Minimizing Probing Cost and Achieving Identifiability in Network Link Monitoring

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

Abstract

Continuously monitoring the link performance is important to network diagnosis. Recently, active probes sent between end systems are widely used to monitor the link performance. In this paper, we address the problem of minimizing the probing cost and achieving identifiability in link monitoring. Given a set of links to monitor, our objective is to select as few probing paths as possible to cover all of them, and the selected probing paths can uniquely identify all identifiable links being monitored. We propose an algorithm based on the linear system model to find out all sets of probing paths that can uniquely identify an identifiable link. We extend the bipartite model to reflect the relation between a set of probing paths and the link that can be uniquely identified. Through the extended bipartite model, our optimization problem is transformed into the classic set cover problem, which is NP-hard. Therefore, we propose a heuristic based algorithm to greedily select the probing paths. Our method eliminates two types of redundant probing paths, i.e., those that can be replaced by others and those that cannot be used 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.

1. Introduction

With the rapid growth of the Internet, efficiently monitoring the performance and robustness of the network becomes more and more important. The service providers need to keep tracking of their networks to ensure that their services satisfy the commitments specified in the service level agreements (SLAs). Additionally, applications sensitive to network performance such as 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 active probe (end-to-end probe for short) has received much attention and has been widely used in network monitoring, such as [1]–[7]. Compared with reply-based probe such as `ping` and `traceroute`, end-to-end probe does not have the problem of being ignored by the intermediate routers [8] and being blocked by the firewall [9]. Furthermore, different from Simple Network Management Protocol (SNMP)-based polling [10], end-to-end probe based approaches do not need to run agents on routers.

One of the major considerations in designing a probing scheme is the probing cost, which can be the number of selected probing paths [2], [11]–[14], the number of end systems used to send and receive probing packets [7], [11], [12], and the cost specified by the network component [6]. Lower probing cost means less negative effects on the normal data transmission and better scalability. Therefore, many existing works [2], [6], [7], [11]–[13] focus on minimizing the probing cost for network monitoring.

This is usually achieved by carefully selecting the probing paths. However, to monitor the performance of a link such as delay and loss rate, selecting probing paths to cover the link may not be sufficient. An end-to-end probe measures the performance of the whole probing path, which may consist of multiple links. In order to uniquely infer the performance of each 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 end-to-end probes [15].

There are two kinds of network links. If the performance of a link can be uniquely inferred from a set of end-to-end probes, it is called *uniquely identifiable* or *identifiable* for short; otherwise, it is called *unidentifiable*. If the probing paths are not properly selected, the performance of a link cannot be inferred even if it is an identifiable link. For example, 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 probing path traversing e_4 also traverses e_5. Selecting the path $p_1(s_1, s_2)$, $p_2(s_1, s_3)$, and $p_4(s_2, s_3)$ can uniquely infer the performance of links e_1, e_2, and e_3. However, only choosing $p_1(s_1, s_2)$ and $p_2(s_1, s_3)$ cannot uniquely infer the performance of any link.

In this paper, we aim at minimizing the probing cost and achieving identifiability in link monitoring. More specifically, given a set of links called *target links*[1] to monitor, our objective is to select as few probing paths as possible that can cover all the target links and uniquely infer the performance of the identifiable target links under monitoring. There are three challenges to achieve our objective.

This work was supported in part by DoD/DTRA under grant DTRA01-03-D-0010-0015.

1. Target links can be links at critical topological location, and their performance usually affects a large area of the network [16].

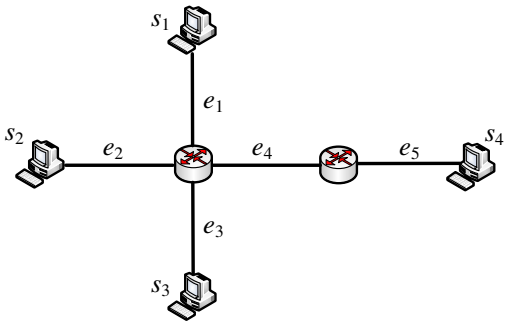


Figure 1. An example of network link monitoring with six probing paths $p_1(s_1, s_2)$, $p_2(s_1, s_3)$, $p_3(s_1, s_4)$, $p_4(s_2, s_3)$, $p_5(s_2, s_4)$, and $p_6(s_3, s_4)$, where $p_i(s_j, s_k)$ denotes the probing path p_i between the end system s_j and s_k.

First, a target link may be unidentifiable. Using multiple probing paths to monitor an unidentifiable link only wastes bandwidth. Therefore, it is necessary to differentiate an identifiable link from an unidentifiable one. For an unidentifiable target link, we only need to select a probing path traversing it. Second, different probing paths have different contributions to achieving identifiability, and we need to determine the most useful probing paths. Third, a probing path may be able to be replaced by other probing paths, and we should avoid selecting redundant probing paths.

The basic idea of our approach is to select probing paths that are the most useful for covering the target links and achieving identifiability. We adopt a linear system to model the relation between probing paths and links. We design a method based on the matrix decomposition and linear replacement to find out irreducible sets of probing paths that can uniquely determine each identifiable target link. Moreover, we extend the bipartite model which is commonly used for modeling the relation between a single probing path and link to reflect the relation between a set of probing paths and a link. Through the bipartite model, our optimization problem is transformed into the classic set cover problem, which is NP-hard. Therefore, we propose a heuristic based algorithm to greedily select the probing paths.

