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

Qiang Zheng, Student Member, IEEE, and Guohong Cao, Fellow, IEEE

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], [2], [3], [4], [5], [6], [7], [8], [9], [10], [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 noncooperative 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], [17], [18], [19], [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], [17], [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 $p_1$, $p_2$, $p_3$, $p_4$, and $p_5$, we can uniquely infer the performance of $e_1$, $e_2$, and $e_3$, but cannot uniquely infer the performance of $e_4$ and $e_5$. To uniquely infer the performance of a link, we need to select probing paths that can uniquely determine the performance of the link.
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.

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 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.

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], [24], [25], [26], [27], we model the network as a connected undirected graph \( G(V, E) \), where \( V \) is the set of nodes (routers) and \( E = \{ e_i | 1 \leq i \leq m \} \) 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 \( r_1 \) to \( s_4 \). 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 \( P = \{ p_i | 1 \leq i \leq n \} \) which can be used for monitoring a set of \( m \) target links \( E_t \subseteq E \). For simplicity, target links are labeled as \( e_1, \ldots, e_m \). 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], [17], [18], [19], [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 probing results. 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 probing results. 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 $LS = \{l_s|1 \leq i \leq n\}$, in which $l_s$ is the $i$th linear equation as shown in (1). Binary variable $a_{ij}$ is 1 if probing path $p_i$ covers link $e_j$; otherwise it is 0. Variables $x_i$ and $b_i$ represent the performance of link $e_j$ and probing path $p_i$. Thus, (1) means that the performance of $p_i$ is the addition of the performance of all links on $p_i$

$$\sum_{j=0}^{m} a_{ij} x_j = b_i.$$  \hspace{1cm} (1)

The linear system can be written in the matrix computation form as shown in (2). Variable $x$ is the vector form of $x_1, \ldots, x_m$ and $b$ is the vector form of $b_1, \ldots, b_n$. The coefficient matrix $A = (a_1, \ldots, a_n)^T$ is also called the dependency matrix [17]

$$Ax = b.$$ \hspace{1cm} (2)

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 $x_1, \ldots, x_5$ as shown in Fig. 2a. The dependency matrix is shown in Fig. 2b. 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. 2a, $b_1$ 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 $\prod_{i=0}^{m}(1-a_{ij}) = 1-q_i$. By applying logarithm to both sides, we have $\log(\prod_{i=0}^{m}(1-a_{ij})) = \log(1-q_i)$, which results in $\sum_{i=0}^{m} \log(1-a_{ij}) = \log(1-q_i)$. There are two cases for the value of $1-a_{ij}$. If $a_{ij} = 1$, i.e., probing path $p_i$ covers link $e_j$, then $1-a_{ij} = 1 - r_j$. If $a_{ij} = 0$, we have $1-a_{ij} = 1$, and thus $\log(1-a_{ij}) = 0$. For each probing path $p_i$, we define $b_i = \log(1-q_i)$. Similarly, for each link $e_j$, let $x_j = \log(1-r_j)$ if $e_j$ is on $p_i$, or $x_j = 0$ otherwise. Through this transformation, we can apply the linear equation in (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 $\log(1-(1-r_1)(1-r_2)) = \log(1-q_1)$. By defining $x_1 = \log(1-(1-r_1))$, $x_2 = \log(1-(1-r_2))$, and $b_1 = \log(1-q_1)$, this equation becomes $x_1 + x_2 = b_1$ which is the first equation in Fig. 2a.

