Bandwidth Tomography with Probabilistic Routing

by

Xiaowei Zhang
M.Sc., University of Victoria, Canada, 2024

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Bandwidth Tomography with Probabilistic Routing

by

Xiaowei Zhang
M.Sc., University of Victoria, Canada, 2024

Supervisory Committee

Dr. Kui Wu, Supervisor
(Department of Computer Science)

Dr. Jianping Pan, Member
(Department of Computer Science)

ABSTRACT

The demands for maintaining optimal Quality of Service (QoS) have become higher with the rise of new Internet applications. Such demands include the need to monitor internet performance metrics such as bandwidth. Nevertheless, directly measuring the bandwidth of every individual link is often impractical due to the significant overhead involved in the measurement. Instead, it is essential to estimate the bandwidth of individual links using the measurements from only a few network nodes, a process known as bandwidth tomography. A recent work by Feng et al. introduced an algorithm to address this challenge, which determines the error bounds for link bandwidth estimates. This work, however, assumes deterministic routing. In real-world scenarios, routing may be probabilistic rather than deterministic. In this thesis, we extend the applicability of this existing work from the deterministic routing scenario to the probabilistic routing scenario by developing an algorithm that maps end-to-end available bandwidth values to their corresponding paths with high probability. Such an extension pushes the state-of-the-art bandwidth tomography toward a more complex scenario.

Contents

Supervisory Committee **ii**

Abstract **iii**

Contents **iv**

List of Tables **vi**

List of Figures **vii**

Acknowledgements **viii**

Dedication **ix**

1 Introduction and Background **1**

2 Related Work **7**

3 Mathematical Model and Problem Formulation **10**

3.1 Overview 10

3.2 Assumptions and Notations 11

3.3 An Algorithm to Solve the min-system 14

4 Algorithm Design **19**

4.1 Introduction 19

4.2 Overview 19

4.3 An Algorithm for Bound Inference 20

5 Performance Evaluation, Analysis and Discussion **25**

5.1 Introduction 25

5.2 Overview 25

5.3 The Ability of Inferring Correct Order of Probability Values 25
5.4 How Many Runs Are Enough? 28
5.5 Basic Demonstration 32
5.6 More Diverse Graphs 34

6 Conclusion and Future work **41**
6.1 Conclusion 41
6.2 Future Work 42

Bibliography **43**

List of Tables

Table 3.1 Three disjoint sets in the example 16

List of Figures

Figure 3.1 An example network 12
Figure 3.2 An example network for showing how CTB works. 14

Figure 4.1 Another example network. 21

Figure 5.1 Percentage of successes, $n = 4, p_1 = 0.1, p_2 = 0.2, p_3 = 0.3, p_4 = 0.4$. 26
Figure 5.2 Percentage of successes, $n = 7, p_1 = 0.02, p_2 = 0.05, p_3 = 0.1, p_4 = 0.15, p_5 = 0.18, p_6 = 0.2, p_7 = 0.3$. 27
Figure 5.3 Quantile-quantile plot of runs. 28
Figure 5.4 The function returned result, $n = 4$, $probArray = [0.1, 0.2, 0.3, 0.4]$ 31
Figure 5.5 The function returned result, $n = 7$, $probArray = [0.02, 0.05, 0.1, 0.15, 0.18, 0.2, 0.3]$ 32
Figure 5.6 A 6-vertex, 10-edge network. 33
Figure 5.7 A 7-node, 13-edge graph. 35
Figure 5.8 An 8-node, 17-edge graph. 36
Figure 5.9 A 9-node, 25-edge graph. 37
Figure 5.10A 10-node, 29-edge graph. 38
Figure 5.11An 11-node, 40-edge graph. 39
Figure 5.12A 12-node, 50-edge graph. 40

ACKNOWLEDGEMENTS

I would like to thank:

Supervisor, Dr. Kui Wu, for giving me a tremendous amount of support in all aspects of my life in the past two years. I wouldn't be where I am today without all the guidance and help from him. Words are not enough for me to say how grateful I am to have him as my supervisor. From the bottom of my heart, I would like to express my sincerest gratitude for him.

My family for always having my back. They shaped who I am today. Being by myself in another country for the past two years, I never felt alone because of them. I know I could always seek some peace of mind from them. My heart is always with them.

Everyone who helped me along the journey for being there for me when I needed it. What you have done means a lot to me.

Thank you.
Xiaowei Zhang

DEDICATION

To my family

Chapter 1

Introduction and Background

The advent of the Internet forwarded human civilization into a new era of connectivity. We have been enjoying the benefits the Internet brought us. Yet, just like any other tool, it needs to be properly maintained to keep being functional. At the early stage, it's relatively straightforward to oversee and maintain the Internet's operations due to its modest size and volume. One noticeable trait is that the number of users that needed to be kept track of was limited back then. As the Internet evolved, this is not the situation anymore. More and more users joined, and the exponential growth of the Internet in scale and volume presented new challenges. Today, the Internet has become a community of billions of active users, and there are still newcomers joining in every second. Under this circumstance, the maintenance of the Internet becomes more challenging than ever. New requirements are demanded, and ensuring the quality of service (QoS) and seamless network operations has become an urgent matter that needs to be dealt with.

In the modern world, the Internet is provided by different Internet service providers (ISPs). In order to uphold optimal QoS, it is necessary that every ISP needs to maintain close oversight of network performance parameters. This includes but is not limited to, the continuous monitoring of variables such as latency, available bandwidth, network congestion, and link losses. The careful observation and analysis of these factors are essential in order to uphold and guarantee optimal QoS, thereby maintaining the uninterrupted functionality of networks.

Yet, with a physical Internet of a gigantic scale, directly measuring individual link performances becomes impractical, primarily because of the substantial measurement overhead involved. Directly measuring the performance is not only economically inefficient but also simply infeasible because some ISPs may hide the details of their

internal infrastructure for security reasons. This calls for alternative approaches for evaluating link performances.

The advent of network virtualization, notably presented through concepts like overlay and the creation of network slices, has provided more flexibility in network resource management. Without heavily relying on the actual physical Internet, it allows network operators to dynamically form virtual networks to handle different applications. For instance, ISPs can take advantage of network slices to dynamically form different virtual networks for dedicated network applications. Unfortunately, network virtualization may make the network performance monitoring every harder. Due to the dynamic changes in network configuration, the importance of assessing the performance of virtual links or virtual services becomes even more complex, yet crucial.

In response to this challenge, a widely recognized strategy involves the indirect measurement of the performance or status of both physical and virtual links through end-to-end measurements. In essence, this approach tries to take advantage of the existing measuring and also exploit a priori knowledge, i.e. the physical structure of the Internet. Coined as "network tomography" in 1996 [29], this methodology has caught substantial attention across diverse research domains since its inception [9,12]. This thesis focuses on a specific problem within the realm of network tomography, in particular, the tomography approach to link bandwidth estimation in a dynamic routing environment.

As mentioned above, there has been a considerable amount of work done in the expansive field of network tomography since its birth. Noteworthy contributions include "Efficient identification of additive link metrics via network tomography" by Ma et al. [20] and "On the optimal monitor placement for inferring additive metrics of interested paths" by Yang et al. [30], among others. While all these diverse explorations have enriched the landscape, a significant amount of the work only concentrates on the additive metrics. As an example to explain the meaning of additive metrics, delay as a metric is additive since the delay of an end-to-end Internet path can be calculated by simply adding up the individual delay of every link on that path. The aforementioned two pieces of research are no exception, and both focus on additive link metrics.

Unlike delay, another primary performance metric of network, bandwidth, has seldom been investigated thoroughly in the network tomography literature, or at least, some aspects of it are missing from existing work. There have been a fair

amount of research papers that have studied methodologies for estimating end-to-end bandwidth, such as [10], in which the end-to-end bandwidth estimation techniques for both wired and wireless networks are thoroughly inspected. During this process, some dedicated measurement tools were also developed. For instance, Pathchar, clink, pchar, and bfind, etc, leverage ICMP "TTL exceeded" messages for bandwidth estimation. However, the primary focus of these tools is the estimation of end-to-end bandwidth. This means they focus more on the performance metrics of a path as a whole, without concerning the bandwidth of individual links. Although some tools, such as pathneck [15], can estimate link bandwidth along the measurement path by utilizing probe packets, these tools still have their limitations. Some require intermediate routers to support ICMP [15], while some exhibit inaccuracies on paths longer than a few hops [18]. What is strikingly absent is a network tomography approach specifically tailored for estimating the bandwidth of individual links throughout the entire network via a limited number of monitoring nodes — a concept denoted as bandwidth tomography in [13].

The other primary branch in the discipline of network tomography is Boolean-based network tomography, which people might think could be used to solve the bandwidth tomography problem. Aimed at identifying link failure or congestion, boolean-based network tomography shares loose relevance with but is fundamentally distinct from bandwidth tomography. The objective of Boolean-based network tomography, as outlined in [2], is to infer the congestion or failure status of a link or node based on the corresponding congestion/failure status of end-to-end paths. In essence, this means that the input and output of Boolean-based network tomography are binary, focusing on the binary states of congestion or failure, and it doesn't provide much value to the evaluation of network performance. This may not be enough to help ISP operators handle complex real-world problems. In contrast, bandwidth tomography pursues a different goal, which is to infer the specific bandwidth values of individual links by examining the end-to-end path bandwidth. This task is markedly different from the binary nature of Boolean-based network tomography and is of more value when it comes to the evaluation of the network performance.

As mentioned above already, most of the work on bandwidth focuses on end-to-end bandwidth estimation. In essence, this also delivers the information that it is more application-oriented and suitable for applications which only care about end-to-end performance instead of individual link performance. However, for the purpose of research and practically even for long-time development on infrastructure to be

made, the dig has to be deeper and individual link performance must be studied. Unfortunately, there hasn't been much work dedicated to this.

A primary reason that results in this absence is the minimum operation involved. With the standard linear equations, the approach is straightforward, as it usually comes down to finding the inverse of matrices or some numerical approach. This is not the case for *min*-equations. There's no direct approach to do a similar operation as in the inverse of matrices. Information loss is the other aspect of the inconvenience the minimum operation brings. In essence, we can formulate a problem at hand into a problem that takes the form of *min*-equations. However, the *min*-equations don't carry all the information of the original problem, because we ever only know from the bandwidth value of an end-to-end path that any constituent link's bandwidth value is not lower than that number.

In 2022, FENG et al. [13] proposed an algorithm dedicated to solving the bandwidth tomography problem with minimum operations. In this pioneering work, they took the groundbreaking step of formally formulating and systematically studying the core problem in bandwidth tomography, as specified as follows:

> *Given a network configuration, a designated set of end-to-end measurement paths, and the corresponding measured end-to-end bandwidth values, determine whether or not a link in the network is identifiable. Note that a link is identifiable if its bandwidth value can be uniquely determined. In instances where the bandwidth value cannot be uniquely determined, i.e., the link is not identifiable, can we estimate the upper and lower bound of its bandwidth value?*

In addressing this complex problem, a polynomial-time algorithm called CTB was presented in the paper [13]. This algorithm not only provides the approach to get the exact bandwidth value for all links that are identifiable within the network but can also obtain the smallest possible error bounds for those links that are not identifiable. Furthermore, they also presented a set of necessary and sufficient conditions for a link to be identifiable. Their solution to this problem lays a good foundation for future research in this crucial area of bandwidth tomography, opening avenues for further exploration and advancement.

It is worth noting that their algorithm is tailored for the deterministic scenario, i.e., under the assumption that the selection of end-to-end measurement paths is predefined and given, and the corresponding end-to-end bandwidth values are known. In this deterministic setting, the *min*-system would be uniquely determined before

applying the algorithm and then sent into the algorithm as input. However, this is not always the case in a more realistic situation. More specifically, in order to be a closer approximation of the real-world examples, the measurement paths may be probabilistic, where each potential path carries a distinct probability of being chosen. In this case, the *min*-system cannot be uniquely determined beforehand and used as the input for the algorithm. A new approach must be derived to handle this situation. In this thesis, we target this more complex scenario, where the paths' selection is not predetermined; instead, each path is associated with a probability of being chosen, adding an additional layer of complexity to the problem at hand.

