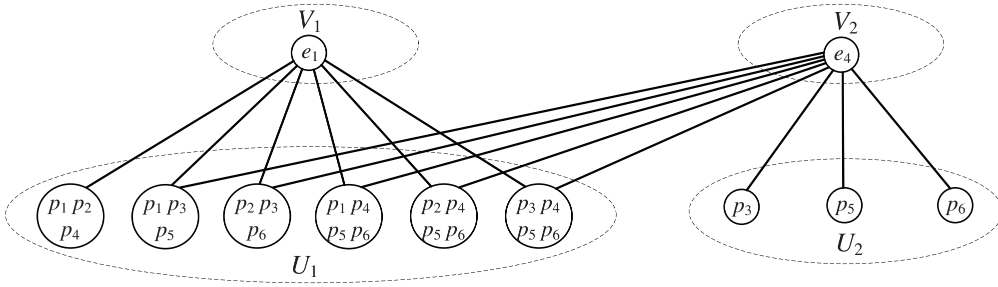


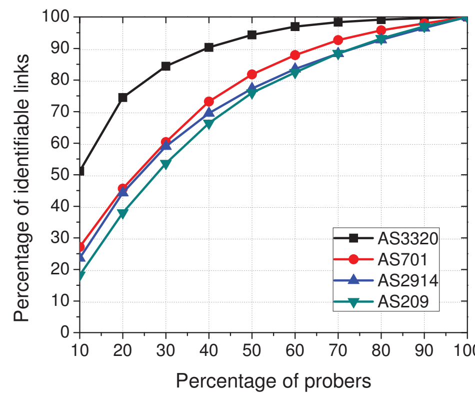
Fig. 4. The extended bipartite graph for selecting probing paths for target links e_1 and e_4 in Fig. 1.

TABLE 3
Summary of Topologies Used in Simulation

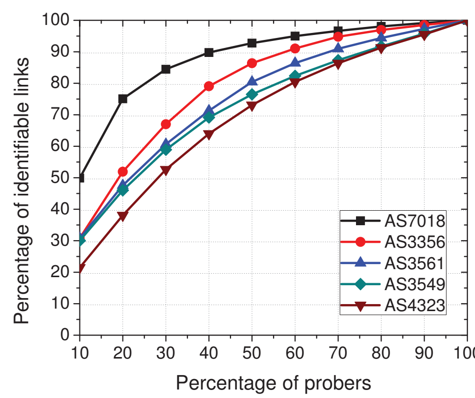
Topology	AS209	AS701	AS2914	AS3320	AS3356	AS3549	AS3561	AS4323	AS7018
Nodes	58	83	70	70	63	61	92	51	115
Links	108	219	111	355	285	486	329	161	148

performance of each link can be directly measured, and thus links are all identifiable. In networks of symmetrical links, the percentage of identifiable links is quite high. In all nine topologies, more than 50% of links are identifiable if using using 30% of nodes as probers. In particular, this percentage is as high as 85% in AS3320 and AS7018.

The results on networks with asymmetrical links are shown in Fig. 6. The percentage of identifiable links is lower than that in networks with symmetrical links, but it is still high. In summary, the simulation result shows that there are quite many identifiable links even when links are asymmetrical.



(a) AS209, AS701, AS2914, AS3320



(b) AS3356, AS3549, AS3561, AS4323, AS7018

Fig. 5. The percentage of identifiable links under different percentages of probers (links are symmetrical).

