NETWORK TOMOGRAPHY

A Dissertation

by

MAHMOOD ETTEHAD

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2020

Major Subject: Mathematics

ABSTRACT

A communication network can be modeled as a directed connected graph with edge weights that characterize performance metrics such as loss and delay.Network tomography aims to infer these edge weights from their pathwise versions measured on a set of intersecting paths between a subset of boundary vertices, and even the underlying graph when this is not known. In particular, temporal correlations between path metrics have been used to infer composite weights on the subpath formed by the path intersection. We call these subpath weights the Path Correlation Data (PCD). In this manuscript we ask the following question: when can the underlying weighted graph be recovered knowing only the boundary vertices and the PCD? We establish necessary and sufficient conditions for a graph to be reconstructible from this information, and describe an algorithm to perform the reconstruction. Subject to fairly general conditions which will be elaborated in next Section, the results applies to directed graphs with asymmetric edge weights, and accommodates paths arising from asymmetric routing in the underlying communication network. We also describe the relationship between the graph produced by our algorithm and the true graph in the case that our conditions are not satisfied.

Establishing the conditions under which the underlying directed graph can be recovered exactly from the pairwise PCD, algorithmically, this enables us to consistently fuse tree-based view of the set of network paths to and from each endpoint to reconstruct the underlying network. However, in practice the PCD is not consistently determined by path measurements. Statistical fluctuations give rise to inconsistent inferred weight of edges from measurement based on different endpoints, as do operational constraints on synchronization, and deviations from the underlying packet transmission model. Furthermore, ad hoc solutions to eliminate noise, such as pruning small weight inferred links, are hard to apply in a consistent manner that preserves known end-to-end metric values. We further take a unified approach to the problem of inconsistent weight estimation. We formulate two types of inconsistency: *intrinsic*, when the weight set is internally inconsistent, and *extrinsic*, when they are inconsistent with a set of known end-to-end path metrics. In both cases we map inconsistent weights to consistent PCD within a least-squares framework.

Finally, we evaluate the performance of this mapping in composition with tree-based inference algorithms.

To Nina

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to my advisor Prof. Simon Foucart for his excellent guidance and valuable suggestions throughout the years I worked under his supervision.

With many thanks to my co-adviser Prof. Nick Duffield from electrical engineering department for introducing the problem and generous advise and financial supports during my Ph.D. program.

I would like to express my special thanks to Prof. Gregory Berkolaiko for all his guidance on this project. Thank you very much for your constant support and belief in me when I met roadblocks in my research.

With many thanks to my committee members Prof. Yalchin Efendiev and Prof. Anirban Bhattacharya for all their generous support and advice.

I would like to send my special thanks to Prof. Peter Howard for his unlimited support on my transfer to mathematics department. I always feel indebted to him for helping me start my study in this amazing program.

Last but not least, I would like to express my thanks to the department of mathematics at Texas A&M University and specially the professors who gave me the chance to learn many things from them.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Simon Foucart and Professors Gregory Berkolaiko and Yalchin Efendiev of the Department of Mathematics and Professor Nick Duffield of the Department of Electrical & Computer Engineering and Professor Anirban Bhattacharya of the Department of Statistics.

The work in Sections 2 and 3 was conducted in joint work with Professor Nick Duffield and Professor Gregory Berkolaiko.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

Graduate study was supported by a fellowship from Texas A&M University.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION AND PRELIMINARY

1.1 Network Tomography as an Inversion Problem

Network performance tomography seeks to infer edge metrics and even the underlying network topology by fusing measurements of streams of packets traversing a set of network paths. Abstractly, for additive metrics, e.g. mean packet delay and log transmission probabilities, a putative solution to the network tomography problem attempts to invert a linear relation between the set of path metrics $\mathcal{D}$ and the link metrics $\mathcal{W}$ in the form

$$\mathcal{D} = \mathcal{A}\mathcal{W} \tag{1.1}$$

Here $\mathcal{A}$ is the incidence matrix of links over paths, $\mathcal{A}_{P,l}$ is 1 if path P traverses link l, and zero otherwise. The linear system (1.1) is generally underconstrained in real-life networks, and hence does not admit a unique solution [9]. To overcome this deficiency, one approach has been to impose conditions on the possible solutions, typically through sparseness, effectively to find the "simplest" explanation of the observed path metrics; see [12, 2]. A different approach in the similar problem of traffic matrix tomography has been to reinterpret (1.1) as applying to bi-modal measurements of packet and bytes counts [17], or of empirical means and variances then imposing constraints between these based on empirical models [19, 20]. However, the high computational complexity of this approach makes it infeasible for real-world communications networks [21], although quasi-likelihood methods offer some reduction in complexity [16]. A related approach known as Network Kriging seeks to reduce dimensionality in the path set by assumption on prior covariances [9].