Building upon the foundation laid by the CTB algorithm, we extend the algorithm so that it can handle this extra complexity of probability, which returns the precise bandwidth value for all identifiable links and the smallest error bounds for unidentifiable links. This solution provides a tailored approach to address the inherent complexities of probability in the context of the CTB algorithm, advancing the solution to the general network tomography in a more practical direction.

This extended algorithm holds significant practical implications. Despite not being seen as a standalone technology on the Internet, network tomography's integration with existing tools can extend their functionality. For instance, when coupled with popular network measurement tools like pathchar, clink, and pchar, this algorithm could return the network-wide, link-level bandwidth results. In the landscape of network monitoring techniques, it is worth mentioning that In-Band Network Telemetry (INT) [26] stands out as a highly advanced approach. It enables individual devices to directly report statistics in the data plane to monitoring applications in the centralized SDN control plane. While the deployment of large-scale INT could potentially compromise the practical need for network tomography, challenges facing large-scale INT, such as orchestration, data aggregation, security, and high monitoring overhead [26] persist. At least in the foreseeable future, INT is likely to have to work with legacy devices and cannot be deployed at large scale, making the incorporation of network tomography a valuable tool in overcoming these challenges. Intriguingly, the concept of bandwidth tomography has recently found applications in diverse domains, including balance inference in payment-channel networks [22], showcasing the versatility and relevance of this innovative approach. The result presented in this thesis, without doubt, expanded the possibility of deploying bandwidth tomography in real-life scenarios.

The rest of this thesis is organized as follows. In Chapter 2, we introduce related

work. In Chapter 3, we introduce the mathematical model and formally formulate the problem, bringing in all the essential concepts to facilitate the discussion and understanding of the rest of the thesis. In Chapter 4, we present our algorithm, along with some examples. In Chapter 5, we evaluate the algorithm and share some insights on its performance. In Chapter 6, we conclude the thesis and discuss future work in this domain.

Chapter 2

Related Work

Large-scale network tomography essentially means to estimate network performance parameters based on measurements at a limited amount of nodes. This sort of problem was first dealt with by Vardi, among others. In 1996, to estimate the traffic intensity between all source-destination (directed) pairs (SD pairs heretofore) of nodes of a network from repeated measurements of traffic flow along the directed links of the network in [29], Vardi coined the term network tomography due to the similarity between network inference and medical tomography. Since then, network tomography has been intensively studied. Two types of network tomography have been addressed in some literature not long after that. One type is link-level parameter estimation based on end-to-end path-level traffic measurements. The other type is sender-receiver path-level traffic intensity estimation based on link-level traffic measurements.

The first type of network tomography research includes work such as "Multicast-based inference of network-internal loss characteristics" by Cáceres et al. [7], "Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements" by Ratnasamy and McCanne [23], "Multicast-based inference of network-internal delay distributions" by Presti et al. [21]. The goal for this kind of network tomography is to estimate each link's queuing delay or loss rate. Towards this goal, the traffic measurements involved usually include time delays between packet transmissions and receptions or counts of packets transmitted and/or received between source and destination nodes. Nevertheless, most of the existing works in this category, if not all, fall into the realm of additive metrics. When it comes to bandwidth tomography that involves minimum operation, very few works can be found in the literature due to difficulties, including those exemplified in the previous chapter.

The second type of network tomography research includes work such as "An EM

approach to OD matrix estimation" by Vanderbei and Iannone [28], "Bayesian inference on network traffic using link count data" by Tebaldi and West [27], "A scalable method for estimating network traffic matrices from link counts" by Cao at al. [8]. In this type of network tomography, the goal is to estimate the amount of traffic originated from a specified node and destined for a specified receiver. Towards this end, the measurements for this kind of network tomography naturally constitute counts of packets that flow through nodes in the network. In this category of network tomography, all existing research assumes additive metrics. When it comes to bandwidth tomography, it is impossible to estimate end-to-end bandwidth based on the bandwidth measurement from partial internal links. This observation is based on the fact that any unmeasured internal link could be a bottleneck link. Hence, the only way to correctly infer the end-to-end bandwidth based on internal link measurement is to measure all internal links.

Another sub-area of network tomography that has drawn lots of research attention is boolean-based network tomography. There has been some work done on this sub-branch, such as "Range tomography: combining the practicality of boolean tomography with the resolution of analog tomography" by Zarifzadeh at al. [31], "Fundamental limits of failure identifiability by Boolean network tomography" by Bartolini et al. [4], "On fundamental bounds on failure identifiability by boolean network tomography" by Bartolini at al. [3]. Boolean-based network tomography identifies link failure or link congestion. Hence, it is quite different from bandwidth tomography, as the primary goal of bandwidth tomography is to infer the bandwidth value rather than the binary status of individual links.

What is also relevant to bandwidth tomography and worth pointing out is extensive research on measuring end-to-end bandwidth [10]. Some examples of advancement in this area include tools such as *clink*, *pathchar* and *bfind*. Nevertheless, it is worth noting that these tools do not care about the evaluation of individual links' bandwidth. In this sense, these tools serve as the foundation and the basic components (w.r.t. end-to-end path bandwidth) for bandwidth tomography. They, however, cannot directly solve the problem of bandwidth tomography that infers the bandwidth value of each individual link.

In the deterministic network calculus domain, *service curve decomposition* is a topic that actually comes close to the problem in bandwidth tomography. Some of the representative work includes "Network tomography using min-plus system theory" by Rizk [24], "On the identifiability of link service curves from end-host measurements

by Rizk and Fidler" [25], "Service curves in Network Calculus: dos and don'ts" by Bouillard at al. [6]. Service curve describes various network elements, including routers, links, etc. It utilises functions of time which specify the service that is offered by the element during a defined time interval. It's also under the so-called min-plus algebra. Given its cumulative nature, time plays a critical role, distinguishing the problem of service curve decomposition from the bandwidth tomography problem explored in this study. To date, service curve decomposition has primarily focused on line and tree topologies, with time serving as a defining factor in the analysis. This significant difference comes from the unique methodology in network calculus, where traffic arrivals and services are modelled with the accumulative amount over time.

In terms of measurement paths between monitors, existing research can be categorized into two distinct categories: uncontrollable measurement and controllable measurement. In the first category, probing packets travel, adhering to the rules defined in the routing tables within the network. This type of work includes "Robust monitoring of link delays and faults in IP networks" by Bejerano and Rastogi [5], "Practical beacon placement for link monitoring using network tomography" by Kumar and Kaur [17]. In the category of controllable measurement, directing probing packets along specified paths using *source routing*. Within this category, we assume that all internal routers support source routing. Most work assumed that the measurement paths are given and studied the strategy of probing the measurement paths. Notably, the strategy of constructing these measurement paths has often been overlooked in existing literature.

In terms of network models, both directed and undirected graphs have been used. In this thesis, we assume the network is undirected, the same as in most previous works done.

Chapter 3

Mathematical Model and Problem Formulation

In this chapter, we introduce some concepts used as the basis of this thesis, as well as the mathematical model of the problem being studied.

3.1 Overview

In this thesis, a network is represented by an edge-labelled graph $\mathcal{G} = (V, L)$, where V denotes the set of nodes and L denotes the set of links. Every link is associated with a value, which represents the bandwidth value of that link. In such a network, we assume that monitors can be deployed at certain nodes, and therefore, the bandwidth of a measurement path can be obtained with some existing methods, such as *pathload* [16]. Note that the term "measurement path" will be formally defined in the section below. In a real-world setting, there's an upper-bound bandwidth for every link due to physical limitations. To simplify the analysis, we assume a maximum bandwidth b_{max} for all the links in the network, which can be set based on the physical specification of the network. Note that this maximum bandwidth must not exceed the physical capacity of the link. Also note that the analytical results of this research can be easily extended to the situation where different links have different maximum bandwidth values. The only change to be made in that case would be instead of assigning every link a "global" b_{max}, every link would have its own individual b_{max}. The discussion of the upper bound of links is necessary to facilitate the research, but it is not our main focus. Compared to the upper bound, we actually

care more about the lower bound. This is mainly due to the following reasons. From a practical point of view, we need to ensure the bandwidth does not get too low to maintain the internet flow and the normal functionality of the network. Hence, the lower bound is the main focus of this research, and the algorithm to be introduced targets at the finding of the highest lower bound of a link.

3.2 Assumptions and Notations

To facilitate the discussion, some basic assumptions and notations are formally introduced as follows:

- $\mathcal{G}$: An edge-labelled, undirected graph. Every edge in the graph represents a link in the network. Edge-labelled means every link is assigned with a value which represents the bandwidth of the link. In addition, we assume that the network satisfies the following constraints:
  - No two links in $\mathcal{G}$ share the same pair of nodes.
  - No self-loop: Each link has distinct end nodes.

  This restriction is necessary for a graph to represent a practical network.

- *Measurement Path* (MP): A non-loop path that contains and only contains two monitors at its end nodes. Through these monitors, the end-to-end bandwidth of an MP could be observed by probing packets. In this thesis, we do not assume source routing where we know exactly which path a probing packet travels. Instead, we assume an uncontrollable routing, where probing packets between one source-destination pair may take different paths.

- The network under consideration is assumed to be "static". That is, either the bandwidth represents statistical characteristics (e.g., mean) that stay constant over time, or it changes fairly slowly relative to the span of the measurement process. This is also a well-acknowledged assumption, adopted in existing network tomography work [19], [30], [14].

- *Probabilistic Routing*: Given two monitors, probabilistic routing means there are a few different actual possible MPs between these two monitors and each of them is associated with a probability, which indicates the possibility of a path

being actually taken. All probabilities for a pair of monitors should add up to 1.

It is worth highlighting again that most of the previous work in network tomography is from an additive metrics point of view, i.e., additive metrics such as delay are used to measure the performance of the network. This work is based on bandwidth tomography, which adopts $min-$operation, that is, the bandwidth of an MP is the minimum bandwidth of all links along this MP. This restricts us from using traditional linear algebra.

Below, we use an example to demonstrate the concept of probabilistic routing.