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

2. Preliminaries

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

2.1. Problem Description

Similar to prior work on network monitoring [2], [11], [17]–[20], we model the network as a connected undirected graph $G(V, E)$ for simplicity, where V is the set of nodes (i.e. routers) and $E = \{e_i | 1 \leq i \leq m\}$ is the set of edges (i.e. communication links between routers). Note that the proposed technique also works for asymmetric links. In the rest of the paper, we use edge and link interchangeably. There are many end systems connecting to the network, and each of them can send and receive probing packets. The probing packet from one end system to another traverses the routing path between them. We call such an end-to-end path a *probing path*. If a probing path traverses a link, the path can *cover* the link. 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_t target links $E_t \subseteq E$. For simplicity, we assume $E_t = \{e_i | 1 \leq i \leq m_t\}$, which equals to relabeling the links in G. With the router-level topology G, we know which links can be covered by a probing path.

Generally speaking, network monitoring by means of end-to-end probe includes two steps. The first step is to select a set of probing paths. Then, end systems on the chosen probing paths issue probing packets and send the result to the central Network Operations Center (NOC). In this study, 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 cost is defined as the number of probing paths selected for 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 as few probing paths as possible from P, such that all links in E_t are covered and the performance of each identifiable link in E_t can be uniquely inferred.

2.2. Linear System Model

For the link performance satisfying the additive metric, e.g. delay, the relation between the probing paths and links can be naturally modeled as a linear system $LS = \{ls_i | 1 \leq i \leq n\}$, in which ls_i is the ith linear equation $\sum_{j=0}^{m} a_{ij} x_j = b_i$. The variable x_j denotes the performance of the link e_j and the variable b_i is the performance of the probing path p_i. The value of a_{ij} is 1 if the probing path p_i traverses the link e_j; otherwise it is 0. The linear equation ls_i means that the performance of p_i is an addition of the performance of all links on p_i. The linear system LS can also be written as Eq. (1), where $A = (\mathbf{a}_1, \cdots, \mathbf{a}_n)^T$ is the coefficient matrix which is also called the *dependency matrix* [12].

$$A\mathbf{x} = \mathbf{b} \tag{1}$$

Take the example in Fig. 1 for instance, 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). We use delay as an example to explain the meaning of the linear

$$\begin{cases} x_1 + x_2 = b_1 \\ x_1 + x_3 = b_2 \\ x_1 + x_4 + x_5 = b_3 \\ x_2 + x_3 = b_4 \\ x_2 + x_4 + x_5 = b_5 \\ x_3 + x_4 + x_5 = b_6 \end{cases} \qquad A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

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

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

equation. For the probing path p_1 between the end system s_1 and s_2, the overall delay of the path is the sum of the delay on link e_1 and e_2. Correspondingly, in the first equation of the linear system in Fig. 2(a), the variable b_1 is equal to the sum of x_1 and x_2.

As mentioned in Section 1, a link can be either identifiable or unidentifiable, and a probing path may be replaced by other probing paths. A link e_i is identifiable if and only if the variable x_i in the linear system is solvable. If a row vector $\mathbf{a}_j$ of the dependency matrix A can be linearly expressed by some other row vectors, the probing path p_j can be replaced by the probing paths corresponding to the row vectors that linearly express $\mathbf{a}_j$. In Fig. 2(b), since the row vector $\mathbf{a}_5$ can be linearly expressed as $\mathbf{a}_5 = -\mathbf{a}_2 + \mathbf{a}_3 + \mathbf{a}_4$, the probing path p_2, p_3, and p_4 together can replace the probing path p_5. Due to the linear dependence among row vectors, there may be several linear equations to solve a solvable variable x_i; i.e., there are multiple sets of probing paths to uniquely determine an identifiable link. For an identifiable link e_i, we define its solution as follows.

Definition 2 (Solution to an identifiable link): A *solution* to an identifiable link e_i is a set of probing paths satisfying: (1) they can uniquely determine e_i, and (2) each probing path in the set cannot be replaced by other probing paths in the set.

With this definition, each probing path in a solution cannot be replaced by other probing paths in the same solution. This is consistent with our objective of minimizing the overall selected probing paths. Consider a set of probing paths that are able to uniquely determine e_i, if a probing path can be replaced by others in the set, after removing this probing path the set can still uniquely determine e_i. Hence, for determining e_i, containing such replaceable probing paths only increases the probing cost. Thus, a solution to an identifiable link e_i should not contain any replaceable probing path. This is the basis of our path selection algorithm.

The solution to a solvable variable x_i in the linear system does not necessarily match a solution to e_i. A solution to x_i is a linear expression $\sum_{j=1}^{n} c_{ij} b_j$. The variable in this linear expression may be replaced by other variables in the same expression. Actually, a solution to e_i corresponds to the solution to x_i satisfying that every variable b_j in it is not replaceable by other variables in the same solution. In the rest of the paper, when we say a solution to a solvable variable x_i, it means a solution whose variables are not replaceable by other variables in it. Since $\mathbf{a}_j\mathbf{x} = b_j$, a variable b_j is replaceable by other b_is if and only if $\mathbf{a}_j$ can be linearly expressed by the row vectors corresponding to these b_is. For example, a solution to the solvable variable x_1 in Fig. 2(a) is $\frac{b_1+b_2-b_4}{2}$. It contains three variables b_1, b_2, and b_4. Each of them is not replaceable by other two variables, because the corresponding row vectors $\mathbf{a}_1$, $\mathbf{a}_2$, and $\mathbf{a}_4$ are linearly independent. Accordingly, the probing path p_1, p_2, and p_4 together can uniquely determine the identifiable link e_1, and each of the three probing paths cannot be replaced by another two probing paths. Therefore, the set of the probing paths $\{p_1, p_2, p_4\}$ is a solution to e_1.

Table 1. Table of notations

Symbols	Meaning
G	network under study
m	number of links in G
e_i	the ith link in G
n	number of probing paths in G
p_i	the ith probing path in G
E_t	set of target links
m_t	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 variable x_i in LS
S^i_j	the jth solution to the variable x_i in LS
L	set of linear expressions, $L = \{l_i \mid 1 \leq i \leq t\}$
l_i	the linear expression for the ith row vector in A_N

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.