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 $\log(1-q_i)$ and $\log(1-r_j)$ 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 percent of them last for less than 1 minute and 90 percent 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 $a_i$ in the dependency matrix can be linearly expressed by the row vectors

$$
\begin{align*}
&x_1 + x_2 = b_1 \\
&x_1 + x_2 = b_1 \\
&x_1 + x_2 = b_1 \\
&x_1 + x_2 = b_1 \\
&x_1 + x_2 = b_1 \\
&x_1 + x_2 = b_1
\end{align*}
$$

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

(a) The linear system model $LS$ (b) The dependency matrix

Fig. 2. The linear system model and the dependency matrix of the example in Fig. 1.
corresponding to this set of probing paths. In Fig. 2b, row vector \(a_5\) can be linearly expressed by row vectors \(a_2, a_3,\) and \(a_4\) as \(a_5 = -a_2 + a_3 + a_4\). 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 a 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. 2a is a solvable variable and \(\frac{1}{2} b_1 + \frac{1}{2} b_2 - \frac{1}{2} b_3\) is a solution to \(x_1.\) It contains three variables \(b_1, b_2,\) and \(b_3,\) and thus probing paths \(p_1, p_2,\) and \(p_3\) together can uniquely determine \(e_1.\) Since row vectors \(a_1, a_2,\) and \(a_3\) are linearly independent, each probing path is not replaceable by other two probing paths. Therefore, \(\{p_1, p_2, p_3\}\) 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 \(b_i/s.\) The corresponding probing paths may contain replaceable ones. For example, the linear expression \(b_1 - \frac{1}{2} b_2 + \frac{1}{2} b_3 - \frac{1}{2} b_4\) is a solution to \(x_1\) in Fig. 2a. However, the set of probing paths \(\{p_1, p_2, p_3, p_5, p_b\}\) 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

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G)</td>
<td>network under study</td>
</tr>
<tr>
<td>(m)</td>
<td>the number of links in (G)</td>
</tr>
<tr>
<td>(e_i)</td>
<td>the (i)th link in (G)</td>
</tr>
<tr>
<td>(n)</td>
<td>the number of probing paths in (G)</td>
</tr>
<tr>
<td>(p_i)</td>
<td>the (i)th probing path in (G)</td>
</tr>
<tr>
<td>(E_n)</td>
<td>set of target links</td>
</tr>
<tr>
<td>(m_t)</td>
<td>the number of target links</td>
</tr>
<tr>
<td>(LS)</td>
<td>linear system model of (G)</td>
</tr>
<tr>
<td>(l_s)</td>
<td>the (i)th linear equation of (LS)</td>
</tr>
<tr>
<td>(b_i)</td>
<td>measured performance of the (i)th probing path</td>
</tr>
<tr>
<td>(A)</td>
<td>dependency matrix</td>
</tr>
<tr>
<td>(S^r)</td>
<td>set of all solutions to the solvable variable (x_i) in (LS)</td>
</tr>
<tr>
<td>(S^l)</td>
<td>the (i)th solution to the solvable variable (x_i) in (LS)</td>
</tr>
<tr>
<td>(L)</td>
<td>set of linear expressions, (L = {l_i</td>
</tr>
<tr>
<td>(l_t)</td>
<td>linear expression for the (i)th row vector in (A_N)</td>
</tr>
</tbody>
</table>

### 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 Sections 3.2 and 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 section.

To decompose the dependency matrix, we divide its row vectors into two groups as shown in (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 \(A_R = (a_1, \ldots, a_r)^T\) and \(A_N = (a_{r+1}, \ldots, a_n)^T.\)

\[
A = \begin{pmatrix} A_R \\ A_N \end{pmatrix}. \tag{3}
\]

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 (2) is also decomposed into two parts as shown in (4), in which vector $b$ is partitioned into two vectors $b_R = (b_1, \ldots, b_{r})^T$ and $b_N = (b_{r+1}, \ldots, b_n)^T$

\[
\begin{bmatrix}
A_R & A_N \\
A_N & b_N \\
\end{bmatrix} x = \begin{bmatrix}
b_R \\
0 \\
\end{bmatrix}.
\]  

(4)