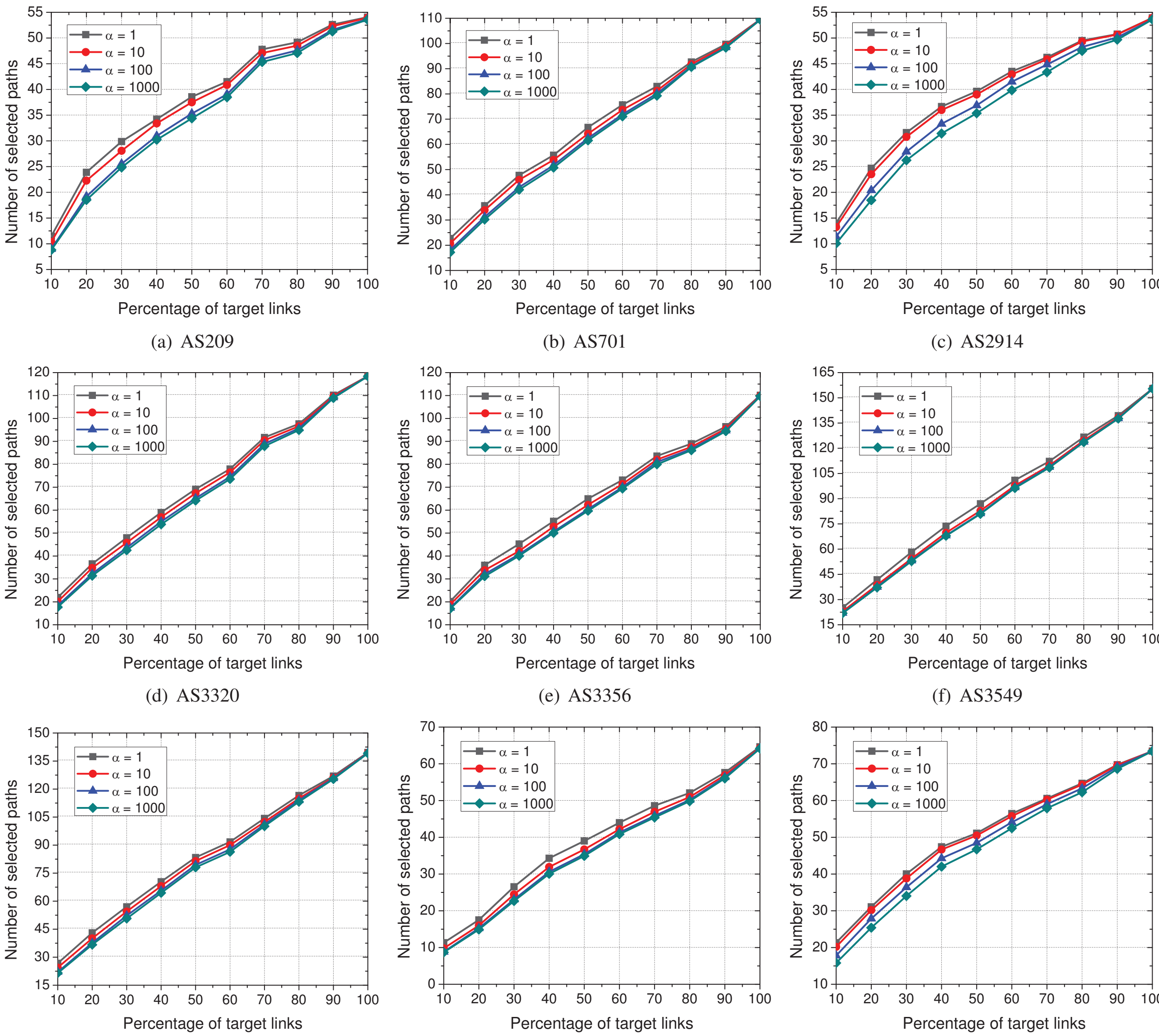
4.3 Overall Probing Cost

Next, we evaluate the overall probing cost. Fig. 7 shows the overall probing cost when 40% of nodes are probers, which has two features. First, increasing parameter α, i.e., using more solutions for path selection, can reduce the overall probing cost, especially in AS209, AS2914, AS4323, and AS7018. These four topologies have small scales compared to others. Hence, for a small-scale network we may choose a large α to achieve good performance. For a large-scale network, we can use a small α to significantly reduce the running time but sacrifice a little bit of performance. Second, as the percentage of target links increases, more and more non-redundant probing paths are selected. Accordingly, the overall probing cost gradually converges to the theoretical upper bound shown in Theorem 4. Therefore, when the percentage of target links reaches 100%, the overall probing cost under different α becomes similar. When 20% and 60% of nodes are probers, the overall probing cost has similar trend, and will not be shown here.