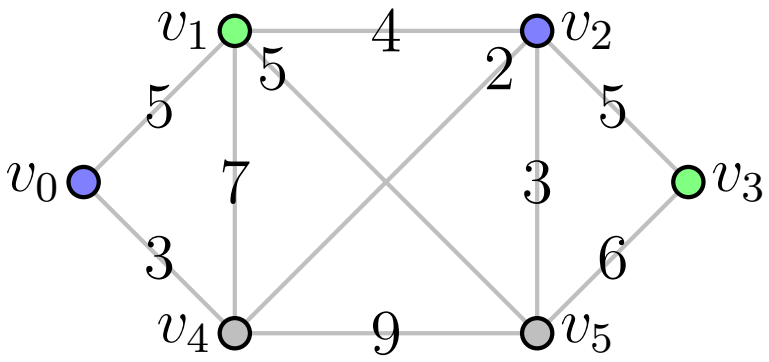

Figure 3.1: An example network

As shown in Figure 3.1, the network contains six nodes. Four nodes are equipped with monitors, marked in blue and green, where the same color denotes a pair of monitors for end-to-end path measurement. We denote $x_{p,q}$ the bandwidth of link $l_{p,q}, (p, q = 0, 1, 2, 3, 4, 5)$. If there's no link between node v_p and v_q, $x_{p,q} = 0$. As mentioned earlier, we assume an undirected graph, which indicates that $x_{p,q} = x_{q,p}$. In this specific example, we denote two monitor pairs $M_i (i = 1, 2)$, $M_1 : v_0$ and v_2 and $M_2 : v_1$ and v_3. For monitor pair M_1, we assume there are two possible MPs $P_1 : v_0 \rightarrow v_1 \rightarrow v_2$ and $P_2 : v_0 \rightarrow v_4 \rightarrow v_2$, associated with probability of being actually taken $p_1 = 0.7$ and $p_2 = 0.3$, respectively. For monitor pair M_2, we assume there are also two possible MPs $P_3 : v_1 \rightarrow v_2 \rightarrow v_3$ and $P_4 : v_1 \rightarrow v_5 \rightarrow v_3$, associated with probability of being actually taken $p_3 = 0.6$ and $p_4 = 0.4$, respectively. For *one measurement*, the observed end-to-end bandwidth from each monitor pair M_i is $b_i (i = 1, 2)$, respectively. Based on these notations, we have in this example the following four possible systems of *min*-equations for one experiment, denoted as *min*-

systems in the rest of the paper:

$$\begin{cases} x_{0,1} \wedge x_{1,2} = b_1 \\ x_{1,2} \wedge x_{2,3} = b_2 \end{cases} \tag{3.1}$$

$$\begin{cases} x_{0,1} \wedge x_{1,2} = b_1 \\ x_{1,5} \wedge x_{5,3} = b_2 \end{cases} \tag{3.2}$$

$$\begin{cases} x_{0,4} \wedge x_{4,2} = b_1 \\ x_{1,2} \wedge x_{2,3} = b_2 \end{cases} \tag{3.3}$$

$$\begin{cases} x_{0,4} \wedge x_{4,2} = b_1 \\ x_{1,5} \wedge x_{5,3} = b_2 \end{cases} \tag{3.4}$$

where $\wedge$ means the *min* operation.

We take *min*-systems (3.1) as an example and denote it into its equivalent matrix form $\mathbf{R} \wedge \mathbf{x} = \mathbf{b}$, where

$$\mathbf{R} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} \tag{3.5}$$

$$\mathbf{x} = \begin{pmatrix} x_{0,1} & x_{1,2} & x_{2,3} \end{pmatrix}^{\mathbf{T}} \tag{3.6}$$

$$\mathbf{b} = \begin{pmatrix} b_1 & b_2 & b_3 \end{pmatrix}^{\mathbf{T}} \tag{3.7}$$

The problem then takes this form: **given R and b, infer the values of x**.

With all the possible MPs for monitor pairs, now the problem is how we could estimate each link bandwidth based on that information. One naïve thought is to first conduct a series of experiments, gather as much path-selection information and build the final *min*-system. Once we have the *min*-system, the problem becomes the problem of solving the *min*-system, which has been addressed already. In [13], an algorithm is proposed to solve *min*-systems, i.e., given a *min*-system, estimate the bandwidth of links in the network. We briefly introduce this algorithm in the next section as background.

3.3 An Algorithm to Solve the min-system

As mentioned earlier in the thesis, linear algebra could not be used to solve the *min*-systems. The Calculate-Tightest-Bounds (CTB) algorithm by Feng et al. [13] addresses this problem. In this section, we briefly introduce this algorithm.

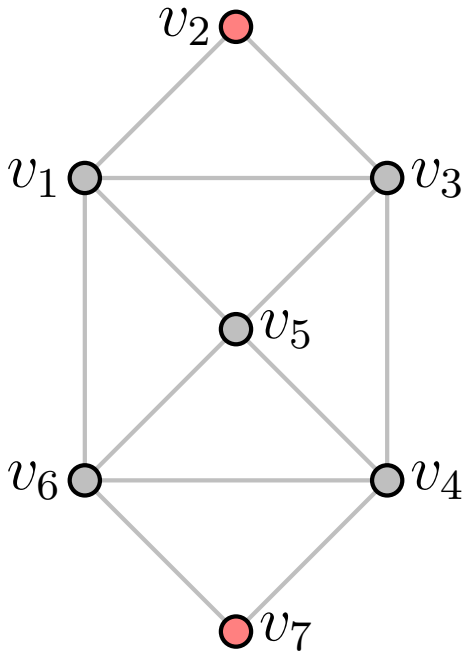

Figure 3.2: An example network for showing how CTB works.

We first give an example to demonstrate how the algorithm works. An example network is shown in Figure 3.2, where two monitors are marked in red. Suppose we have built the *min*-system based on five MPs as follows:

$$\begin{cases} P_1 : x_{2,3} \wedge x_{3,4} \wedge x_{4,7} = 3 \\ P_2 : x_{2,3} \wedge x_{3,5} \wedge x_{5,4} \wedge x_{4,7} = 5 \\ P_3 : x_{2,1} \wedge x_{1,6} \wedge x_{6,7} = 5 \\ P_4 : x_{2,1} \wedge x_{1,5} \wedge x_{5,4} \wedge x_{4,7} = 4 \\ P_5 : x_{2,1} \wedge x_{1,5} \wedge x_{5,6} \wedge x_{6,7} = 3 \end{cases} \tag{3.8}$$

This *min*-system contains three distinct end-to-end bandwidth values (b value), and the maximum is 5. Therefore, the greatest lower bound for any link covered is at most 5. To simplify the later analysis, we sort the *min*-equations in the non-increasing

order of the b value and have the equivalent matrix form parameters as follows:

$$\mathbf{R} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{3.9}$$

$$\mathbf{x} = \begin{pmatrix} x_{2,1} & x_{2,3} & x_{1,5} & x_{1,6} & x_{3,4} & x_{3,5} & x_{4,7} & x_{5,4} & x_{5,6} & x_{6,7} \end{pmatrix}^{\mathbf{T}} \tag{3.10}$$

$$\mathbf{b} = \begin{pmatrix} 5 & 5 & 4 & 3 & 3 \end{pmatrix}^{\mathbf{T}} \tag{3.11}$$

Note that the links that are not covered by any MP are not included in the $\mathbf{x}$ vector, as there's no information at all for us to give any deduction on the bandwidth value of those links. We consider the paths with end-to-end bandwidth value 5 (i.e., P_2 and P_3). We can conclude that any link covered by these two paths must have a bandwidth no smaller than 5. We put all the links covered by these two paths into a set S_5 and put it aside for now. The subscript 5 denotes the bandwidth value 5. Then we turn to consider the rest links. The new *min*-system we have now is:

$$\mathbf{R_{new}} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{3.12}$$

$$\mathbf{x_{new}} = \begin{pmatrix} x_{1,5} & x_{3,4} & x_{5,6} \end{pmatrix}^{\mathbf{T}} \tag{3.13}$$

$$\mathbf{b_{new}} = \begin{pmatrix} 4 & 3 & 3 \end{pmatrix}^{\mathbf{T}} \tag{3.14}$$

This process is the most crucial part of the CTB algorithm, which is to eliminate some variables and simplify the question for further analysis. In this example, after we have this new *min*-system, we can derive the bandwidth value of the rest three links easily. It is clear that $l_{1,5}$ is identifiable and its value $x_{1,5}$ is 4. The same then goes for $l_{3,4}$ and $l_{5,6}$, with their values $x_{3,4}$ and $x_{5,6}$ being 3 and 3, respectively. This part of the process is reflected in the rest part of the CTB algorithm, which is to find all the identifiable links in the *min*-system, and put all the links into disjoint sets based on their maximum lower bounds, like what we did with the set S_5. In this

Table 3.1: Three disjoint sets in the example

	Covered links
S_5	$l_{2,1}, l_{2,3}, l_{3,5}, l_{5,4}, l_{4,7}, l_{1,6}, l_{6,7}$
S_4	$l^*_{1,5}$
S_3	$l^*_{3,4}, l^*_{5,6}$

Note: * means identifiable.

example, we have three of these disjoint sets S_3, S_4 and S_5, as shown in Table 3.1.

Here, we outline two key properties of the CTB algorithm. It can determine whether a link's bandwidth value can be uniquely derived. If not, it returns the link's tightest error bound, where the error bound is defined as the interval between the upper and lower bounds of the link's bandwidth.

For the thesis to be self-contained, we present the CTB algorithm as shown in Algorithm 1 [13]. Lines 1-8 are the variable elimination part of the algorithm. Lines 9-20 are the part that assigns intervals and determines the bounds, as mentioned earlier in the example.

Algorithm 1 The CTB Algorithm [13]

input : a *min*-system, $\mathbf{R} \wedge \mathbf{x} = \mathbf{b}$, where $\mathbf{R} = [r_{ij}]_{m \times n}$ is a Boolean matrix, $\mathbf{x} = \begin{pmatrix} x_1 & ... & x_n \end{pmatrix}^{\mathbf{T}}$ and $\mathbf{b} = \begin{pmatrix} b_1 & ... b_m \end{pmatrix}^{\mathbf{T}}$, m *min*-equations ordered in the $non-increasing$ order of their values

output: the tightest error bound for every link

```
for i ← 1 to m − 1 do
    for j ← i + 1 to m do
        if b_i == b_j then
            continue;
        end
        r_jk = max r_jk - r_ik, 0, k = 1,...,n;
    end
end
for i ← 1 to m do
    for j ← 1 to n do
        if r_ij == 1 then
            Assign x_j into interval [b_i,b_max], i.e., x_j's error bound is b_max − b_i;
        end
    end
end
for i ← 1 to m do
    if there is only one r_ij == 1(j = 1, ..., n) then
        x_j = b_i, i.e., x_j's error bound reduces to 0;
    end
end
```

In this chapter, we have defined concepts and notations that will be used throughout the thesis. We also briefly reviewed the CTB algorithm, upon which our research on probabilistic routing is built, to lay the foundation for further presentation of the research.

In the next chapter, we will thoroughly introduce our algorithm design and continue using the notations and concepts from this chapter.

Chapter 4

Algorithm Design

4.1 Introduction

In this chapter, we introduce our solution to the probabilistic routing problem outlined in the previous chapter. We first overview the problem's essential components, laying the groundwork for the subsequent development of our mathematical model. We then present our algorithm.

4.2 Overview

In the previous chapter, we explained the probabilistic routing problem. Based on that, we now extract the essential elements of the problem and formulate the mathematical model. Some examples will be used to illustrate the idea better.

Let's imagine an experiment of throwing balls over a wall and explain how this imaginary experiment maps to our problem at hand. The experiment is as follows. Assume that there are n boxes behind a wall. Each box is associated with a value, and each box has a speaker that speaks out its associated value when a ball falls into the box. We assume that different boxes have different values. On the other side of the wall, a person who cannot see the boxes throws balls over the wall, and we assume that each ball will land on one of the boxes. The person throws $m > n$ balls and thus can record m results (some results may have the same value). Based on the m results and a prior probability that a ball lands on each box, the person wants to estimate the value of each box.

The above process offers a simulated probabilistic routing scenario. We can treat

the possible MPs associated with an end-to-end bandwidth value for a monitor pair as different boxes. One measurement thus corresponds to throwing one ball over the wall. We only know the value of the measurement but do not know exactly which MP this measurement takes. Our task is to estimate each MP's end-to-end bandwidth based on repeated measurements. To simplify the analysis, we assume that:

1. Between a pair of monitors, there are n possible MPs. For one end-to-end measurement, the probe packet may use MP $i(i = 1, 2, \ldots, n)$ with probability p_i where $\sum_{i=1}^{n} p_i = 1$. We assume that these MPs and the values of p_i are known. This assumption, while seemingly strong, is reasonable because we can use tools, such as *traceroute*, to probe the reachability between the two monitor nodes. After enough *traceroute* tests, we can know the constitution of a path and the associated probability that a probe uses this path. Without the loss of generality (WLOG), we assume that p_i's are arranged in ascending order.

2 We assume that even if we do not know the n values associated with the n MPs, respectively, these values are different. With this assumption, we should continue our measurement $m > n$ times until we record n different values[1] This assumption is to facilitate our solution. Note that we have not developed an effective solution when MPs have the same end-to-end bandwidth. In the extreme case, all MPs have the same end-to-end bandwidth. In this case, our proposed idea will not work because repeated measurements do not grant us any new information to infer which path a probe packet passes through.