Since each row vector of $A_N$ is a linear combination of row vectors of $A_R$, a solvable/unsolvable variable in linear system $A_R x = b_R$ is also solvable/unsolvable in linear system $A x = b$. By solving linear system $A_R x = b_R$, we can determine whether a target link is identifiable or not. Also, we calculate a solution $S_i^1$ to each solvable variable $x_i$, which is referred to as the base solution. Solving linear system $A_R x = b_R$ 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 $b_i$s. Each row vector of $A_N$ can be linearly expressed by row vectors of $A_R$. By solving the linear system in (5), we can calculate the linear expression for row vector $a_{r+i}$ of $A_N$, where $i = 1, \ldots, n - r$

\[
\sum_{j=1}^{r} c_{ij} a_j + a_{r+i} = 0.
\]  

(5)

Multiplying vector $x$ to both sides of (5) results in

\[
\sum_{j=1}^{r} c_{ij} a_j x + a_{r+i} x = 0.
\]  

Since $a_j x = b_j$ for $1 \leq j \leq r$ and $a_{r+i} x = b_{r+i}$, we have the linear expression in (6). Through it, we obtain the relationship between $b_i$s, which will be used for calculating all solutions to identifiable target links.

We use $L = \{l_i | 1 \leq i \leq n - r\}$ to represent the set of these linear expressions, where $l_i$ is the expression containing $b_{r+i}$

\[
\sum_{j=1}^{r} c_{ij} b_j + b_{r+i} = 0.
\]  

(6)

The dependency matrix in Fig. 2b shows an example. A decomposition of this matrix is $A_R = (a_1, a_2, a_3, a_4)^T$ and $A_N = (a_5, a_6)^T$. The corresponding partition of vector $b$ is $b_R = (b_{1}, b_{2}, b_{3}, b_{4})^T$ and $b_N = (b_{5}, b_{6})^T$. By solving linear system $A_R x = b_R$, 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 $b_1 - b_2 - b_4 + b_6 = 0$.

### 3.3 Solution Calculation

In this section, 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 $S_i^r$ obtained from linear system $A_R x = b_R$. For each linear expression $l_j \in L$, if it has a common variable $b_k$ with $S_i^r$, replacing $b_k$ in $S_i^r$ 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 $S_i^r$ 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 $S_i^t$ from the queue and apply all possible linear replacements to it. A linear replacement generates a linear expression $S_i^p$ (line 8). Then, in line 9, the algorithm checks if each variable in $S_i^p$ is replaceable by other variables in $S_i^r$. If not, $S_i^p$ corresponds to a solution to link $e_i$. We record $S_i^p$ by putting it into queue $Q$, if it is not in queue $Q$ and set $S_i^p$. After applying all linear replacements to the current solution $S_i^t$, we put it into set $S_i^s$. As a result, at any moment set $S_i^s$ 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 $S_i^s$.

Algorithm 1. SolutionCalculation

**Input**: Base solution $S_i^r$ and set $L$ of linear expressions

**Output**: Set $S_i^s$ containing all solutions to $x_i$

**Procedure**:

1. INIT($Q$) //Initialize a queue $Q$
2. $S_i^s \leftarrow \emptyset$
3. ENQUEUE($Q, S_i^r$)
4. while $Q \neq \emptyset$ do
5. $S_i^p \leftarrow$ DEQUEUE($Q$)
6. for each linear expression $l_j \in L$ do
7. for each common variable $b_k$ in $S_i^p$ and $l_j$ do
8. $S_i^t \leftarrow$ replace $b_k$ in $S_i^t$ with $l_j$
9. if each variable in $S_i^t$ cannot be replaced by other variables in $S_i^r$ and $S_i^t \notin Q \land S_i^t \notin S_i^s$ then ENQUEUE($Q, S_i^r$)
10. end if
11. end for
12. end for
13. end for
14. $S_i^l \leftarrow S_i^r \cup S_i^p$
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 $S_i^q$ which is not found by the algorithm. The missed solution $S_i^q$ cannot be linearly expressed by base solution $S_i^r$ 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 $b_j$, $S_i^q$ is in the following form:

\[
x_i = \sum_{j=1}^{n} d_{ij} b_j.
\]  

(7)