4.4 Cost Per Identifiable Target Link

Fig. 8 shows the cost per identifiable target link with $\alpha = 1000$, when 20%, 40%, and 60% of nodes are probers. In all topologies, this cost quickly decreases as the percentage of target links increases. When the percentage of target links reaches 100%, the cost per identifiable target link is very close to the lower bound 1. This indicates that our path selection algorithm can effectively eliminate redundant probing paths and choose probing paths that are the most useful for determining multiple identifiable target links. The figure also shows that using more nodes as probers is useful for reducing the cost. Given n probers, we have $\frac{n(n-1)}{2}$ usable probing paths. Hence, we have much more usable probing paths when the percentage

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.



(a) AS209 (b) AS701 (c) AS2914
(d) AS3320 (e) AS3356 (f) AS3549
(g) AS3561 (h) AS4323 (i) AS7018

Fig. 7. The overall probing cost of the path selection algorithm with $\alpha = 1, 10, 100, 1000$ (40% nodes as probers).

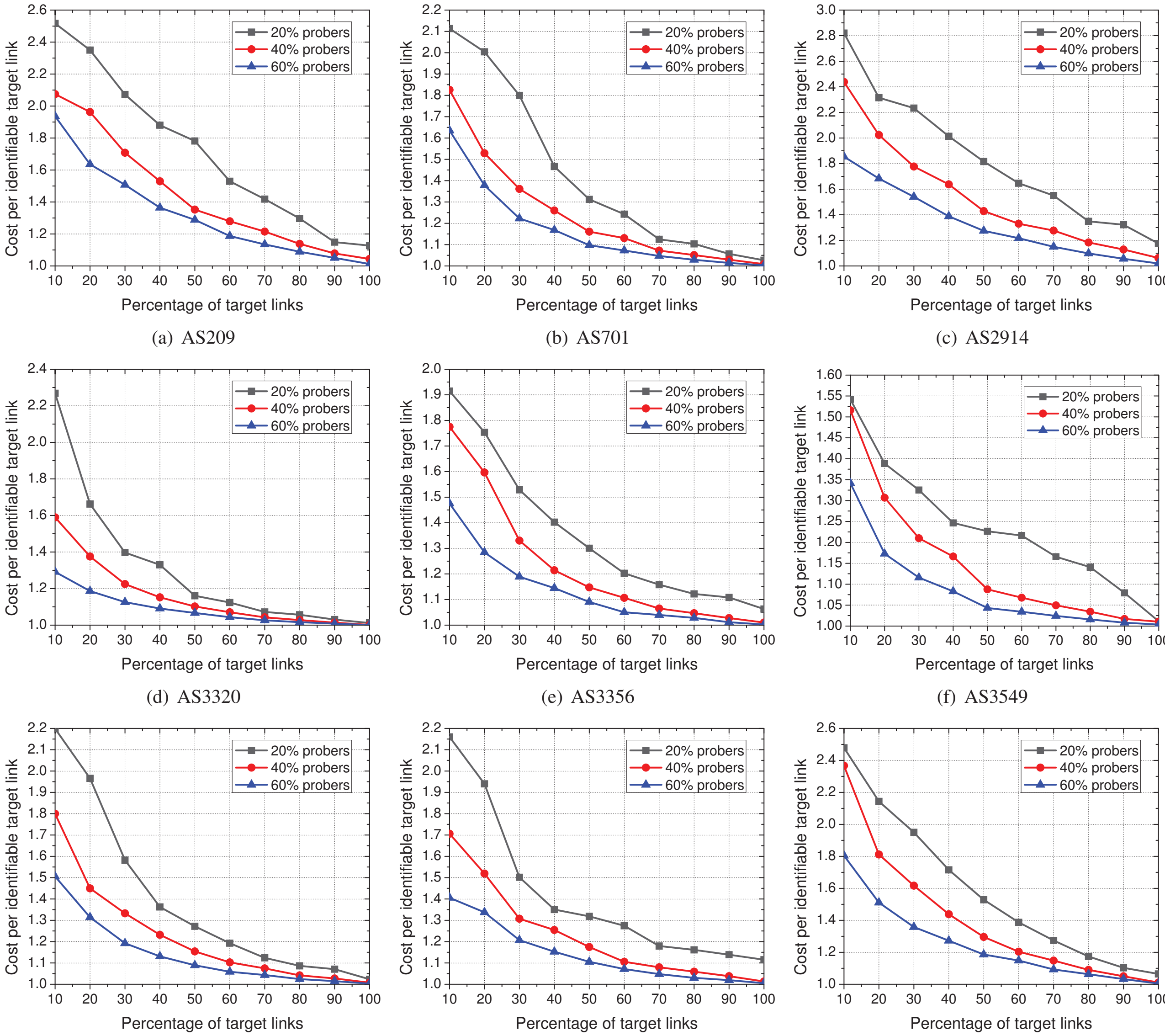
of probers increases. As a result, our algorithm can pick out better probing paths to reduce this amortized cost. When α is 1, 10, and 100, the cost per identifiable target link has similar trend, and will not be shown here.