4.3 An Algorithm for Bound Inference

Our idea is as follows. After $m(> n)$ measurements, we group the m results into n groups, each group representing a different value. Assume that each group includes m_i results, i.e., the empirical frequency is $\frac{m_i}{m}$. Order the groups in the ascending order of the frequency. Then, we assume that this frequency order mirrors the prior probability distribution of $p_i(i = 1, 2 \ldots, n)$. In essence, the group with the fewest hits is associated with the smallest probability, the group with the second-fewest hits is associated with the second-smallest probability, and so on. This algorithm assumes

[1]In the real world, it is nearly impossible that two measurements, even if using the same MP, obtain the exact same measurement results. People normally relax the accuracy, e.g., rounding up to tens, so that close values are treated as equal.

that the values of the ordered groups are one-to-one mapping to the values of the n boxes.

In our probabilistic routing context, this idea could be interpreted as follows: Assume that we have n paths for a pair of monitors. The precise bandwidth values for each path are unknown. However, the probabilities of being taken associated with each path during routing are known. We run the experiment $m(m > n)$ times and gain m observed measurement values for the monitor pair. Then we group the m results into n groups. The group with the least counting signifies the smallest probability, that value of this group is therefore assigned to the MP the smallest probability. The group with the second least counting signifies the second smallest probability, the value of that group is therefore assigned to the MP of the second-smallest probability, and so forth.

The above procedure is applied to other monitor pairs. After we conduct the experiments for a given number of times[2], we get the information required to build a *min*-system to which we could then apply the CTB algorithm and get the tightest bound for every link, or the bandwidth value of the link if the link is identifiable.

Again, we use a specific example to illustrate the algorithm idea concretely. We consider a network as shown in Figure 4.1 below:

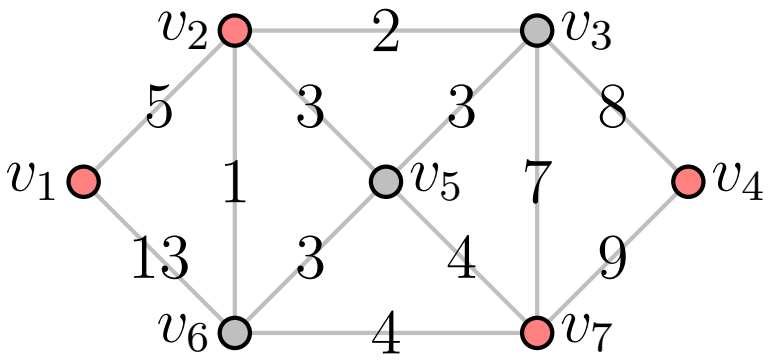

Figure 4.1: Another example network.

In this scenario, we have two monitor pairs: v_1 and v_4, v_2 and v_7. To facilitate a clearer discussion, we use M_1 to denote the monitor pair v_1 and v_4, and M_2 to the monitor pair v_2 and v_7. For M_1, we assume that there are four possible paths P_1, P_2, P_3 and P_4: $P_1 : v_1 \to v_2 \to v_3 \to v_4$, $P_2 : v_1 \to v_6 \to v_7 \to v_4$, $P_3 : v_1 \to v_6 \to$

[2]The number of experiments is determined by the users based on their measurement budget. Note that users also determine whether or not to measure all monitor pairs.

$v_5 \to v_7 \to v_4$, $P_4 : v_1 \to v_2 \to v_6 \to v_7 \to v_4$, whose equation forms are shown below. Each path is associated with probabilities of being taken, say, 0.2, 0.4, 0.3, and 0.1, respectively. We'll use these denotations throughout our discussion. The '?' in the equations means that we are simulating the real-world scenario and don't know the exact bandwidth value for each path beforehand.

$$\begin{cases} P_1 : x_{1,2} \wedge x_{2,3} \wedge x_{3,4} =? \\ P_2 : x_{1,6} \wedge x_{6,7} \wedge x_{7,4} =? \\ P_3 : x_{1,6} \wedge x_{6,5} \wedge x_{5,7} \wedge x_{7,4} =? \\ P_4 : x_{1,2} \wedge x_{2,6} \wedge x_{6,7} \wedge x_{7,4} =? \end{cases} \tag{4.1}$$

For M_2, we assume three possible MPs Q_1, Q_2 and Q_3: $Q_1 : v_2 \to v_5 \to v_7$, $Q_2 : v_2 \to v_3 \to v_7$, $Q_3 : v_2 \to v_6 \to v_7$, whose equation forms are shown below. Each path is associated with probabilities of being taken, say, 0.5, 0.3 and 0.2, respectively.

$$\begin{cases} Q_1 : x_{2,5} \wedge x_{5,7} =? \\ Q_2 : x_{2,3} \wedge x_{3,7} =? \\ Q_3 : x_{2,6} \wedge x_{6,7} =? \end{cases} \tag{4.2}$$

Assume that we conduct the experiment a set number of times, say 1000, and record the observed end-to-end bandwidth values from monitor pairs every time. Based on the recorded results, we need to infer the bound inference. For M_1, assume the number of appearances of the observed end-to-end bandwidth for each value group is: 2 (220 times), 4 (396 times), 3 (307 times), 1 (77 times). We arrange these groups in ascending order of the number of appearances of the value associated as 1 (77 times), 2 (220 times), 3 (307 times), and 4 (396 times).

Since we know the probability for each path ordered in ascending order: P_4: 0.1, P_1: 0.2 P_3: 0.3 and P_2: 0.4. Based on the above information, we assume that P_1 is associated with the end-to-end bandwidth value 2, P_2 is associated with the end-to-end bandwidth value 4, P_3 is associated with the end-to-end bandwidth value 3, and P_4 is associated with the end-to-end bandwidth value 1.

For M_2, assume the number of appearances for each observed value group is 3 (485 times), 2 (283 times), 1 (232 times). Similarly, we then arrange these groups in ascending order of the number of appearances of the value associated as 1 (232 times), 2 (283 times), 3 (485 times).

Since also have the probability for each path ordered in the ascending order: Q_3:

0.2, Q_2: 0.3 and Q_1: 0.5. Accordingly, we assume that Q_1 is associated with the end-to-end bandwidth value 3, Q_2 is associated with the end-to-end bandwidth value 2, and Q_3 is associated with the end-to-end bandwidth value 1.

Combining all the above information together, we construct the *min*-system as follows:

$$\begin{cases} P_1 : x_{1,2} \wedge x_{2,3} \wedge x_{3,4} = 2 \\ P_2 : x_{1,6} \wedge x_{6,7} \wedge x_{7,4} = 4 \\ P_3 : x_{1,6} \wedge x_{6,5} \wedge x_{5,7} \wedge x_{7,4} = 3 \\ P_4 : x_{1,2} \wedge x_{2,6} \wedge x_{6,7} \wedge x_{7,4} = 1 \\ Q_1 : x_{2,5} \wedge x_{5,7} = 3 \\ Q_2 : x_{2,3} \wedge x_{3,7} = 2 \\ Q_3 : x_{2,6} \wedge x_{6,7} = 1 \end{cases} \tag{4.3}$$

Now that we have the *min*-system, we can then apply the CTB algorithm to get the error-bound inference for every link.

We formally present the above idea in Algorithm 2. Lines 1 - 6 of the algorithm build the *min* system based on the known network structure and measured end-to-end bandwidth values from the experiments. Lines 7 - 27 apply the CTB algorithm to the acquired *min* system and get the tightest error bound for every link or the bandwidth value of the link if it is identifiable.

In the next chapter, we will focus on the numerical evaluation of our algorithm and bring in some insightful discussion.

Algorithm 2 The Probabilistic Routing Algorithm

Input : the network, the monitor pairs $M_i(i = 1, 2, ..., m)$, all the possible MPs $P_{ij}(j = 1, 2, ..., m_i)$ for each monitor pair M_i, with its corresponding equation form $E_{ij}(j = 1, 2, ..., m_i)$(the left part of the *min*-equation) and probability $p_{ij}(j = 1, 2, ..., m_i)$ of being taken. The number of each measured end-to-end bandwidth value $b'_{ij}s(j = 1, 2, ..., m_i)$ appearance $C_{ij}(j = 1, 2, ..., m_i)$ for each $M_i(i = 1, 2, ..., m)$ in a given number of experiments

Output: the tightest error bound for every link

Order all the MPs $P_{ij}(j = 1, 2, ..., m_i)$ for each $M_i(i = 1, 2, ..., m)$ in the ascending order of their corresponding probability $p_{ij}(j = 1, 2, ..., m_i)$. Denote these ordered MPs $\mathbf{P}_{ij}(j = 1, 2, ..., m_i)$ and the corresponding equation form $\mathbf{E}_{ij}(j = 1, 2, ..., m_i)$. Order the number of appearance $C_{ij}(j = 1, 2, ..., m_i)$ in ascending order. Denote the ordered number of appearance $\mathbf{C}_{ij}(i = 1, 2, ..., m, j = 1, 2, ..., m_i)$ and its corresponding measured end-to-end bandwidth value $\mathbf{b}_{ij}(i = 1, 2, ..., m, j = 1, 2, ..., m_i)$.

for $i \leftarrow 1$ **to** m **do**
  for $j \leftarrow 1$ **to** m_i **do**
    $\mathbf{E}_{ij} = \mathbf{b}_{ij}$, add this equation to the *min*-system
  end
end

Apply the CTB algorithm to the *min*-system, $\mathbf{R} \wedge \mathbf{x} = \mathbf{b}$, where $\mathbf{R} = [r_{ij}]_{m \times n}$ is a Boolean matrix, $\mathbf{x} = \begin{pmatrix} x_1 & ... & x_n \end{pmatrix}^{\mathbf{T}}$ and $\mathbf{b} = \begin{pmatrix} b_1 & ... b_m \end{pmatrix}^{\mathbf{T}}$, m *min*-equations ordered in the $non - increasing$ order of their values.

for $i \leftarrow 1$ **to** $m - 1$ **do**
  for $j \leftarrow i + 1$ **to** m **do**
    if $b_i == b_j$ **then**
      continue;
    end
    r_{jk} = maxr_{jk} - r_{ik}, 0, k = 1,...,n;
  end
end
for $i \leftarrow 1$ **to** m **do**
  for $j \leftarrow 1$ **to** n **do**
    if $r_{ij} == 1$ **then**
      Assign x_j into interval [b_i,b_{max}], i.e., x_j's error bound is $b_{max} - b_i$;
    end
  end
end
for $i \leftarrow 1$ **to** m **do**
  if *there is only one* $r_{ij} == 1(j = 1, ..., n)$ **then**
    $x_j = b_i$, i.e., x_j's error bound reduces to 0;
  end
end

Chapter 5

Performance Evaluation, Analysis and Discussion

5.1 Introduction

In the preceding chapter, we presented the problem definition and designed an algorithm to solve the problem. In this chapter, we conduct a series of simulation studies to evaluate the performance of our algorithm.

5.2 Overview

In the first part of the evaluation, we test whether our simple frequency-based statistics can be used to estimate the correct order of probability values (Sections 5.3 and 5.4). After that, we apply our algorithm to a simple network to demonstrate its effectiveness (Section 5.5). In the last part of the evaluation, we test the reliability of our algorithm with more diverse graphs (Section 5.6).

5.3 The Ability of Inferring Correct Order of Probability Values

The core idea of the algorithm is that the frequency is a good approximation of the probability distribution if the number of measurements is big enough. To validate, we run experiments with the following setting: Let n denote the number of the possible

paths for a monitor pair. Denote the bandwidth value associated with each path as $v_i(i = 1, 2, \ldots, n)$, the corresponding probability of each path being chosen as $p_i(i = 1, 2, \ldots, n)$. We then run simulations in groups, with each group consisting of multiple runs. A group is said to be a success if the order of paths according to the times of being chosen is the same as the order of probability values associated with the paths. For instance, assume that $n = 4$ and $p_1 = 0.1, p_2 = 0.2, p_3 = 0.3, p_4 = 0.4$. Assume that the number of runs in a group is 100. If we find that v_1 12 times, v_2 23 times, v_3 31 times and v_4 34 times, then we say this group of simulation is successful because the order $34 > 31 > 23 > 12$ exactly matches the order $p_4 > p_3 > p_2 > p_1$.