The above equation together with base solution $S_i^r$ and linear expressions in $L$ form a linear system as shown in (8). The first equation is a base solution $S_i^r$. Since it is computed from linear system $A_R x = b_R$, the value of $x_i$ is a linear combination of $b_j$s, where $1 \leq j \leq r$. The
The second equation is the same as (7). For the other $n - r$ equations, each of them is a linear expression in set $L$, which is shown in (6)

$$
\begin{align*}
& \sum_{i=1}^{r} d_{1i} b_i - x_i = 0 \\
& \sum_{i=1}^{n} d_{2i} b_i - x_i = 0 \\
& \sum_{j=1}^{r} c_{1j} b_j + b_{r+1} = 0 \\
& \sum_{j=1}^{n-r} c_{n-r,j} b_j + b_{n} = 0.
\end{align*}
$$

(8)

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

$$
\begin{bmatrix}
    b_1 & \cdots & b_r & b_{r+1} & \cdots & b_n & x_i \\
    d_{11} & \cdots & d_{1r} & 0 & \cdots & 0 & -1 \\
    d_{q1} & \cdots & d_{qr} & d_{qr+1} & \cdots & d_{qn} & -1 \\
    c_{11} & \cdots & c_{1r} & 1 & \cdots & 0 & 0 \\
    \vdots & & \vdots & \vdots & & \vdots & \vdots \\
    c_{n-r,1} & \cdots & c_{n-r,r} & 0 & \cdots & 1 & 0
\end{bmatrix}.
$$

(9)

we subtract $d_{q,r+k}$ times of the $(2+k)$th row from the second row, where $k = 1, \ldots, n - r$. 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 (10). Compared with the matrix in (9), the difference is only the second row. For $j = 1, \ldots, r$, variable $d^r_{qj}$ is shown in (11)

$$
\begin{bmatrix}
    b_1 & \cdots & b_r & b_{r+1} & \cdots & b_n & x_i \\
    d_{11} & \cdots & d_{1r} & 0 & \cdots & 0 & -1 \\
    d_{q1} & \cdots & d_{qr} & d_{qr+1} & \cdots & d_{qn} & -1 \\
    c_{11} & \cdots & c_{1r} & 1 & \cdots & 0 & 0 \\
    \vdots & & \vdots & \vdots & & \vdots & \vdots \\
    c_{n-r,1} & \cdots & c_{n-r,r} & 0 & \cdots & 1 & 0
\end{bmatrix}.
$$

(10)

$$
d^r_{qj} = d_{qj} - \sum_{k=1}^{n-r} c_{kj} d_{q,r+k}.
$$

(11)

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

A solution to variable $x_i$ is a linear combinations of $b_1, \ldots, b_n$. Since $b_j$ corresponds to probing path $p_j$, we can easily obtain the corresponding solution to identifiable link $e_j$. 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 $l_1 : b_2 - b_3 - b_4 = 0$ and $l_2 : b_1 - b_3 - b_4 = 0$. The base solution has two common variables $b_2$ and $b_3$ with linear expression $l_1$. Replacing $b_2$ in $S_1^1$ with $b_2 + b_3 - b_4$ results in $\frac{1}{2} b_1 + \frac{1}{2} b_3 - \frac{1}{2} b_4$, which is a new solution to $x_1$. We name it as $S_2^1$. Similarly, replacing $b_1$ in $S_1^1$ with $b_2 - b_3 + b_4$ produces $\frac{1}{2} b_1 + \frac{1}{2} b_3 - \frac{1}{2} b_4$. Then, we use $l_2$ to replace $b_1$ and $b_2$ in $S_1^1$. Both of them result in $\frac{1}{2} b_1 + \frac{1}{2} b_3 - \frac{1}{2} b_4$, 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 $b_1 - b_3 - b_4 = 0$ to replace $b_2$, we get a linear expression $b_1 - \frac{1}{2} b_2 + \frac{3}{2} b_3 - b_4 + \frac{1}{2} b_6$. Since the corresponding row vectors $\{a_1, a_2, a_3, a_4, a_5\}$ are not linearly independent, we do not take it as a solution to $x_1$. 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.