4.5 Comparisons

The SelectPath algorithm [18] is the closest work in this area. It can select a minimum set of probing paths to uniquely determine all identifiable links and cover all unidentifiable links in the network. However, this algorithm is only a special case of what our algorithm can do, i.e., when all links are target links. Even for this special case, we show that the performance of our solution is guaranteed to be better than or equal to theirs.

Since the SelectPath algorithm cannot be modified for an arbitrary set of target links, we only compare the performance on the case that all links are target links, although this is not fair to our algorithm which is more general. As shown in [18], the overall probing cost of the SelectPath algorithm is equal to the row rank of the dependency matrix. According to Theorem 4, the row rank of the dependency matrix is the upper bound of the overall probing cost of our algorithm. As a result, our method is theoretically not worse than the SelectPath algorithm.

In addition to the above theoretical result, we also use simulations to compare the overall probing cost when all links are target links. In all 3 (percentages of probers) $\times 9$ (topologies) $\times$ 4 $(\alpha) \times 100$ (runs) $= 10,800$ runs of simulations, there is not a single case that the SelectPath algorithm outperforms our algorithm, which is consistent with the theoretical result. Table 4, Table 5, and Table 6 show the average overall probing cost of our algorithm with different α and the SelectPath algorithm when 20%, 40%, and 60% of nodes are probers. The simulation result indicates that our algorithm is better than the SelectPath algorithm.



(a) AS209 (b) AS701 (c) AS2914
(d) AS3320 (e) AS3356 (f) AS3549
(g) AS3561 (h) AS4323 (i) AS7018

Fig. 8. The cost per identifiable target link of the path selection algorithm with $\alpha = 1000$. The lower bound is 1.

TABLE 4
Comparison of overall probing cost of our algorithm and the SelectPath algorithm (percentage of probers: 20%)

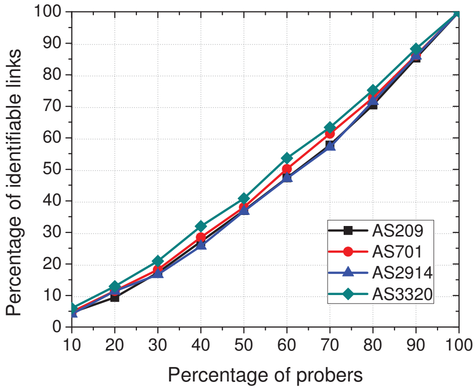
Topology	α				SelectPath
	1	10	100	1000	
AS209	23.9	23.7	23.4	23.3	25.4
AS701	50.6	50.5	50.4	50.4	51.6
AS2914	27.9	27.6	27.2	27.1	29.9
AS3320	48.9	48.8	48.8	48.8	49.3
AS3356	47.9	47.6	47.2	47.2	48.8
AS3549	53.2	53.0	53.0	53.0	53.8
AS3561	58.0	58.0	57.8	57.7	59.8
AS4323	26.8	26.8	26.7	26.7	27.1
AS7018	42.8	42.7	42.6	42.6	43.4

TABLE 5
Comparison of overall probing cost of our algorithm and the SelectPath algorithm (percentage of probers: 40%)