Analysis 1: Set $n = 4, p_1 = 0.1, p_2 = 0.2, p_3 = 0.3, p_4 = 0.4$, we run experiments in 1000 groups, with each group consisting of the same number of runs. We change the number of runs in each group from 100 to 1000 and conduct a series of experiments. We then calculate the success rate of groups. Figure 5.1 shows how the percentage of success groups changes as the number of runs in each group grows.

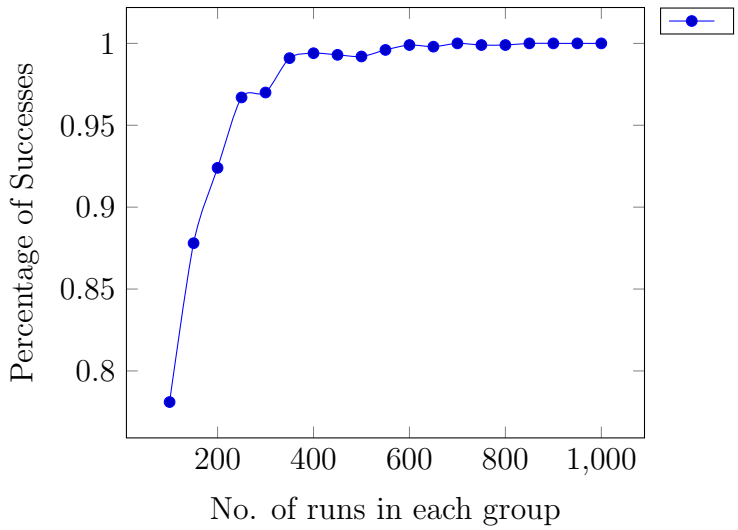

Figure 5.1: Percentage of successes, $n = 4, p_1 = 0.1, p_2 = 0.2, p_3 = 0.3, p_4 = 0.4$.

As we can observe from the figure, the percentage of success groups becomes higher as the number of runs in each group gets bigger and converges to 1. This is easy to understand because as the volume of the sample data grows, the frequency values should become closer and closer to the true probabilities. While we do not have a theoretical analysis on the relationship between the percentage of success groups and the number of runs, the above result suggests that there seems to be a threshold (e.g., 400 runs) above which the percentage of success groups becomes stably high.

Analysis 2: Set $n = 7, p_1 = 0.02, p_2 = 0.05, p_3 = 0.1, p_4 = 0.15, p_5 = 0.18, p_6 = 0.2, p_7 = 0.3$, we change the number of runs in each group ranges from 100 to 15000 and run experiments again. Same as before, we draw the figure to show how the percentage of success groups changes as the number of runs in each group grows, as shown in Figure 5.2.

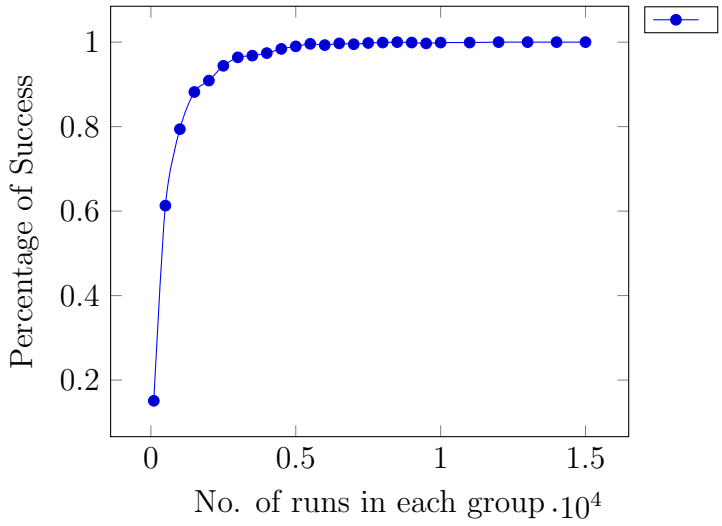

Figure 5.2: Percentage of successes,
$n = 7, p_1 = 0.02, p_2 = 0.05, p_3 = 0.1, p_4 = 0.15, p_5 = 0.18, p_6 = 0.2, p_7 = 0.3$.

In this example, the probabilities are much closer to each other. As a result, we observe that a larger number of runs is required to approximate the right order of the probability values closely.

Analysis 3: Inspired by the idea of the quantile-quantile plot [1], we draw a quantile-quantile plot in Figure 5.3, using the same setting in Analysis 1, i.e., $n = 4, p_1 = 0.1, p_2 = 0.2, p_3 = 0.3, p_4 = 0.4$. We have the number of runs increase each step by 50 from 100 all the way to 1000. In the quantile-quantile plot, the x-axis represents the number of different values that appeared in different numbers of runs. The y-axis represents the number calculated by the ground-truth probability. For instance, if we set the number of runs to be 100, the theoretical numbers should be $10, 20, 30$ and 40, respectively. These theoretical numbers are labelled on the y-axis. In the simulation, if v_1 appears 12 times, v_2 appears 19 times, v_3 appears 31 times, v_4 appears 38 times, these numbers are recorded on the x-axis. As the number of runs increases, we should spot a trend of the line closer to the straight line of slope 1. The figure supports this conclusion.

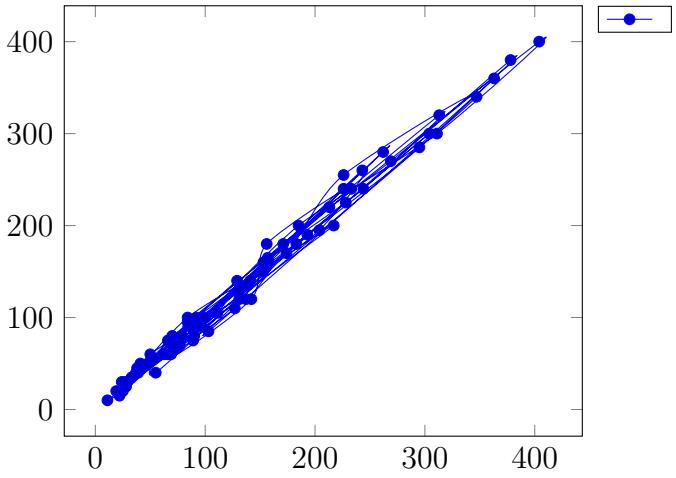

Figure 5.3: Quantile-quantile plot of runs.

5.4 How Many Runs Are Enough?

As the data above shows, our algorithm works well when the number of runs is large enough. Naturally, a critical question is how we know we have reached a sufficient number of runs. In other words, we should find the relationship between the number of runs and the probability of assigning measurement results to their corresponding MPs correctly.

We first pose the above question mathematically to illustrate the difficulties in theoretical analysis. Due to the difficulties, we try to approach this question quantitatively rather than analytically. The mathematical idea was already formulated in the previous chapter, but we will reiterate it below for a cohesive discussion.

Assume that we have n boxes behind a wall and each box is associated with a value $v_i (i = 1, \ldots, n)$ unknown to the observer who is on the other side of the wall. W.L.O.G., assume that v_i's have different values. The observer throws a ball over the wall, and the ball lands on one of the n boxes with probability $p_i (i = 1, \ldots, n)$, where $\sum_{i=1}^{n} p_i = 1$. Assume that the observer knows these p_i values. Assume that p_i's have different values and are ordered in the ascending order of their values. When the ball lands on box $i (i = 1, \ldots, n)$, the box, equipped with a sensor and a speaker, tells the value which can be heard by the observer. The observer, however, does not know which box the ball has landed on. The observer throws $m (> n)$ balls and records the corresponding m results. Assume that the m results cover all the n values. The observer's goal is to infer the value of each box.

The observer puts the same results into one group and then obtains n groups.

Assume that each group includes m_i results. Order the groups in the ascending order of m_i. The observer concludes that the values of the ordered groups are one-to-one mapping to the (ordered) values of the n boxes.

- **Q**: What is the probability that the observer finds out all the values of the boxes correctly? This is the equivalent question of the probability of our algorithm assigning the measurement results to their corresponding MPs between one monitor pair correctly.

A numerical solution to Q: Since it is difficult to obtain a closed-form theoretical result, we present an algorithm to recursively calculate the probability, as shown in Algorithm 3. The input of the algorithm includes n, m, and $probArray$, where n is the total number of boxes, m is the total number of balls (hits), and $probArray$ is an array recording the probabilities associated with each box in ascending order.

We briefly go through the idea of this recursive calculation. Essentially, we need to calculate the probability that the first box gets hit the least times, the second box the second least times, the third box the third least times, and so on. The probability of this event could be calculated recursively in the following way. We first consider all the possible numbers of hits for the first box, i.e., from 1 to the total number of hits. For each of these different numbers, the recursive function includes a chunk (marked in black in Line 9 of Algorithm 3) that multiplies three factors together: the probability of hits corresponding to that number, the corresponding binomial coefficient, and the corresponding probability of the possible number of hits for the rest of boxes, which is also calculated using this function recursively. Then, the summation of these chunks returns the final probability value.

In the recursive function, the base case is reached when the recursion comes to the last box because once we have assigned that box with a number of hits, the number of hits for all the boxes will be set. The recursive function includes three parameters, $currentBox, remainingHits$, and $currentBoxLBounds$.

The parameter $currentBox$ indicates which box the recursive calls have come to. For instance, $currentBox = 2$ means we have sorted out all the possible numbers of hits for the first box and are now trying to assign the rest boxes with all possible numbers of hits, starting from the second box. $remainingHits$ is the number of total hits left to assign with for the boxes from what $currentBox$ indicates up to the last box.

The parameter $remainingHits$ indicates the number of remaining balls. For example, if $remainingHits = 100$ and we have the first box assigned with 1 hit, then the rest of the boxes from the second box to the last one can only have 99 balls (i.e., hits). This parameter should be 99 at the next call. If we have the first box assigned with 2 hits, then the rest of the boxes from the second box to the last one can only have 98 balls, and this parameter should be 98 at the next call, and so on. The realization of this process also involves the parameter $currentBoxLBounds$, which indicates the lower bound for the current $currentBox$th box based on the instantiation of the boxes before. For the same $currentBox$, we can have different $currentBoxLBounds$ values. Following the example shown before, if the first box gets 1 hit, then the second box can only have values equal to or bigger than 2, in which case, $currentBoxLBounds$ is 2. If the first box gets 2 hits, then the second box can only have values equal to or bigger than 3, in which case, $currentBoxLBounds$ is 3, and so on.