3. Path Selection Algorithm

A solution to an identifiable link is a set of probing paths that can uniquely determine this link. Since a probing path may be replaceable by others, an identifiable link may have multiple solutions and a probing path may be useful for uniquely determining multiple identifiable links. Consequently, different probing paths have different contributions to determining the identifiable links. In this section, we propose an algorithm to calculate all solutions to each identifiable target link. Among these solutions, we determine the contribution of each probing path to achieving identifiability and then choose the most useful probing paths.

3.1. Decomposition of Linear System

The simplest way to determine if a link is identifiable or not is to solve the linear system LS. Many techniques in the linear algebra, such as Gaussian elimination, can achieve this. The linear dependence in row vectors of the dependency matrix only affects how many solutions an identifiable link has, but does not affect whether a link is identifiable or not.

Therefore, the first step for calculating all solutions to each identifiable target link is to decompose the linear system so as to determine whether a target link is identifiable and produce a solution to each of identifiable target links.

We divide the row vectors of the dependency matrix into two groups. As a result, the dependency matrix A is decomposed into the form shown as Eq. (2). In this decomposition, the matrix A_R contains r linearly independent row vectors of A, and A_N contains other $n-r$ row vectors. Each row vector of A_N can be linearly expressed by the 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 $A_N = (\mathbf{a}_{r+1}, \cdots, \mathbf{a}_n)^T$.

$$A = \begin{pmatrix} A_R \\ A_N \end{pmatrix} \tag{2}$$

Accordingly, the linear system LS can be expressed as Eq. (3), in which the vector $\mathbf{b}$ is partitioned into two vectors $\mathbf{b}_R = (b_1, \cdots, b_r)^T$ and $\mathbf{b}_N = (b_{r+1}, \cdots, b_n)^T$. Since each row vector of A_N is a linear combination of row vectors of A_R, a solvable/unsovlable variable in the linear system $A_R\mathbf{x} = \mathbf{b}_R$ is also solvable/unsolvable in the linear system in Eq. (1).

$$\begin{pmatrix} A_R \\ A_N \end{pmatrix} \mathbf{x} = \begin{pmatrix} \mathbf{b}_R \\ \mathbf{b}_N \end{pmatrix} \tag{3}$$

To construct the decomposition of the matrix A, we use the algorithm introduced in [13], which is based on standard rank-revealing decomposition techniques [21]. It is possible that a dependency matrix has multiple decompositions of Eq. (2). In the next part, we will see that our algorithm for calculating all solutions to an identifiable target link does not have any requirement on the decomposition of the linear system. Therefore, we can use any tie breaking strategy to choose a decomposition.

By solving the linear system $A_R\mathbf{x} = \mathbf{b}_R$, we can determine whether a target link is identifiable or not and calculate a solution S_1^i to the solvable variable x_i. The solution calculated from the linear system $A_R\mathbf{x} = \mathbf{b}_R$ is referred to as the *base solution*. There is only one base solution to each solvable variable x_i because the row vectors of A_R are linearly independent. Since the row vector of A_N can be linearly expressed by the row vectors of A_R, we calculate the linear expression for each row vector of A_R as shown in Eq. (4), where $i = 1, \cdots, n-r$.

$$\sum_{j=1}^{r} c_{ij}\mathbf{a}_j + \mathbf{a}_{r+i} = 0 \tag{4}$$

Multiplying vector $\mathbf{x}$ to both sides of Eq. (4) results in $\sum_{j=1}^{r} c_{ij}\mathbf{a}_j\mathbf{x} + \mathbf{a}_{r+i}\mathbf{x} = 0$. Since $\mathbf{a}_i\mathbf{x} = b_i$ exists for $1 \leq i \leq n$, we can get the linear expression in Eq. (5). We use $L = \{l_i | 1 \leq i \leq n-r\}$ to denote the set of these linear expressions, in which l_i is the expression containing b_{r+i}.

$$\sum_{j=1}^{r} c_{ij}b_j + b_{r+i} = 0 \tag{5}$$

Take the example in Fig. 2(b) for instance, a decomposition of the matrix A 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 $\mathbf{b}$ is $\mathbf{b}_R = (b_1, b_2, b_3, b_4)^T$ and $\mathbf{b}_N = (b_5, b_6)^T$. By solving the linear system $A_R\mathbf{x} = \mathbf{b}_R$, we discover that e_1, e_2, and e_3 are identifiable but e_4 and e_5 are unidentifiable. The base solution to x_1 is $x_1 = \frac{b_1+b_2-b_4}{2}$. It corresponds to a solution to e_1, i.e. the set $\{p_1, p_2, p_4\}$. The set L contains two linear expressions, in which l_1 is $b_2 - b_3 - b_4 + b_5 = 0$ and l_2 is $b_1 - b_3 - b_4 + b_6 = 0$.

3.2. Calculation of Solutions

By decomposing the linear system LS, we find out all identifiable target links, obtain the base solution to each solvable variable, and calculate the linear expression for $b_{r+1}, \cdots, b_n$. In this subsection, we propose an algorithm to calculate all solutions to a solvable variable, through which we can obtain all solutions to the corresponding identifiable link.

The basic idea of calculating all solutions to a solvable variable x_i is to replace the variables in the solutions with the linear expression in the set L. Initially, we have only the base solution S_1^i obtained from the 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. Then we check if this linear expression has any variable that can be replaced by other variable in it. If not, this linear expression is a solution to x_i. We apply similar linear replacements to all solutions until no new solution can be produced. Later we will see that this algorithm can find all solutions to a solvable variable x_i.