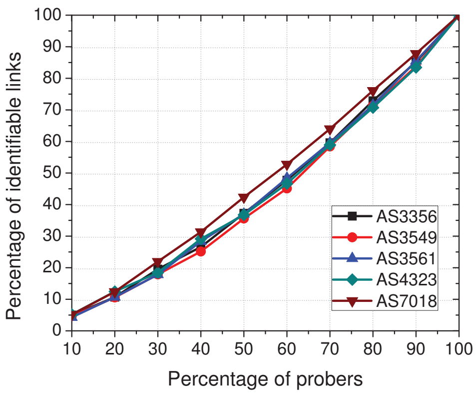
Topology	α				SelectPath
	1	10	100	1000	
AS209	52.7	52.6	52.3	52.2	55.7
AS701	107.6	107.6	107.5	107.4	109.0
AS2914	54.2	54.2	53.9	53.9	56.0
AS3320	119.0	119.0	118.9	118.9	119.3
AS3356	107.2	107.0	106.8	106.7	108.6
AS3549	155.0	155.0	155.0	155.0	157.3
AS3561	137.6	137.3	137.2	137.1	140.6
AS4323	64.2	63.9	63.7	63.7	64.8
AS7018	73.6	73.6	73.6	73.6	74.1

The SelectPath algorithm selects probing paths corresponding to the maximal independent set of row vectors of the dependency matrix. Although vectors in this set are linearly independent, it does not necessarily mean there is no redundancy. Fig. 2(b) is a simple example to show this fact. There are three identifiable links e_1, e_2, and e_3. The maximal

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.



(a) AS209, AS701, AS2914, AS3320



(b) AS3356, AS3549, AS3561, AS4323, AS7018

Fig. 6. The percentage of identifiable links under different percentages of probers (links are asymmetrical).

TABLE 6
Comparison of overall probing cost of our algorithm and the SelectPath algorithm (percentage of probers: 60%)

Topology	α				SelectPath
	1	10	100	1000	
AS209	74.9	74.8	74.6	74.6	76.7
AS701	149.2	149.2	149.2	149.1	151.0
AS2914	74.8	74.8	74.7	74.7	76.0
AS3320	194.3	194.2	94.2	194.2	194.7
AS3356	174.9	174.9	174.9	174.8	175.7
AS3549	269.2	268.9	268.7	268.7	272.2
AS3561	208.4	208.3	208.1	207.9	210.3
AS4323	99.6	99.4	99.2	99.2	100.6
AS7018	100.3	100.3	100.2	100.2	100.4

independent set of row vectors contains 4 vectors. However, we can use only three probing paths p_1, p_2, and p_4 to uniquely determine links e_1, e_2, and e_3. Adding any other probing paths has no contribution to achieving identifiability, no matter it is replaceable by p_1, p_2, and p_4 or not. Our algorithm outperforms the SelectPath algorithm because we try to avoid not only replaceable probing paths but also probing paths without any contribution to achieving identifiability.

5 Related Work

Minimizing the probing cost is usually achieved by carefully selecting probing paths. This problem has been studied without resource limitation [3], [18], [20], and in scenarios with explicit operational requirement [9] and resource constraint [31]. Various kinds of probing costs and monitoring objectives have been studied. However, as pointed out in [8], many network inference problems are ill-posed, because the number of measurements are not sufficient to uniquely determine the result. Our work jointly addresses the problem of minimizing the probing cost and achieving identifiability.

Achieving identifiability in network link monitoring is also considered in [4], [16]–[18]. However, our work is quite different from theirs. Zhao *et al* [4] proposed a method for determining the shortest sequence of links whose properties can be uniquely identified by end-to-end probes. It is similar to differentiating the identifiable from unidentifiable links in our work. However, their method does not consider how to minimize the probing cost. The failure localization in [16] aims at accurately pinpointing a failed link, which in essence is to achieve identifiability. However, it only focuses on locating a single link. Our method is able to uniquely determine every identifiable link. Both [17] and [18] deal with the problem of selecting a minimum set of probing paths that can uniquely determine all identifiable links in the network. It is only a special case of what our algorithm can address, i.e., when all links are target links. Even for this special case, the performance of our solution is guaranteed to be better than or equal to theirs. Another major difference is the method we adopt. Their methods are based on eliminating replaceable paths; i.e., linearly dependent row vectors in the dependency matrix. We take a different approach, and only select probing paths that are the most useful to our objective. In addition to eliminating replaceable probing paths, we also try to avoid selecting probing paths without any contribution to achieving identifiability.