Algorithm 3 probCalculation

input : The total number of boxes: n.
Total number of ball hits: m.
The array recording the ground truth p_i values: $probArray$

output: The probability that the observer finds out all the values of the boxes correctly.

```
1  currentBox = 1
   currentBoxLBounds = 1
   remainingHits = m
   return recursion(currentBox, currentBoxLBounds, remainingHits)
2  Function recursion(currentBox, currentBoxLBounds, remainingHits):
3      Result=0
4      if currentBox == n then
5          return [ProbArray[currentBox − 1]]^remainingHits
6      else
7          N = floorFunction((remainingHits − ((n − currentBox + 1) ∗ (n −
             currentBox)/2))/(n − currentBox + 1))
8          for i ← currentBoxLBounds to N do
9              Result ← add binomial(remainingHits, i) ∗ [ProbArray[currentBox −
                 1]]^i ∗ recursion(currentBox + 1, remainingHits − i, i + 1)
10         end
11         return Result
12     end
```

To show the correctness of Algorithm 3, we use it in some basic examples that could also be easily calculated by other means. As a first example, we set the pa-

rameters to $n = 2, m = 100, probArray = [0.45, 0.55]$. These parameters mean 2 boxes, throwing the ball 100 times and the probability of hitting the 1st box and 2nd box being 0.45 and 0.55, respectively. Algorithm 3 returns 0.8172718153138593. Since this is essentially a binomial distribution, we can simply calculate this probability by using "binom" module in "scipy". That gives us 0.8172718153138547, which is nearly identical to the result derived from our algorithm. The negligible difference between the two results is due to the accuracy limit in Python numerical packages. We then raise the times of throwing the ball up to 1000, i.e., $m = 1000$. Our algorithm returns 0.9991534507838472, whereas the latter "binom" approach gives us 0.9991534507833807. We also test on the situation where we have $n = 3, m = 10, probArray = [0.1, 0.3, 0.6]$. Our algorithm returns 0.32878483199999997. In this specific setting, we have 3 boxes and throw the ball 10 times. We can list out all the possible combinations, including 4 combinations as follows, shown as tuples in the order of the 1st box, the 2nd box, then the 3rd box: (1, 4, 5), (1, 3, 6), (2, 3, 5), (1, 2, 7). We can directly calculate this probability with "multinomial" module in "scipy", which gives us 0.3287848320000001. All the above examples verify the correctness of Algorithm 3.

We apply Algorithm 3 in more complex examples. We first set the parameters to $n = 4$, $probArray = [0.1, 0.2, 0.3, 0.4]$. We have the value of m increase from 50 to 1000. We record the data and draw a figure showing the probability changes as the value of m changes, as shown in Figure 5.4.

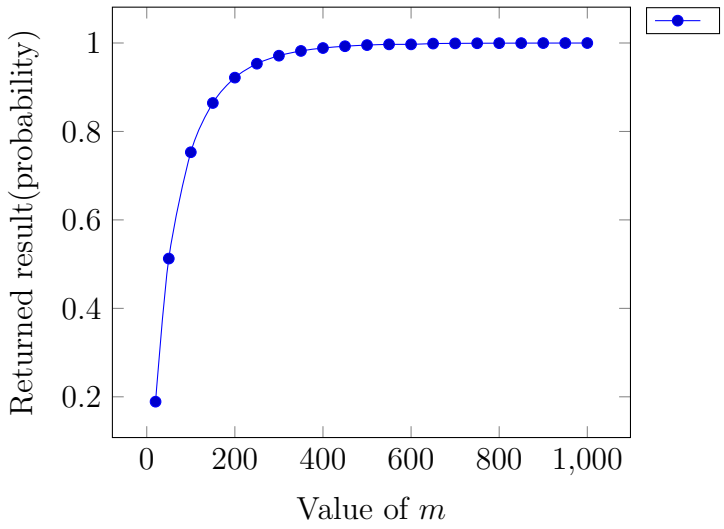

Figure 5.4: The function returned result, $n = 4$, $probArray = [0.1, 0.2, 0.3, 0.4]$

From Fig. 5.4, we can see that as the value of m increases, we get higher and higher probability of returning the values of all the boxes correctly. This probability has the trend of converging to 1, which makes perfect sense based on the law of large numbers. Also, based on the figure, we can see that setting $m = 300$ is good enough, i.e., the number of runs 300 is enough.

We then set the parameters to $n = 7$, $probArray = [0.02, 0.05, 0.1, 0.15, 0.18, 0.2, 0.3]$ and do a similar experiment. The results are as shown in Figure 5.5.

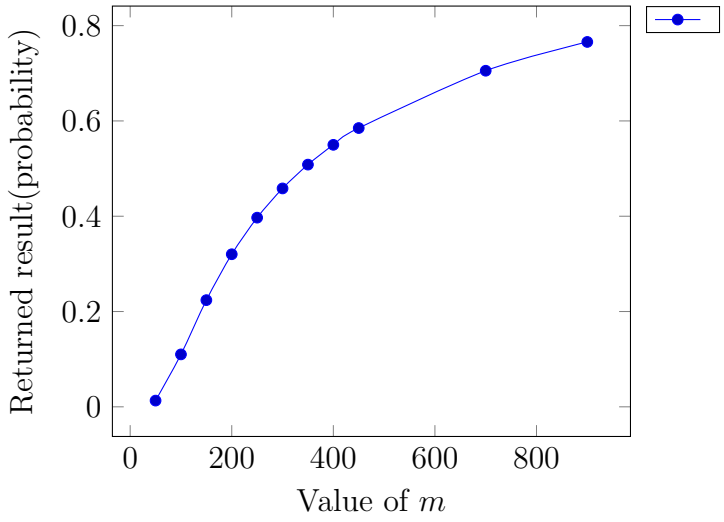

Figure 5.5: The function returned result, $n = 7$, $probArray = [0.02, 0.05, 0.1, 0.15, 0.18, 0.2, 0.3]$

As we increase the value of m, Algorithm 3 took 48 hours to return the result when m is set to 900 on a Macbook Air (8-core CPU, 16 GB memory, macOS). We can see that the returned probability increases as the number of balls increases. At $m = 900$, we reach a probability of 0.76577 of returning all the values of the boxes correctly. By having more runs, we can expect this probability to increase further and converge to 1, based on the law of large numbers.

5.5 Basic Demonstration

After we have shown that our simple frequency-based statistics can be used to estimate the correct order of probability values, we in this section test Algorithm 2 on a simple network shown in Figure 5.6.

This network has 6 vertices and 10 edges. Each edge is labelled with a number denoting the ground true bandwidth value of the edge. We have our experiment

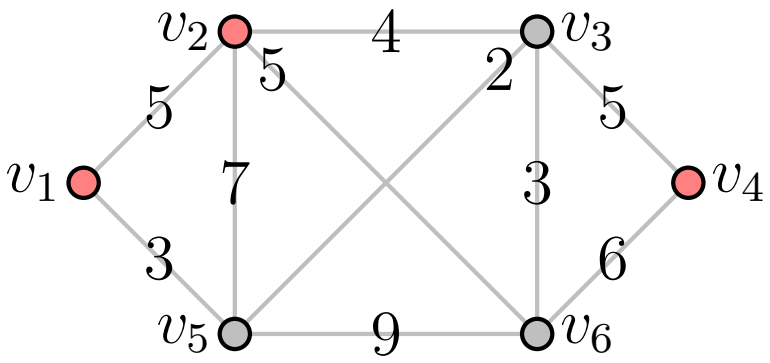

Figure 5.6: A 6-vertex, 10-edge network.

setting as follows: We consider 2 source-destination pairs, v_1-v_4 and v_2-v_4. For the v_1-v_4 pair, we consider two MPs, v_1-v_2-v_3-v_4 and v_1-v_5-v_6-v_4. The probability of taking each path is 0.8 and 0.2, respectively. For the v_2-v_4 pair, we consider three MPs, v_2-v_3-v_4, v_2-v_6-v_4 and v_2-v_5-v_6-v_4. The probability of taking each path is 0.5, 0.4 and 0.1, respectively. Then, we conduct a series of experiments where we apply Algorithm 2 to this graph and collect the results.

Here we introduce some key ideas in our evaluation process. First, we compare the bandwidth value of every link obtained with Algorithm 2 with the ground truth. In the case of the gained result being a single value (identifiable), we say that our algorithm fails if it gives a value that differs from the ground truth. In the case of the link being unidentifiable (the gained result being a range), we say the algorithm fails if that range doesn't include the ground truth. We call all the links corresponding to these failed results failing links, and call the rest non-failing links. We then can get the percentage of non-failing links. In addition, among all the non-failing links, we add their error bounds altogether. In this specific example, because for each of the two monitor pairs, the end-to-end bandwidth value for each MP is distinct, we could then predetermine the true *min*-system and input them into the CTB algorithm and get a result as well. This result is the best possible result we can derive, given the information, because we pretend that we know the paths used in the experiments. We then also add the non-failing and unidentifiable links' error bounds derived by this approach all together. We then can compute the ratio of "our algorithm non-failing links total error bounds"/"corresponding links best possible total error bounds". Then, we can evaluate the performance of Algorithm 2 from two aspects: the first aspect is the non-failing links percentage, and the second aspect is how close the result from our

approach among non-failing links is to the best possible result.

For each pair of monitors, we test with 110000 runs. This number is used to ensure that we have enough runs for each tested scenario. In fact, the probability that the observer finds out all the values of the MPs for the v_2-v_4 pair correctly, according to our Algorithm 3, is about 0.99 when the number of runs is set to 1000, so 110000 is more than enough to ensure the accuracy. With the aforementioned probability, the *min*-system we end up getting is exactly the true *min*-system, so the result we end up getting is exactly "the best possible total error bounds" result. In other words, 1) The non-failing links percentage is 100%. 2) The result from our algorithm among non-failing links is exactly the same as the best possible result.

5.6 More Diverse Graphs

In this section, we test Algorithm 2 on more diverse graphs. We randomly generate 6 graphs and source-destination pairs within these graphs. The main tool we use to generate graphs can be found at [11], a package written in Python to generate graphs. To use this package, we need to provide the number of nodes, the number of edges, and some other parameters. In our case, we start with the number of nodes from 7 up to 12. Note that given the number of nodes, the maximum number of edges is determined. For instance, with the number of nodes being 7, the maximum number of edges is 21. We also feed the $-w$ parameter of the Python package with int and assign each edge with an integer weight in the range $[0, 100]$. The weight on each link serves as the ground truth bandwidth for that link. Then, for each graph, we choose some monitor pairs randomly.

We illustrate the graph creation process with an example below. With the 7 nodes graph, we set 13 edges. The graph is shown in Figure 5.7. Then we randomly generate 6 numbers (repetition allowed) among all the numbers associated with nodes. Then, we form the first source-destination pair with the first 2 numbers, the second source-destination pair with the next 2 numbers, and the third source-destination pair with the rest 2 numbers. For the 7 nodes graph, we get source-destination pairs v_0-v_6, v_0-v_2 and v_1-v_4. We will use a similar approach to generate source-destination pairs for the upcoming graphs as well. For each upcoming pair of monitors, we still test with 110000 runs. Again, this number is used to ensure that we have enough runs for each tested scenario. In the upcoming examples, the most MPs one S-D pair has is seven. According to our Algorithm 3 and experiments in section 5.4, for instance,

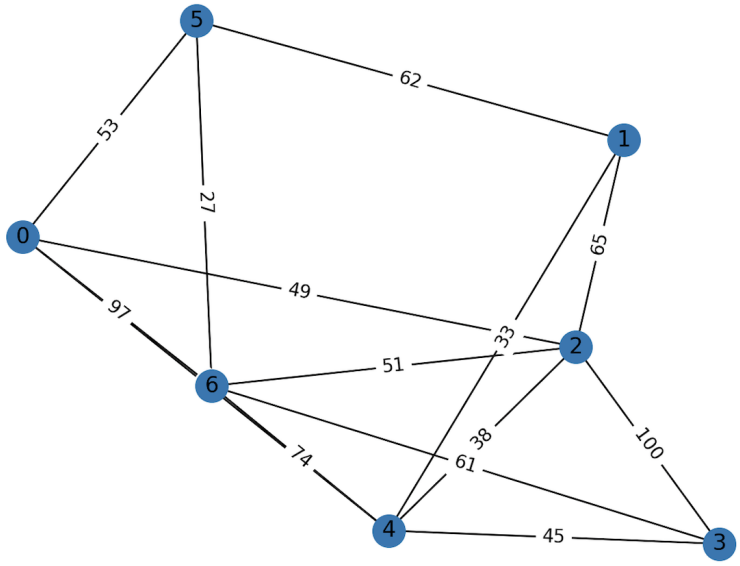

Figure 5.7: A 7-node, 13-edge graph.

when we have the *probArray* as $[0.02, 0.05, 0.1, 0.15, 0.18, 0.2, 0.3]$, corresponding to seven MPs, the probability of finding boxes correctly, is no less than 0.77 when the number of runs is set to 900, so 110000 is more than enough to ensure the accuracy. As for the MPs, for v_0-v_6, we consider five possible MPs: v_0-v_6, v_0-v_2-v_6, v_0-v_2-v_4-v_6, v_0-v_5-v_6 and v_0-v_2-v_3-v_6, with the probability of each path being taken being 0.4, 0.25, 0.11, 0.09 and 0.15, respectively. For v_0-v_2, we consider six possible MPs: v_0-v_2, v_0-v_6-v_2, v_0-v_6-v_4-v_2, v_0-v_5-v_6-v_2, v_0-v_5-v_1-v_2 and v_0-v_6-v_3-v_2, with the probability of each path being taken being 0.15, 0.16, 0.12, 0.1, 0.17 and 0.3, respectively. For v_1-v_4, we consider five possible MPs: v_1-v_4, v_1-v_2-v_4, v_1-v_2-v_3-v_4, v_1-v_2-v_6-v_4 and v_1-v_5-v_6-v_4, with the probability of each path being taken being 0.25, 0.2, 0.15, 0.3 and 0.1, respectively. We then apply Algorithm 2 on this graph.