### 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 $G_B = (U, V, E)$, 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 $G_B^e = (U, V, E)$ 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 one.

<table>
<thead>
<tr>
<th>TABLE 2</th>
<th>Solutions to Variable $x_1$ in Linear System and Corresponding Solutions to Link $e_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solution to $x_1$</td>
<td>Solution to $e_1$</td>
</tr>
<tr>
<td>$\frac{1}{2} b_1 + \frac{1}{2} b_2 - \frac{1}{2} b_4$</td>
<td>${p_1, p_2, p_3}$</td>
</tr>
<tr>
<td>$\frac{1}{2} b_1 + \frac{1}{2} b_3 - \frac{1}{2} b_5$</td>
<td>${p_1, p_2, p_3}$</td>
</tr>
<tr>
<td>$\frac{1}{2} b_2 + \frac{1}{2} b_3 - \frac{1}{2} b_6$</td>
<td>${p_1, p_2, p_3}$</td>
</tr>
<tr>
<td>$b_3 + \frac{1}{2} b_4 - \frac{1}{2} b_5$</td>
<td>${p_3, p_4, p_5, p_6}$</td>
</tr>
<tr>
<td>$b_1 - \frac{1}{2} b_4 - \frac{1}{2} b_5 + \frac{1}{2} b_6$</td>
<td>${p_1, p_4, p_5, p_6}$</td>
</tr>
<tr>
<td>$b_2 - \frac{1}{2} b_4 + \frac{1}{2} b_5 - \frac{1}{2} b_6$</td>
<td>${p_2, p_4, p_5, p_6}$</td>
</tr>
</tbody>
</table>
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:

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 $u_j^1 \in U_1$ represents a set of probing paths and vertex $u_j^2 \in U_2$ represents a single probing path. Consider the number of probing paths represented by a vertex in $U_1$ and $U_2$, we have $|u_j^1| \geq 1$ and $|u_j^2| = 1$. By setting $|u_j^1|$ to 1 for each vertex $u_j^1 \in U_1$, 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 $\alpha$ to control how many solutions are used for an identifiable target link. The computational complexity of the algorithm directly relates to $\alpha$. A smaller $\alpha$ makes the algorithm faster. On the other hand, a larger $\alpha$ 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.

![An illustration of the extended bipartite model.](image)

**Algorithm 2. PathSelection**

**Input:** The linear system $LS$ and parameter $\alpha$

**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** $\alpha$ 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. **while** $V_1$ has uncovered vertices **do**
9. **Select** an uncovered vertex $u_m^1$ from $U_1$ with the largest $\frac{\text{numNewVertices}^1}{\text{numNewPaths}}$, where $\text{numNewVertices}$ is the number of uncovered vertex in $V_1$ connected to $u_m^1$, and $\text{numNewPaths}$ 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**
probing paths can uniquely determine multiple identifiable target links (line 17-25). When all identifiable target links are covered, this process continues until all identifiable target links are considered. When selecting such a vertex $v_i^1$ from $U_1$, 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 $U_1$ 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 $\frac{\text{numNewVertices}}{\text{numNewPaths}}$ in each round as shown in line 19. 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.

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 $P_r$ can uniquely determine all identifiable target links and cover all unidentifiable target links. Thus, we need to prove that removing replaceable probing paths from $P_r$ does not affect the correctness.

Suppose set $P_r$ contains probing paths $p_1, \ldots, p_s$, in which $p_{s+1}, \ldots, p_r$ are replaceable and removed from set $P_r$. Consider a removed probing path $p_k$. In the dependency matrix, row vector $a_k$ can be linearly expressed by row vectors $a_1, \ldots, a_s$. Accordingly, variable $b_k$ is a linear combination of $b_1, \ldots, b_r$ as shown in (12).