There are some other approaches to infer the delay and loss rate of network links. Nguyen *et al* [8] exploited the second-order statistics for estimating the loss rate of links. It first infers the variance of the loss rate of links, and then uniquely determines the loss rate of some links with the highest variance. Based on this method, Ghita *et al* [10] designed an algorithm to minimize the estimation error rate by carefully selecting links with the highest variance of loss rate. Both approaches focus on providing an estimation of loss rate for each link. However, they do not deal with minimizing the probing cost. Additionally, they do not differentiate identifiable links from unidentifiable links. As a result, the estimated loss rate of an identifiable link may be inaccurate. Compared with them, our approach can obtain the accurate loss rate of each identifiable link.

6 Conclusions and Future Work

End-to-end probe has received considerable attention in network link monitoring. In this paper, we propose an approach to minimize the probing cost and achieve identifiability. Given a set of target links, the objective is to choose the minimum number of probing paths that can uniquely determine identifiable target links and cover unidentifiable target links. The basic idea is to select probing paths that are the most useful for achieving identifiability and covering unidentifiable target links, and prevent choosing redundant probing paths. Our method eliminates two types of redundant probing paths, i.e., those that can be replaced by others and those without any contribution to achieving identifiability. With our approach, the number of selected probing paths is proved to be bounded. Experiments based on ISP topologies demonstrate that our approach can achieve identifiability with very low probing cost. Compared with prior work, our method is more general and has better performance.

The algorithm proposed for calculating all solutions to an identifiable link may generate a large number of path sets when the network scale is large. It is possible that using only a small portion of these path sets can produce good enough results. As future work, we will investigate possible methods to effectively select the probing path sets that are most useful for reducing the overall probing cost.

7 Acknowledgments

The authors would like to thank the anonymous reviewers whose insightful comments helped us improve the presentation of the paper. A preliminary version [32] of the paper appeared in IEEE ICDCS '10. This work was supported in part by the Defense Threat Reduction Agency under grant HDTRA1-10-1-0085.