As for the result, other than completely off on the estimation of 3 edges out of 13, our algorithm successfully (which means the true bandwidth value is within the estimated range) gives its estimate on the rest 10 edges. For these 10 edges, the best possible total error bound we could derive using the baseline algorithm is 1030, provided that we know which path it actually takes every time we get an observed value, whereas our approach gets 1586. The ratio of the result of our approach to that of the ideal approach is 1.54.

Next, we come to the setting of an 8 nodes, 17 edges graph, as shown in Figure 5.8.

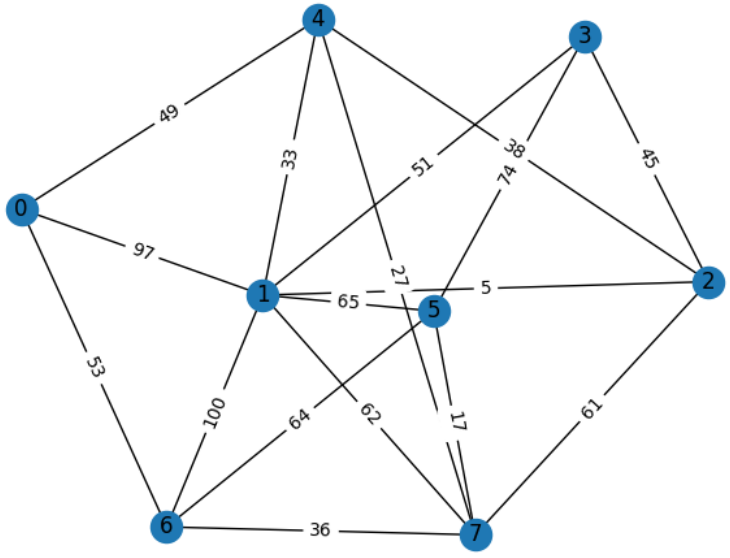

Figure 5.8: An 8-node, 17-edge graph.

This time, we have S-D pairs v_3-v_5, v_5-v_7 and v_2-v_4. As for the MPs, for v_3-v_5, we consider four possible MPs: v_3-v_5, v_3-v_1-v_5, v_3-v_2-v_1-v_5 and v_3-v_2-v_7-v_5, with the probability of each path being taken being 0.4, 0.3, 0.1 and 0.2, respectively. For v_5-v_7, we consider five possible MPs: v_5-v_7, v_5-v_1-v_7, v_5-v_6-v_7, v_5-v_3-v_2-v_7 and v_5-v_1-v_4-v_7, with the probability of each path being taken being 0.1, 0.3, 0.25, 0.23 and 0.12, respectively. For v_2-v_4, we consider two possible MPs: v_2-v_4 and v_2-v_1-v_4, with the probability of each path being taken being 0.6 and 0.4, respectively. Our algorithm is completely off on 5 edges out of 17. On the other 12 edges, the best possible total error bound is 1640 whereas our algorithm gives 2289. The ratio of the result of our approach to that of the ideal approach is 1.4.

As a hypothesis, adding more S-D pairs would allow our algorithm to return a better estimate of the bandwidth values of links. So, we added some more S-D pairs to validate this hypothesis. With the addition of more S-D pairs, the ideal algorithm gave a better estimate, from the total error bound being 1640 with 3 S-D pairs to 879 with 8 S-D pairs. Our algorithm, however, didn't gain a significant performance boost. Up until the addition of the fifth S-D pair, the total error bound derived from our algorithm remained the same as what it was before we added any more S-D pairs. This result suggests that adding more S-D pairs in the context of probabilistic routing does not necessarily improve the inference results.

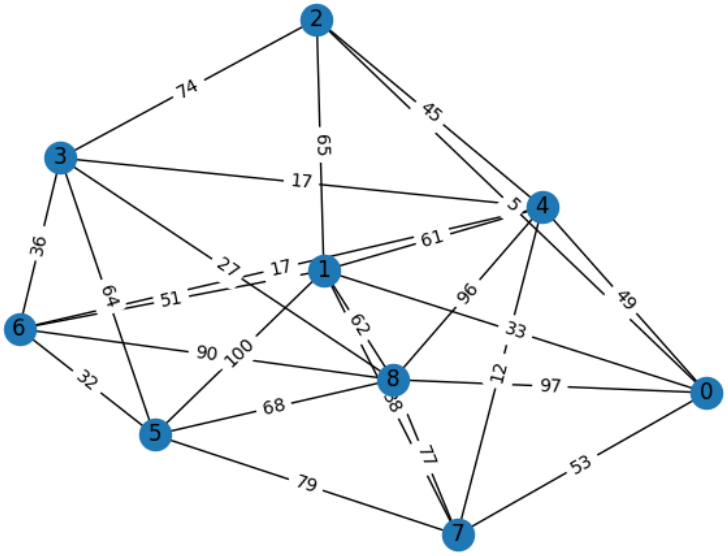

Figure 5.9: A 9-node, 25-edge graph.

We continue with the expansion of nodes and edges. This time, we come to a graph with 9 nodes and 25 edges, as shown in Figure 5.9. We have S-D pairs this time v_2-v_3, v_0-v_1 and v_6-v_7. As for the MPs, for v_2-v_3, we consider four possible MPs: v_2-v_3, v_2-v_1-v_5-v_3, v_2-v_1-v_6-v_3 and v_2-v_4-v_6-v_3, with the probability of each path being taken being 0.47, 0.3, 0.15 and 0.08, respectively. For v_0-v_1, we consider five possible MPs: v_0-v_1, v_0-v_8-v_1, v_0-v_4-v_1, v_0-v_7-v_1 and v_0-v_2-v_1, with the probability of each path being taken being 0.18, 0.35, 0.26, 0.16 and 0.05, respectively. For v_6-v_7, we consider four possible MPs: v_6-v_5-v_7, v_6-v_8-v_7, v_6-v_1-v_7 and v_6-v_4-v_7, with the probability of each path being taken being 0.2, 0.46, 0.24 and 0.1, respectively. This time, our algorithm performed pretty well. It failed at the estimation on only 1 edge. For the rest 24 edges, the total error bound given by the ideal algorithm is 3760, whereas our algorithm gives us 4456. The ratio of the result of our approach to that of the ideal approach is 1.19.

We next come to a graph with 10 nodes and 29 edges, as shown in Figure 5.10. We have S-D pairs v_7-v_6, v_0-v_2 and v_0-v_1. As for the MPs, for v_7-v_6, we consider two possible MPs: v_7-v_5-v_6 and v_7-v_1-v_6, with the probability of each path being taken being 0.65 and 0.35, respectively. For v_0-v_2, we consider four possible MPs: v_0-v_2, v_0-v_8-v_2, v_0-v_4-v_2 and v_0-v_9-v_2, with the probability of each path being taken being 0.1, 0.33, 0.37 and 0.2, respectively. For v_0-v_1, we consider five possible MPs: v_0-

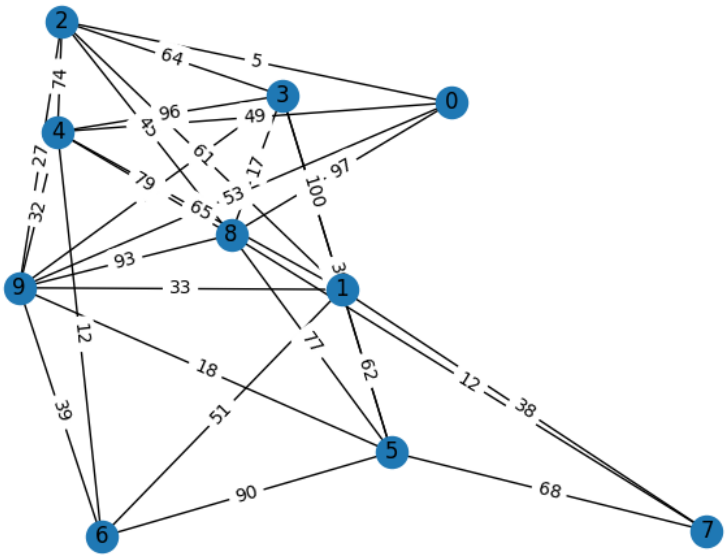

Figure 5.10: A 10-node, 29-edge graph.

v_2-v_3-v_1, v_0-v_8-v_9-v_1, v_0-v_2-v_4-v_1, v_0-v_9-v_5-v_1 and v_0-v_9-v_6-v_1, with the probability of each path being taken being 0.06, 0.3, 0.06, 0.18 and 0.4, respectively. Our algorithm again performs pretty well, failing at the estimation on only 1 edge. For the rest 28 edges, the ideal algorithm gives a total error bound of 5344, whereas our algorithm gives an estimation of 5990. The ratio of the result of our approach to that of the ideal approach is 1.12.

We then come to a graph with 11 nodes and 40 edges, as shown in Figure 5.11. We have S-D pairs v_5-v_2, v_5-v_0 and v_9-v_8. As for the MPs, for v_5-v_2, we consider five possible MPs: v_5-v_2, v_5-v_9-v_2, v_5-v_8-v_2, v_5-v_1-v_2 and v_5-v_3-v_2, with the probability of each path being taken being 0.3, 0.46, 0.07, 0.1 and 0.07, respectively. For v_5-v_0, we consider five possible MPs: v_5-v_1-v_0, v_5-v_8-v_0, v_5-v_9-v_0, v_5-v_3-v_0 and v_5-v_6-v_0, with the probability of each path being taken being 0.27, 0.05, 0.25, 0.15 and 0.28, respectively. For v_9-v_8, we consider seven possible MPs: v_9-v_8, v_9-v_5-v_8, v_9-v_1-v_8, v_9-v_2-v_8, v_9-v_4-v_8, v_9-v_3-v_8 and v_9-v_0-v_8, with the probability of each path being taken being 0.4, 0.04, 0.17, 0.12, 0.03, 0.06 and 0.18, respectively. This time, our algorithm fails at 5 edges. For the rest 35 edges, the ideal algorithm gives a total error bound of 6674, whereas our algorithm gives an estimation of 7286. The ratio of the result of our approach to that of the ideal approach is 1.09.