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

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 $p_{s+1}, \ldots, p_r$. Hence, the linear system variable $x_j$ can be linearly expressed by variables $b_{s+1}, \ldots, b_r$ as shown in (13).

$$x_j = \sum_{i=1}^{s} d_i b_i + \sum_{i=s+1}^{r} c_i b_i + \sum_{i=s+1}^{r} d_i b_i.$$  

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

$$x_j = \sum_{i=1}^{s} d_i b_i + \sum_{i=s+1}^{r} d_i b_i + \sum_{i=s+1}^{r} c_i b_i + \sum_{i=s+1}^{r} d_i b_i.$$  

Since $r + 1 \leq k \leq s$, the above equation is equivalent to (15). It shows that $x_j$ is a linear combination of $b_{s+1}, b_{s+2}, \ldots, b_r$, which means that link $e_k$ can be uniquely determined after removing the probing path $p_k$.

$$x_j = \sum_{i=1}^{s} d_i b_i + \sum_{i=s+1}^{r} d_i b_i + \sum_{i=s+1}^{r} c_i b_i + \sum_{i=s+1}^{r} d_i b_i.$$  

When removing probing path $p_k$, variable $b_k$ in (14) is replaced by a linear combination of $b_{s+1}, \ldots, b_r$. We can remove all replaceable probing paths from set $P_r$ with the same method. Finally, $x_j$ becomes a linear combination of $b_{s+1}, \ldots, b_r$. It shows that $e_j$ can be uniquely determined by $p_{s+1}, \ldots, p_r$.

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 $a_k$ can be linearly expressed by row vectors $a_1, \ldots, a_s$, elements $a_{1,u}, \ldots, a_{s,u}$ cannot be all 0, which means that at least one of $a_{1,u}, \ldots, a_{s,u}$ is 1. That is, at least one probing path among $p_1, \ldots, p_r$ can cover link $e_u$. Therefore, removing $p_k$ does not affect covering $e_u$. 


In conclusion, after removing redundancy, probing paths in set \( P \) 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 \( P \) before removing its replaceable probing paths, we have \( P \subseteq \hat{P} \). Similar to \( P \), set \( \hat{P} \) corresponds to a linear system \( LS \), in the form of (2). The dependency matrix \( A \) of \( LS \) is formed by some row vectors of the original dependency matrix \( A \). Therefore, the row rank of \( A \) is no larger than that of \( A \).

After removing all replaceable probing paths, the probing paths left in \( \hat{P} \) correspond to another linear system \( LS' \). Its dependency matrix \( A' \) is formed by the maximal independent set of row vectors of \( A \). Therefore, the rank of \( A' \) is the same as that of \( A \), which is no larger than the row 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 \( e_3 \) 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 \( p_1, p_5, \) and \( p_6 \) that traverse unidentifiable target link \( e_3 \). Vertex \( e_3 \) 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 \( e_1 \). 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 \( \frac{\text{sum New Vertices}}{\text{sum New Probes}} \). 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 percent 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 percent of routers as probers. The third is parameter \( \alpha \) in the algorithm PathSelection. We set it to 1, 10, 100, and 1,000. The path selection algorithm does not specify how to select \( \alpha \) solutions; thus we randomly select \( \alpha \) solutions in the evaluation. Different strategies for selecting \( \alpha \) 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.

<table>
<thead>
<tr>
<th>Topology</th>
<th>AS209</th>
<th>AS701</th>
<th>AS2914</th>
<th>AS3320</th>
<th>AS3356</th>
<th>AS3549</th>
<th>AS3561</th>
<th>AS4323</th>
<th>AS7018</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>58</td>
<td>83</td>
<td>70</td>
<td>70</td>
<td>63</td>
<td>61</td>
<td>92</td>
<td>51</td>
<td>115</td>
</tr>
<tr>
<td>Links</td>
<td>108</td>
<td>219</td>
<td>111</td>
<td>355</td>
<td>285</td>
<td>486</td>
<td>329</td>
<td>161</td>
<td>148</td>
</tr>
</tbody>
</table>
2. Cost per identifiable target link: The algorithm Path-Selection 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 section, 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 percent 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 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 percent of links are identifiable if using 30 percent of nodes as probers. In particular, this percentage is as high as 85 percent 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.