The algorithm for calculating all solutions to a solvable variable x_i is shown in Algorithm 1. The input is the base solution S_1^i obtained from $A_R\mathbf{x} = \mathbf{b}_R$ and the set L of the linear expressions. We use a queue to store the solutions. As shown in line 5–14, each time we take out a solution from the queue and apply all possible linear replacements to it. The 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 S_t^i does not have any replaceable variable, it is a solution to x_i. For a solution not in Q and S^i, it is a new solution. Thus we put it into the queue. After applying all linear replacements to the current solution S_c^i, we put it into the solution set S^i. As a result, at any moment the set S^i contains all solutions that have been applied linear replacements, and the queue Q contains all solutions that will be applied linear replacements. When the queue Q becomes empty, the algorithm stops and all solutions to x_i are in the set S^i.

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

Proof: Suppose there is a solution S_q^i to the solvable variable x_i that is not found out by the algorithm. This

Algorithm 1 SolutionCalculation

Input: The base solution S_1^i, and the set of the linear expressions L
Output: A set S^i containing all solutions to x_i
Procedure:
INIT(Q) //Initialize a queue Q
$S^i = \emptyset$
ENQUEUE(Q, S_1^i)
while $Q \neq \emptyset$ **do**
  S_c^i = DEQUEUE(Q)
  for each linear expression $l_j \in L$ **do**
    for each b_k in both S_c^i and l_j **do**
      $S_t^i \leftarrow$ replace b_k in S_c^i with l_j
      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**
        ENQUEUE(Q, S_t^i)
      end if
    end for
  end for
  $S^i = S^i \cup S_c^i$
end while

solution has the following form.

$$x_i = \sum_{j=1}^{n} d_{qj} b_j \tag{6}$$

From the algorithm `SolutionCalculation` we can see that all solutions in S^i is a linear combinations of S_1^i and the linear expressions in L. Therefore, this missed solution S_q^i cannot be linearly expressed by S_1^i and linear expressions in L. Consider the linear system in Eq. (7) that includes S_1^i, S_q^i, and all linear expressions in L.

$$\begin{cases} \sum_{j=1}^{r} d_{1j} b_j - x_i = 0 \\ \sum_{j=1}^{n} d_{qj} b_j - x_i = 0 \\ \sum_{j=1}^{r} c_{1j} b_j + b_{r+1} = 0 \\ \vdots \\ \sum_{j=1}^{r} c_{n-r,j} b_j + b_n = 0 \end{cases} \tag{7}$$

The coefficient matrix of this linear system is as Eq. (8).

$$\begin{array}{c} \begin{array}{ccccccc} b_1 & \cdots & b_r & b_{r+1} & \cdots & b_n & x_i \end{array} \\ \begin{pmatrix} d_{11} & \cdots & d_{1r} & 0 & \cdots & 0 & -1 \\ d_{q1} & \cdots & d_{qr} & d_{q,r+1} & \cdots & d_{qn} & -1 \\ c_{11} & \cdots & c_{1r} & 1 & \cdots & 0 & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots \\ c_{n-r,1} & \cdots & c_{n-r,r} & 0 & \cdots & 1 & 0 \end{pmatrix} \end{array} \tag{8}$$

By subtracting $d_{q,r+k}$ times the $(2+k)$th row from the second row ($k = 1, \cdots, n-r$), the matrix turns into the form as in Eq. (9), where $d'_{qj} = d_{qj} - \sum_{k=1}^{n-r} d_{q,r+k} c_{kj}$ for $j = 1, \cdots, r$.

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

Since S_q^i cannot be linearly expressed by S_1^i and linear expressions in L, the first two row vectors in Eq. (9) are linearly independent. It means that $d_{1j} \neq d_{qj}$ for $j = 1, \cdots, r$. Therefore, there are two solutions to x_i in the linear system $A_R \mathbf{x} = \mathbf{b}_R$, i.e. $\sum_{j=1}^{r} d_{1j} b_j$ and $\sum_{j=1}^{r} d'_{qj} b_j$. It contradicts with the fact that $A_R \mathbf{x} = \mathbf{b}_R$ has only one solution to each solvable variable as mentioned before. Therefore, no such S_q^i exists. In conclusion, `SolutionCalculation` can find out all solutions to the solvable variable x_i. □

As mentioned in Section 2.2, each solution to a solvable variable x_i corresponds to a solution to the identifiable link e_i. For each variable b_j contained in a solution to x_i, the corresponding solution to e_i contains the probing path p_j. Therefore, after calculating all solutions to x_i, we can easily obtain all solutions to e_i.

To make it more clear, we use an example to show how to calculate all solutions to the identifiable link e_1 in Fig. 1. Initially, we have a solution S_1^1 that is $\frac{b_1+b_2-b_4}{2}$. The linear expression set L contains two 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 solution S_1^1 has two common variables b_2 and b_4 with the linear expression l_1. Replacing b_2 in S_1^1 with $b_3 + b_4 - b_5$ results in $\frac{b_1+b_3-b_5}{2}$, which is a new solution. We name it as S_2^1. Similarly, replacing b_4 in S_1^1 with $b_2 - b_3 + b_5$ produces $\frac{b_1+b_3-b_5}{2}$. Then we use l_2 to replace b_1 and b_4 in S_1^1, both resulting in $\frac{b_2+b_3-b_6}{2}$ named as S_3^1. Until now, we have already applied all possible linear replacement to S_1^1. Next, we apply linear replacements toward S_2^1 and S_3^1. The generated new solutions are put into the queue and the linear replacement continues until the queue is empty. When applying l_1 to the solution $\frac{2b_1-b_4-b_5+b_6}{2}$ to replace b_4, we get a linear expression $\frac{2b_1-b_2+b_3-2b_5+b_6}{2}$. 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 $\frac{b_1+b_2-b_4}{2}$, $\frac{b_1+b_3-b_5}{2}$, $\frac{b_2+b_3-b_6}{2}$, $\frac{2b_3+b_4-b_5-b_6}{2}$, $\frac{2b_1-b_4-b_5+b_6}{2}$, and $\frac{2b_2-b_4+b_5-b_6}{2}$. Accordingly, there are six solutions to the link e_1: $\{p_1, p_2, p_4\}$, $\{p_1, p_3, p_5\}$, $\{p_2, p_3, p_6\}$, $\{p_3, p_4, p_5, p_6\}$, $\{p_1, p_4, p_5, p_6\}$, and $\{p_2, p_4, p_5, p_6\}$.