References

[1] Y. Shavitt, X. Sun, A. Wool, and B. Yener, "Computing the unmeasured: an algebraic approach to Internet mapping," in *IEEE INFOCOM*, 2001.
[2] R. R. Kompellay, J. Yatesz, A. Greenbergz, and A. C. Snoeren, "IP fault localization via risk modeling," in *USENIX NSDI*, 2005.
[3] F. Li and M. Thottan, "End-to-end service quality measurement using source-routed probes," in *IEEE INFOCOM*, 2006.
[4] Y. Zhao, Y. Chen, and D. Bindel, "Towards unbiased end-to-end network diagnosis," in *ACM SIGCOMM*, 2006.
[5] A. Dhamdhere, R. Teixeira, C. Dovrolis, and C. Diot, "NetDiagnoser: troubleshooting network unreachabilities using end-to-end probes and routing data," in *ACM CoNEXT*, 2007.
[6] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, "Detection and localization of network black holes," in *IEEE INFOCOM*, 2007.
[7] P. P. C. Lee, V. Misra, and D. Rubenstein, "Toward optimal network fault correction via end-to-end inference," in *IEEE INFOCOM*, 2007.
[8] H. X. Nguyen and P. Thiran, "Network loss inference with second order statistics of end-to-end flows," in *ACM SIGCOMM Conference on Internet Measurement*, 2007.
[9] Y. Zhao, Z. Zhu, Y. Chen, D. Pei, and J. Wang, "Towards efficient large-scale VPN monitoring and diagnosis under operational constraints," in *IEEE INFOCOM*, 2009.
[10] D. Ghita, H. Nguyen, M. Kurant, K. Argyraki, and P. Thiran, "Netscope: Practical network loss tomography," in *IEEE INFOCOM*, 2010.
[11] Y. Gu, G. Jiang, V. Singh, and Y. Zhang, "Optimal probing for unicast network delay tomography," in *IEEE INFOCOM*, 2010.
[12] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2 (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., 1998.
[13] H. H. Song, L. Qiu, and Y. Zhang, "NetQuest: a flexible framework for large-scale network measurement," in *ACM SIGMETRICS*, 2006.
[14] M. H. Gunes and K. Sarac, "Analyzing router responsiveness to active measurement probes," in *Passive and Active Measurement Conference*, 2009.
[15] R. Sherwood and N. Spring, "Touring the Internet in a TCP sidecar," in *ACM SIGCOMM Conference on Internet Measurement*, 2006.
[16] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in IP networks," in *IEEE INFOCOM*, 2003.
[17] H. X. Nguyen and P. Thiran, "Active measurement for multiple link failures diagnosis in IP networks," in *Passive and Active Measurement Conference*, 2004.
[18] Y. Chen, D. Bindel, H. Song, and R. H. Katz, "An algebraic approach to practical and scalable overlay network monitoring," in *ACM SIGCOMM*, 2004.
[19] S. Agrawal, K. Naidu, and R. Rastogi, "Diagnosing link-level anomalies using passive probes," in *IEEE INFOCOM*, 2007.
[20] P. Barford, N. Duffield, A. Ron, and J. Sommers, "Network performance anomaly detection and localization," in *IEEE INFOCOM*, 2009.
[21] T. Bu, N. Duffield, F. L. Presti, and D. Towsley, "Network tomography on general topologies," in *ACM SIGMETRICS*, 2002.
[22] J. Wu, Y. Zhang, Z. M. Mao, and K. G. Shin, "Internet routing resilience to failures analysis and implications," in *ACM CoNEXT*, 2007.
[23] Y. Breitbart, C.-Y. Chan, M. Garofalakis, R. Rastogi, and A. Silberschatz, "Efficiently monitoring bandwidth and latency in IP networks," in *IEEE INFOCOM*, 2001.
[24] L. Li, M. Thottan, B. Yao, and S. Paul, "Distributed network monitoring with bounded link utilization in IP networks," in *IEEE INFOCOM*, 2003.
[25] M. Thottan, L. E. Li, B. Yao, V. S. Mirrokni, and S. Paul, "Distributed network monitoring for evolving IP networks," in *IEEE ICDCS*, 2004.
[26] K. Suh, Y. Guo, J. Kurose, and D. Towsley, "Locating network monitors: complexity, heuristics, and coverage," in *IEEE INFOCOM*, 2005.
[27] Q. Zheng, G. Cao, T. L. Porta, and A. Swami, "Detecting and localizing large-scale router failures using active probes," in *IEEE MILCOM*, 2011.
[28] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," in *ACM SIGCOMM Workshop on Internet measurment*, 2002.
[29] G. H. Golub and C. F. V. Loan, *Matrix Computations (3rd Edition)*. Johns Hopkins University Press, 1996.
[30] R. Sherwood, A. Bender, and N. Spring, "Measuring ISP topologies with Rocketfuel," in *ACM SIGCOMM*, 2002.
[31] H. X. Nguyen, R. Teixeira, P. Thiran, and C. Diot, "Minimizing probing cost for detecting interface failures: Algorithms and scalability analysis," in *IEEE INFOCOM*, 2009.
[32] Q. Zheng and G. Cao, "Minimizing probing cost and achieving identifiability in network link monitoring," in *IEEE ICDCS*, 2010.

Qiang Zheng received the BS degree from Nankai University, Tianjin, in 2004, and the ME degree from Chinese Academy of Sciences, Beijing, in 2007. He is currently a PhD candidate in the Department of Computer Science and Engineering, the Pennsylvania State University, University Park. His research interests include network failure detection, localization, and fast recovery. He is a student member of the IEEE.

Guohong Cao received the BS degree from Xian Jiaotong University, China. He received the MS degree and PhD degree in computer science from the Ohio State University in 1997 and 1999 respectively. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently a Professor. His research interests are wireless networks and mobile computing. He has published more than 150 papers in the areas of cache management, data access and dissemination, wireless sensor networks, wireless network security, vehicular ad hoc networks, and distributed fault tolerant computing. He has served on the editorial board of IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, IEEE Transactions on Vehicular Technology, and has served on the organizing and technical program committees of many conferences. He was a recipient of the NSF CAREER award in 2001. He is a Fellow of the IEEE.