4.3 Overall Probing Cost

Next, we evaluate the overall probing cost. Fig. 7 shows the overall probing cost when 40 percent of nodes are probers, which has two features. First, increasing parameter $\alpha$, 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 $\alpha$ to achieve good performance. For a large-scale network, we can use a small $\alpha$ to significantly reduce the running time but sacrifice a little bit of performance. Second, as the percentage of target links increases, more and more nonredundant 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 percent, the overall probing cost under different $\alpha$ becomes similar. When 20 and 60 percent 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 $\alpha = 1,000$, when 20, 40, and 60 percent 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 percent, 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 $\alpha$ probers, we have $\frac{n(n-1)}{2}$ usable probing paths. Hence, we have much more usable probing paths when the percentage of probers increases. As a result, our algorithm can pick out better probing paths to reduce this amortized cost. When $\alpha$ 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 $3(\text{percentages of probers}) \times 9(\text{topologies}) \times 4(\alpha) \times 100(\text{runs}) = 10,800$ runs of simulations, there is not a single case that the SelectPath algorithm outperforms our algorithm, which is consistent with the theoretical result. Tables 4, 5, and 6 show the average overall probing cost of our algorithm with different $\alpha$ and the SelectPath algorithm when 20, 40, and 60 percent of nodes are probers. The simulation result indicates that our algorithm is better than the SelectPath algorithm.

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. 2b is a simple example to show this fact. There are three identifiable links $e_1$, $e_2$, and $e_3$. The maximal independent set of row vectors contains four 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.

![Fig. 8. The cost per identifiable target link of the path selection algorithm with $\alpha = 1.000$. The lower bound is 1.](image-url)

**Table 4** Comparison of Overall Probing Cost of Our Algorithm and the SelectPath Algorithm (Percentage of Probers: 20 Percent)

<table>
<thead>
<tr>
<th>Topology</th>
<th>1</th>
<th>10</th>
<th>100</th>
<th>1000</th>
<th>SelectPath</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS209</td>
<td>23.9</td>
<td>23.7</td>
<td>23.4</td>
<td>23.3</td>
<td>25.4</td>
</tr>
<tr>
<td>AS701</td>
<td>50.6</td>
<td>50.5</td>
<td>50.4</td>
<td>50.4</td>
<td>51.6</td>
</tr>
<tr>
<td>AS2914</td>
<td>27.9</td>
<td>27.6</td>
<td>27.2</td>
<td>27.1</td>
<td>29.9</td>
</tr>
<tr>
<td>AS330</td>
<td>48.9</td>
<td>48.8</td>
<td>48.8</td>
<td>48.8</td>
<td>49.3</td>
</tr>
<tr>
<td>AS356</td>
<td>47.9</td>
<td>47.6</td>
<td>47.2</td>
<td>47.2</td>
<td>48.8</td>
</tr>
<tr>
<td>AS3549</td>
<td>53.2</td>
<td>53.0</td>
<td>53.0</td>
<td>53.0</td>
<td>53.8</td>
</tr>
<tr>
<td>AS3561</td>
<td>58.0</td>
<td>58.0</td>
<td>57.8</td>
<td>57.7</td>
<td>59.8</td>
</tr>
<tr>
<td>AS4323</td>
<td>26.8</td>
<td>26.8</td>
<td>26.7</td>
<td>26.7</td>
<td>27.1</td>
</tr>
<tr>
<td>AS7018</td>
<td>42.8</td>
<td>42.7</td>
<td>42.6</td>
<td>42.6</td>
<td>43.4</td>
</tr>
</tbody>
</table>
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], [17], [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 for 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 and Thiran [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.

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


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 working toward the PhD degree 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, and the MS and PhD degrees in computer science from the Ohio State University, 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 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 US National Science Foundation (NSF) CAREER award in 2001. He is a fellow of the IEEE.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/publications/dlib.