3.3. Extended Bipartite Model

As in the literature [5], the relation between the probing paths and the 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. A vertex in the set U and V denotes a probing path and a target link, respectively. If a probing path p_i traverses a target link e_j, there is an edge between the vertex $u_i \in U$ and $v_j \in V$. Thus the edge reflects the coverage relation between the probing path and the target link.

However, this bipartite model cannot deal with our path selection problem, because it is unable to reflect the relation between an identifiable target link and its solutions. Therefore, we propose an extended bipartite model. In our bipartite

model $G'_B = (U, V, E)$, the vertex set U has two subset U_1 and U_2, and the vertex set V also has two subsets V_1 and V_2, shown in Fig. 3. A vertex in the set V_1 and V_2 represents an identifiable and unidentifiable target link, respectively. Each vertex in U_1 denotes a solution to an identifiable target link. Hence it corresponds to a set of probing paths. It is possible that a probing path is not in any solution. We use a vertex in U_2 to denote each of such probing paths. There are three kinds of edges in the extended bipartite mode as follows.

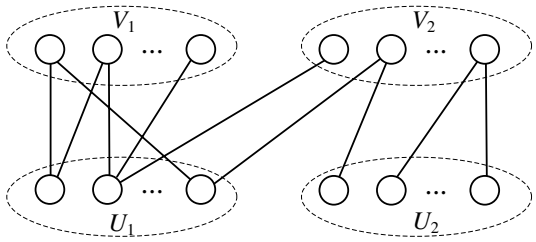


Figure 3. An illustration of the extended bipartite model.

1) The edge between V_1 and U_1. For an identifiable target link denoted by the vertex $v_i^1 \in V_1$, if the solution represented by the vertex $u_j^1 \in U_1$ is a solution to this identifiable link, there is an edge connecting vertex v_i^1 and u_j^1. Hence the edge reflects the identifiability relation 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 a probing path in the solution represented by vertex $u_j^1 \in U_1$ covers it, 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 the vertex $v_i^2 \in V_2$, if a probing path represented by the vertex $u_j^2 \in U_2$ traverses it, there is an edge connecting 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 set V_1 and U_2. For identifiable target links, we need to select probing paths to uniquely determine them, which equals to the problem of choosing vertices from U_1 such that their neighbors include all vertices in V_1. The edges between V_1 and U_1 are sufficient to model the problem. Thus, we do not add any edge between an identifiable target link and a single probing path.

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

We can construct the bipartite graph G'_B in the following three steps. First, we build sets V_1, V_2, and U_1, and let U_2 be empty. Then we construct the edges between U_1 and V_1, and the edges between U_1 and V_2. Finally, for each probing path that is not in the set U_1, we add a vertex to U_2 to denote it and construct the edges between it and the vertex in V_2.

3.4. 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. This problem is NP-hard, because it can be easily transformed into the set cover problem. Thus we design a heuristic based algorithm to solve it.

The basic idea of the path selection algorithm is to greedily select probing paths until all identifiable target links can be uniquely determined and all unidentifiable target links are covered. For each identifiable target link, we select a set of probing paths based on its solutions. Currently, we give the identifiable target link a priority higher than the unidentifiable target link, because uniquely determining an identifiable link is more complex and needs more probing paths than covering an unidentifiable link. Hence, the algorithm first deals with the identifiable target link and then the unidentifiable target link. An important point is that the selected probing paths may contain replaceable probing paths. Therefore, after finishing the path selection, the algorithm removes all replaceable ones from the selected probing paths. The linear system based method for finding out all replaceable probing paths is introduced in Section 2.2.

For a large scale network, there can be thousands of probing paths. Accordingly, an identifiable target link may have a large number of solutions. In order to enhance the scalability of our method, we introduce a parameter α to control how many solutions to an identifiable target link are used in path selection. The path selection algorithm is shown as Algorithm 2. It takes the linear system LS and the 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. Firstly, the algorithm calculates solutions to each identifiable target link with the algorithm `SolutionCalculation`, and then select α solutions for each identifiable target link. Next, the algorithm builds the extended bipartite graph $G'_B = (U, V, E)$ based on the linear system LS and selected solutions. After building the extended bipartite graph, the algorithm starts to use a heuristic to select probing paths. From line 8 to 16, the algorithm greedily selects vertices from U_1 until all vertices in V_1 are covered. Each time, the algorithm chooses an uncovered vertex with the maximal value of $\frac{numNewVertices1}{numNewPaths}$. Intuitively, it means that we like to add as few probing paths as possible and uniquely determine as many identifiable target links as possible. When all identifiable target links are handled, the unselected probing paths are contained in

both U_1 and U_2. Thus the algorithm selects vertices from $U_1 \cup U_2$ for the uncovered unidentifiable target links until all of them are covered. Finally, the algorithm removes all replaceable probing paths from the set.

Algorithm 2 PathSelection

Input: The linear system LS and the parameter α
Output: A set of probing paths P_s
Procedure:
1: **for** each identifiable target link **do**
2: calculate solutions to it by means of 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 the set P_s.