We then come to our final graph with 12 nodes and 50 edges, as shown in Fig-

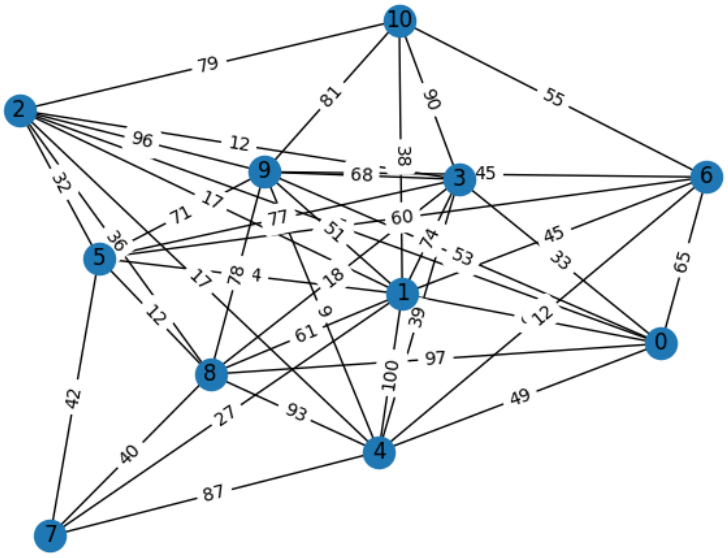

Figure 5.11: An 11-node, 40-edge graph.

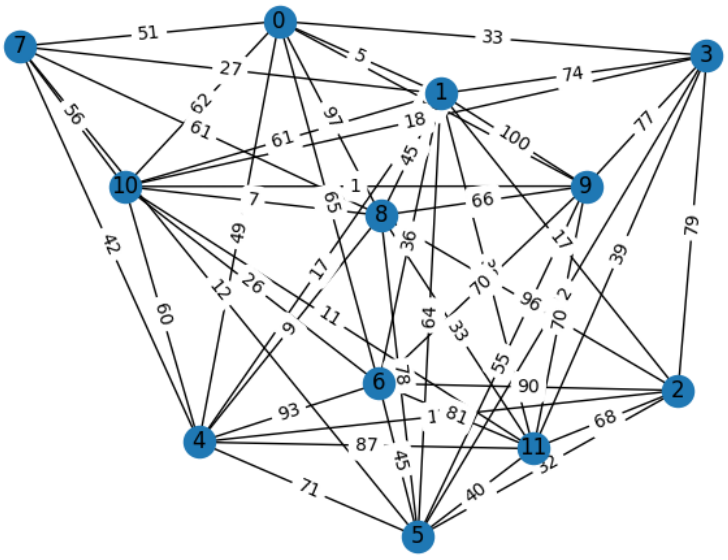

Figure 5.12: A 12-node, 50-edge graph.

ure 5.12. We have S-D pairs v_5-v_3, v_8-v_5 and v_6-v_2. As for the MPs, for v_5-v_3, we consider five possible MPs: v_5-v_3, v_5-$v_1$1-v_3, v_5-v_2-v_3, v_5-v_9-v_3 and v_5-v_1-v_3, with the probability of each path being taken being 0.08, 0.18, 0.14, 0.28 and 0.32, respectively. For v_8-v_5, we consider seven possible MPs: v_8-v_5, v_8-v_4-v_5, v_8-$v_1$1-v_5, v_8-v_2-v_5, v_8-v_1-v_5, v_8-v_7-v_5 and v_8-v_9-v_5, with the probability of each path being taken being 0.4, 0.027, 0.1, 0.096, 0.135, 0.036 and 0.206, respectively. For v_6-v_2, we consider five possible MPs: v_6-v_2, v_6-v_5-v_2, v_6-$v_1$1-$v_0$2, v_6-v_4-v_2 and v_6-v_1-v_2, with the probability of each path being taken being 0.5, 0.128, 0.254, 0.05 and 0.068, respectively. This time, our algorithm fails at 11 edges. For the rest 39 edges, the ideal algorithm gives a total error bound of 7966, whereas our algorithm gives an estimation of 8308. The ratio of the result of the naive approach to that of the ideal approach is 1.04.

In summary, extensive tests with more diverse networks have shown that our algorithm's performance varies from network to network, with a ratio over the best possible results ranging between 1.04 and 1.55. In addition, there seems to be a trend that this ratio drops when the network becomes more complex (i.e., more nodes and more edges). It is worth noting that exceptions to the above conclusion can always be found, but the conclusion generally holds because the networks under our evaluation are all generated randomly.

Chapter 6

Conclusion and Future work

6.1 Conclusion

In this thesis, we investigated a bandwidth tomography problem under the context of probabilistic routing. The problem is based on the observations that (1) source routing is not universally supported by routers, and thus, we cannot fully control the path a probing packet uses, and (2) some intermediate routers may be equipped with load balancers and thus may deliver packets for the same destination over multiple paths. The bandwidth tomography problem has been studied in the deterministic setting [13] where the path taken by a probing packet from the source to the destination is known. An algorithm named CTB [13] was developed to estimate link error bounds with some given end-to-end bandwidth data and network topology information. Nevertheless, it is not straightforward whether or not the CTB algorithm can be applied to the probabilistic routing scenario. The major contribution of this thesis is expanding the CTB algorithm to solve the bandwidth tomography problem under probabilistic routing.

To be more specific, the original CTB algorithm only deals with the predetermined, complete *min*-system, whereas our extended algorithm first determines the *min*-system based on the end-to-end measurement data and probabilities that measurement paths would be taken by probing packets. After we assign the measurement data to their corresponding measurement paths, we then apply the CTB algorithm to find the error bound of each individual link. We perform extensive simulation studies to evaluate our solution. Since this is the first study of the bandwidth tomography problem under probabilistic routing, we can only compare our solution with an ideal

case that returns the best possible answers. The evaluation results disclose the effectiveness of our solution. Compared to the ideal case, our algorithm achieves a performance ratio ranging between 1.04 and 1.55. Note that since the performance is measured in terms of total error bound, a higher value means a worse performance.

6.2 Future Work

Due to the high complexity of probabilistic bandwidth tomography, we only came up with a numerical solution. Deriving any closed-form analytical result remains an open challenge. This will be our future work.

Another more challenging problem is to relax the assumption about the constant bandwidth value of a measurement path/link. For instance, each link's bandwidth may follow some specific distribution (e.g., $\sim N(\mu, \sigma)$) rather than a constant value. This new problem aligns with real-world scenarios where the links' bandwidth is highly dynamic. So, even during a (short) time horizon of measurements, we cannot assume the links' bandwidth remains unchanged. A solution to this problem will broaden the application of bandwidth tomography to more dynamic networks.

Bibliography

[1] Jerry Banks. *Discrete event system simulation*. Pearson Education India, 2005.

[2] Novella Bartolini, Ting He, Viviana Arrigoni, Annalisa Massini, Federico Trombetti, and Hana Khamfroush. On fundamental bounds on failure identifiability by boolean network tomography. *IEEE/ACM Transactions on Networking*, 28(2):588–601, 2020.

[3] Novella Bartolini, Ting He, Viviana Arrigoni, Annalisa Massini, Federico Trombetti, and Hana Khamfroush. On fundamental bounds on failure identifiability by boolean network tomography. *IEEE/ACM Transactions on Networking*, 28(2):588–601, 2020.

[4] Novella Bartolini, Ting He, and Hana Khamfroush. Fundamental limits of failure identifiability by boolean network tomography. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[5] Yigal Bejerano and Rajeev Rastogi. Robust monitoring of link delays and faults in ip networks. In *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428)*, volume 1, pages 134–144. IEEE, 2003.

[6] Anne Bouillard, Laurent Jouhet, and Eric Thierry. *Service curves in Network Calculus: dos and don'ts*. PhD thesis, INRIA, 2009.

[7] Ramón Cáceres, Nick G Duffield, Joseph Horowitz, and Donald F Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions on Information theory*, 45(7):2462–2480, 1999.

[8] Jin Cao, Scott Vander Wiel, Bin Yu, and Zhengyuan Zhu. A scalable method for estimating network traffic matrices from link counts. *Preprint. Available at http://stat-www. berkeley. edu/binyu/publications. html*, 2000.

[9] Rui Castro, Mark Coates, Gang Liang, Robert Nowak, and Bin Yu. Network Tomography: Recent Developments. *Statistical Science*, 19(3):499 – 517, 2004.

[10] Shilpa Shashikant Chaudhari and Rajashekhar C Biradar. Survey of bandwidth estimation techniques in communication networks. *wireless personal communications*, 83(2):1425–1476, 2015.

[11] deyuan. Github. `https://github.com/deyuan/random-graph-generator`, 2022. Accessed on November 8, 2023.

[12] Nick Duffield. Network tomography of binary network performance characteristics. *IEEE Transactions on Information Theory*, 52(12):5373–5388, 2006.

[13] Cuiying Feng, Jianwei An, Kui Wu, and Jianping Wang. Bound inference and reinforcement learning-based path construction in bandwidth tomography. *IEEE/ACM Transactions on Networking*, 30(2):501–514, 2021.

[14] Ting He, Liang Ma, Ananthram Swami, and Don Towsley. *Network tomography: identifiability, measurement design, and network state inference*. Cambridge University Press, 2021.

[15] Ningning Hu, Li Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang. Locating internet bottlenecks: Algorithms, measurements, and implications. *ACM SIGCOMM Computer Communication Review*, 34(4):41–54, 2004.

[16] Manish Jain and Constantinos Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput. *ACM SIGCOMM Computer Communication Review*, 32(4):295–308, 2002.

[17] Ritesh Kumar and Jasleen Kaur. Practical beacon placement for link monitoring using network tomography. *IEEE Journal on Selected Areas in Communications*, 24(12):2196–2209, 2006.

[18] Kevin Lai and Mary Baker. Measuring link bandwidths using a deterministic model of packet delay. In *Proceedings of the conference on applications, technologies, architectures, and protocols for computer communication*, pages 283–294, 2000.

[19] Liang Ma, Ting He, Kin K Leung, Ananthram Swami, and Don Towsley. Inferring link metrics from end-to-end path measurements: Identifiability and monitor placement. *IEEE/ACM transactions on networking*, 22(4):1351–1368, 2014.

[20] Liang Ma, Ting He, Kin K Leung, Don Towsley, and Ananthram Swami. Efficient identification of additive link metrics via network tomography. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, pages 581–590. IEEE, 2013.

[21] F Lo Presti, Nick G Duffield, Joseph Horowitz, and Don Towsley. Multicast-based inference of network-internal delay distributions. *IEEE/ACM Transactions On Networking*, 10(6):761–775, 2002.

[22] Yan Qiao, Kui Wu, and Majid Khabbazian. Non-intrusive and high-efficient balance tomography in the lightning network. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 832–843, 2021.

[23] Sylvia Ratnasamy and Steven McCanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, volume 1, pages 353–360. IEEE, 1999.

[24] A Rizk. *Network tomography using min-plus system theory.* PhD thesis, TU Darmstadt, 2008.

[25] Amr Rizk and Markus Fidler. On the identifiability of link service curves from end-host measurements. In *International Conference on Network Control and Optimization*, pages 53–61. Springer, 2008.

[26] Lizhuang Tan, Wei Su, Wei Zhang, Jianhui Lv, Zhenyi Zhang, Jingying Miao, Xiaoxi Liu, and Na Li. In-band network telemetry: A survey. *Computer Networks*, 186:107763, 2021.

[27] Claudia Tebaldi and Mike West. Bayesian inference on network traffic using link count data. *Journal of the American Statistical Association*, 93(442):557–573, 1998.

[28] Robert J Vanderbei and James Iannone. An em approach to od matrix estimation. Technical report, Technical Report SOR 94-04, Princeton University, 1994.

[29] Yehuda Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the American statistical association*, pages 365–377, 1996.

[30] Rongwei Yang, Cuiying Feng, Luning Wang, Weiwei Wu, Kui Wu, Jianping Wang, and Yinlong Xu. On the optimal monitor placement for inferring additive metrics of interested paths. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pages 2141–2149. IEEE, 2018.

[31] Sajjad Zarifzadeh, Madhwaraj Gowdagere, and Constantine Dovrolis. Range tomography: combining the practicality of boolean tomography with the resolution of analog tomography. In *Proceedings of the 2012 Internet Measurement Conference*, pages 385–398, 2012.