Theorem 2: The upper bound of the number of probing paths selected by the algorithm `PathSelection` is the row rank of the dependency matrix.

Proof: Consider the set P_s before removing the replaceable paths in it, we have $P_s \subseteq P$. For the linear system LS_s formed by the probing paths in P_s, the row rank of the coefficient matrix A_s is no larger than that of the dependency matrix A in Eq. (1). After removing all replaceable probing paths, the number of probing paths in P_s is equal to the row rank of A_s, which is unchanged after removing replaceable paths. Therefore, 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 the algorithm for selecting probing paths for the target link e_1 and e_4 in Fig. 1. In the example, we use all solutions for path selection. The corresponding bipartite model is shown in Fig. 4. The set U_1 has six vertices, each of which represents a solution to the identifiable link e_1. The set U_2 is empty, because all probing paths traversing the unidentifiable link e_4 are in the set U_1. The vertex e_4 in V_2 has no edge to the vertex $\{p_1, p_2, p_4\}$ in U_1 because these three paths do not traverse e_4. The algorithm first chooses a path set for e_1. Among all six vertices in U_1, the vertex $\{p_1, p_2, p_4\}$, $\{p_1, p_3, p_5\}$, 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 an unidentifiable link, the algorithm selects a path set from 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 $\{p_1, p_3, p_5\}$.

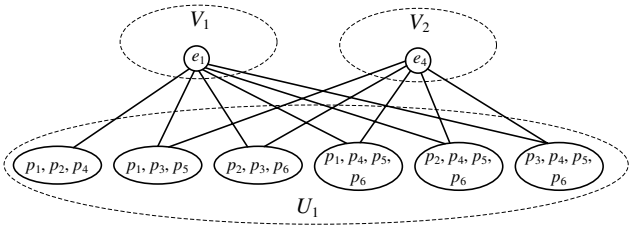


Figure 4. The extended bipartite model for selecting probing paths for the target link e_1 and e_4 in Fig. 1.

4. Performance Evaluations

In this section, we evaluate the performance of our algorithm that can select a minimal set of probing paths to cover a set of target links and uniquely determine all identifiable target links. We compare the performance of our method with others [13], and study how network topologies and the percentages of target links affect the performance.

4.1. Simulation Setup

The simulation is based on eight ISP topologies derived by the Rocketfuel project [22]. Table 2 summarizes these topologies. The first column shows the total number of nodes and the second column shows the number of leaf nodes for each topology. Similar to [14], we connect end systems to the leaf nodes to send and receive probing packets. All topologies adopt the shortest path routing calculated based on the link cost in the topologies. Since the link is symmetrical in these topologies, the routing path between two nodes is symmetrical. Thus the number of probing paths is $\frac{n_l(n_l-1)}{2}$, where n_l is the number of leaf nodes. Due to shortest path routing, each topology has some links that cannot be covered by the probing paths. We list the number of links that can be covered by probing paths in the fourth column. Among these links, some can be uniquely determined by probing paths. The last column shows the number of identifiable links and the percentage of links covered by the probing paths. In six topologies, more than

Table 2. Summary of Topologies Used in Simulation

Topology	Total Nodes	Leaf Nodes	Total Links	Covered Links	Identifiable Links (%)
AS1	42	18	55	42	28 (66.7%)
AS209	58	25	108	65	41 (63.1%)
AS701	83	35	219	106	89 (84.0%)
AS1299	31	14	44	29	11 (37.9%)
AS1668	53	30	64	53	36 (67.9%)
AS2548	42	16	58	38	20 (52.6%)
AS2828	40	27	45	40	33 (82.5%)
AS2914	70	23	111	61	29 (47.5%)

half of the links covered by probing paths are identifiable. In topology AS701, it is as high as 84%.

For each topology, we randomly select a certain percent of the link that can be covered by probing paths as the target link. The percentage of the target link ranges from 10% to 100%. We run each simulation for 20 times and take the statistical average as the result. For each test case, we run the algorithm `PathSelection` with $\alpha = 1, 10, 100, 1000, 2000$. In our path selection algorithm, we do not specify how to select α solutions, thus we randomly select solutions in the evaluation. Different strategies for selecting solutions to build extended bipartite graph are left for future work.

The major performance metric in our evaluation is the number of the selected probing paths, which is referred to as the *overall probing cost*. We also use two other metrics to show the amortized probing cost.

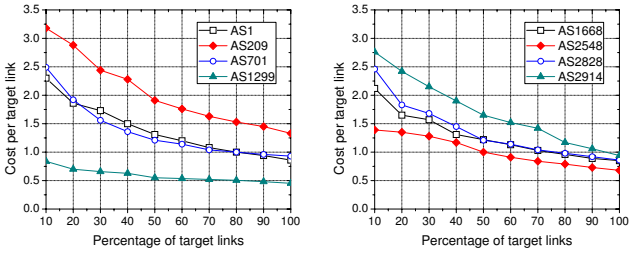
1) *Cost per target link*: It is the ratio of the overall probing cost to the number of target links. This metric quantifies the amortized cost on each target link.
2) *Overhead per identifiable target link*: The algorithm introduced in [11] can select a minimal set of probing paths to cover all target links. The difference between the overall probing cost and the number of probing paths for covering all target links can be considered as additional probing paths used for achieving identifiability. The overhead per identifiable target link is the ratio of such additional cost to the number of identifiable target links.

4.2. Overall Probing Cost

The overall probing cost of our path selection algorithm on eight topologies is shown in Fig. 5. On all these topologies, the overall probing cost of the algorithm with $\alpha = 1000$ is almost the same as that of the algorithm with $\alpha = 2000$ under each percentage of target links. Hence, in Fig. 5 we omit the case of $\alpha = 2000$. From the figure we can see that increasing α, namely using more solutions for path selection, can reduce the overall probing cost. 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 2. Therefore, when the percentage of target links reaches 100%, the overall probing cost under different α becomes similar.

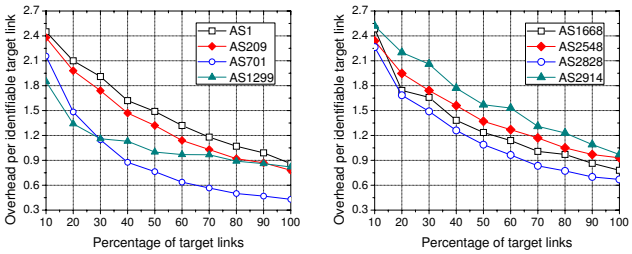
4.3. Amortized Probing Cost

In this part, we show the amortized probing cost of the path selection algorithm with $\alpha = 1000$. As shown in Fig. 6, the probing cost per target link decreases as the percentage of target links increases. The overhead per identifiable target link shown in Fig. 7 has similar trend. When the percentage of target links increases, the amortized overhead decreases. For both of them, when the percentage of target links reaches 100%, the amortized cost reduces to about 1/3 of the case when 10% links are target links. This indicates that our path selection algorithm can effectively choose the probing paths that are most useful for identifying multiple target links and eliminate redundant probing paths.



(a) AS1, AS209, AS701, AS1299 (b) AS1668, AS2548, AS2828, AS2914

Figure 6. The cost per target link of the path selection algorithm with $\alpha = 1000$.

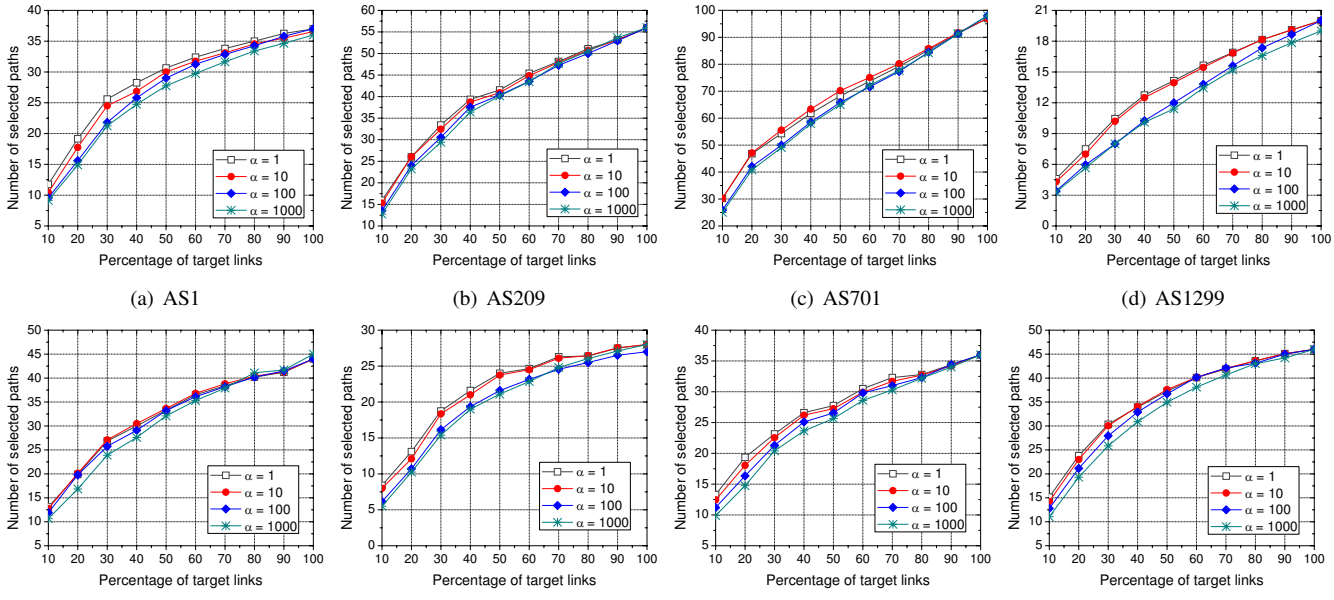


(a) AS1, AS209, AS701, AS1299 (b) AS1668, AS2548, AS2828, AS2914

Figure 7. The overhead per identifiable target link of the path selection algorithm with $\alpha = 1000$.

4.4. Comparisons

The closest work in this area is the SelectPath algorithm [13], which can select a minimal set of probing paths to uniquely determine all identifiable links in the network. However, the SelectPath 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 or equal to theirs in this subsection.



(a) AS1 (b) AS209 (c) AS701 (d) AS1299

(e) AS1668 (f) AS2548 (g) AS2828 (h) AS2914

Figure 5. The overall probing cost of the path selection algorithm with $\alpha = 1, 10, 100, 1000$.

Table 3. Largest Overall Probing Cost of The PathSelection Algorithm with Different α and Overall Probing Cost of The SelectPath Algorithm

Topology	α					SelectPath [13]
	1	10	100	1000	2000	
AS1	37	37	37	36	36	37
AS209	56	56	56	56	56	57
AS701	97	97	98	98	98	98
AS1299	20	20	20	19	19	22
AS1668	44	44	44	45	45	46
AS2548	28	28	27	28	29	29
AS2828	36	36	36	36	36	37
AS2914	46	46	46	46	46	49

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 [13], the overall probing cost of the SelectPath algorithm is equal to the row rank of the dependency matrix. According to Theorem 2, 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 no worse than the SelectPath algorithm.

For each topology and α, we choose the largest overall probing cost among 20 simulation runs. Table 3 shows the largest overall probing cost of our algorithm with different α and the overall probing cost of the SelectPath algorithm. On five topologies, our algorithm with each α outperforms the SelectPath algorithm. On the other three topologies AS1, AS701 and AS2548, the performance of our algorithm is equal to the SelectPath algorithm.

The SelectPath algorithm selects the probing paths corresponding to the basis of row vectors of the dependency matrix. Although the vectors in the basis are linearly independent, it does not mean there is no redundancy. Fig. 2(b) is a simple example to explain this fact. There are three identifiable links e_1, e_2, and e_3. The basis of the row vectors contains 4 linearly independent vectors. However, only three probing paths p_1, p_2, and p_4 are adequate to uniquely determining these three identifiable links. Adding any other probing paths does not contribute 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 the replaceable probing path but also the probing path 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 [2], [13], [14], and in scenarios with explicit operational requirement [7] and resource constraint [23]. Various kinds of probing costs and monitoring objectives have been studied. However, As pointed out in [24], many network inference problems are ill-posed, because the number of measurements are not sufficient to uniquely determine the result. Our work jointly addresses minimizing the probing cost and achieving identifiability.

Achieving identifiability in network link monitoring is also considered in [3], [11]–[13]. However, our work is quite different from them. Zhao *et al* [3] 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 [11] 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 [12] and [13] deal with the problem of selecting a minimal 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 or equal to theirs. Another major difference is the method we adopt. Their methods are based on eliminating the replaceable path; i.e., the linearly dependent row vector in the dependency matrix. We take a different approach, and only select the probing paths that are the most useful to our objective. Besides eliminating the replaceable probing path, we also try to avoid selecting the probing path without any contribution to achieving identifiability.

6. Conclusions and Future Work

The 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. The basic idea is to select probing paths that are the most useful for covering the target links and achieving identifiability. Furthermore, 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 eight ISP topologies demonstrate that our approach can achieve identifiability with a minimal set of probing paths. Compared with prior works, 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 the calculated path sets can produce good enough results. As future work, we will investigate possible solutions that can effectively pick the probing paths most useful in reducing the overall probing cost. Also, we will study other heuristics for path selection, and evaluate our approaches using different kind of probing cost.

References

[1] R. R. Kompellay, J. Yatesz, A. Greenbergz, and A. C. Snoeren, "IP fault localization via risk modeling," in *USENIX NSDI*, 2005, pp. 57–70.

[2] F. Li and M. Thottan, "End-to-end service quality measurement using source-routed probes," in *IEEE Infocom*, 2006, pp. 1–12.

[3] Y. Zhao, Y. Chen, and D. Bindel, "Towards unbiased end-to-end network diagnosis," in *ACM SIGCOMM*, 2006, pp. 219–230.

[4] 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.

[5] R. R. Kompella, J. Yates, and A. G. A. C. Snoeren, "Detection and localization of network black holes," in *IEEE Infocom*, 2007, pp. 2180–2188.

[6] P. P. C. Lee, V. Misra, and D. Rubenstein, "Toward optimal network fault correction via end-to-end inference," in *IEEE Infocom*, 2007, pp. 1343–1351.

[7] 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.

[8] M. H. Gunes and K. Sarac, "Analyzing router responsiveness to active measurement probes," in *Passive and Active Measurement Conference*, 2009, pp. 23–32.

[9] R. Sherwood and N. Spring, "Touring the Internet in a TCP sidecar," in *ACM SIGCOMM Conference on Internet Measurement*, 2006, pp. 339–344.

[10] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2 (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[11] Y. Bejerano and R. Rastogi, "Robust monitoring of link delays and faults in IP networks," in *IEEE Infocom*, 2003, pp. 134–144.

[12] H. X. Nguyen and P. Thiran, "Active measurement for multiple link failures diagnosis in IP networks," in *Passive and Active Measurement Conference*, 2004, pp. 185–194.

[13] Y. Chen, D. Bindel, H. Song, and R. H. Katz, "An algebraic approach to practical and scalable overlay network monitoring," in *ACM SIGCOMM*, 2004, pp. 55–66.

[14] P. Barford, N. Duffield, A. Ron, and J. Sommers, "Network performance anomaly detection and localization," in *IEEE Infocom*, 2009.

[15] T. Bu, N. Duffield, F. L. Presti, and D. Towsley, "Network tomography on general topologies," in *ACM SIGMETRICS*, 2002, pp. 21–30.

[16] J. Wu, Y. Zhang, Z. M. Mao, and K. G. Shin, "Internet routing resilience to failures analysis and implications," in *ACM CoNEXT*, 2007, pp. 1–12.

[17] Y. Breitbart, C.-Y. Chan, M. Garofalakis, R. Rastogi, and A. Silberschatz, "Efficiently monitoring bandwidth and latency in IP networks," in *IEEE Infocom*, 2001, pp. 933–942.

[18] L. Li, M. Thottan, B. Yao, and S. Paul, "Distributed network monitoring with bounded link utilization in IP networks," in *IEEE Infocom*, 2003, pp. 1189–1198.

[19] M. Thottan, L. E. Li, B. Yao, V. S. Mirrokni, and S. Paul, "Distributed network monitoring for evolving IP networks," in *IEEE ICDCS*, 2004, pp. 712–719.

[20] K. Suh, Y. Guo, J. Kurose, and D. Towsley, "Locating network monitors: complexity, heuristics, and coverage," in *IEEE Infocom*, 2005, pp. 351–361.

[21] G. H. Golub and C. F. V. Loan, *Matrix Computations (3rd Edition)*. Johns Hopkins University Press, 1996.

[22] R. Sherwood, A. Bender, and N. Spring, "Measuring ISP topologies with Rocketfuel," in *ACM SIGCOMM*, 2002, pp. 303–314.

[23] 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.

[24] 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, pp. 227–240.