In practice, both the measurement and inference functions are distributed across a set of network hosts. Each host performs inference from packet measurement on the subset of network paths of which it is an endpoint (i.e. the source or the destination of measurement packets). The inference produced by each host takes the form of a *logical* weighted subgraph that estimates the spanning graph of the paths terminating at that host, which each logical edge representing a subpath comprising of one or more edges in the underlying network, with a weight corresponding to the aggregate performance metric on that subpath. Hosts exchange these inferred subgraphs or raw packet measurement data with other hosts, or transmit these to a central location where they are fused to perform network-wide inference.

1.2 Network Model and the Notion of Partial Network Graphs

We represent the communication network by a directed edge-weighted graph $G = (V, E, \mathcal{W})$ with vertices V, edge set E, a single edge-based non-negative metric $\mathcal{W} : E \to \mathbb{R}_{\geq 0}$. Edges in E represent links between router identified with vertices, while the weights represent packet performance metrics associated with each edge. A *partial network graph* $\mathcal{G} = (G, V_B, \mathcal{P})$ consists of the graph G together with a set V_B of boundary vertices and the set $\mathcal{P}$ of directed paths $P_{u,v}$ between *some* ordered pairs (u, v) of boundary vertices. We shall call u the *source* and v the *receiver* associated with the path $P_{u,v}$. In the context of network tomography, the boundary nodes V_B act as the sources and sinks of measurement packets that traverse the network on the paths in $\mathcal{P}$. The remaining vertices $V_I = V \setminus V_B$ will be called the interior vertices. The notion of partial network graph will be used to describe both the underlying network (in which case it is often not "partial": there is a path in $\mathcal{P}$ between each ordered pair of boundary vertices) and various inferred networks (in which case it is "partial" due to the limited data available to the agent preforming inference). In the latter case, the topology of $G = (V, E, \mathcal{W})$ may be known in advance, or it may itself be inferred from the measurements.

Fusion of partial subgraphs is a key task both at individual hosts and for network wide inference. For example, a common inference primitive involves a host correlating two end-to-end performance measurements collected from routes to a pair of remote hosts. The result is a logical spanning binary tree with a single interior vertex and three leaves. The root host then fuses the set of binary trees obtained by iteration over all remote host pairs, in order to infer logical tree spanning paths between itself and the other endpoints [25]. For network-wide inference, the set of such trees generated by all measurement hosts would be fused to infer the spanning logical performance network that connects them.

1.3 Inference and Fusion of Logical Network Subgraph

In this Section we describe scenarios for inference and fusion of *logical network subgraphs* in which our methods apply, and review prior work on the problem of subgraph fusion.

1.3.1 Logical Network Subgraphs in Tomography

The logical network subgraph associated with a partial network graph $(G, V_B, \mathcal{P})$ comprises the partial network graph $(G', V_B, \mathcal{P}')$ with $G' = (V', E', \mathcal{W}')$ defined as follows: V' is the union of V_B and the branch points of the path set $\mathcal{P}$ with V_I. $(u, v) \in E'$ if u, v lie on a directed path in $\mathcal{P}$ without another node in V' between them. $W'_{u,v}$ is the aggregate weight of directed edges in E that connect u to v along the unique path in $\mathcal{P}$ that joins them. $\mathcal{P}'_{u,v} \in \mathcal{P}'$ comprises the edges in E' that are subpaths in $\mathcal{P}_{u,v}$. In the following scenarios, we assume the full physical network $G = (V, E^{(0)}, \mathcal{W})$ together with a maximal set of possible boundary vertices $V_B^{(0)}$ and routes $\mathcal{P}^{(0)}$ between them. For one or more boundary subsets $V_B \subset V_B^{(0)}$ and their connecting routes $\mathcal{P} \subset \mathcal{P}^{(0)}$, tomography is used to estimate the logical network subgraph $(G', V_B, \mathcal{P}')$ associated with the partial network graph $(G, V_B, \mathcal{P})$. Our methods then concern how to consistently fuse the resulting set of logical subgraphs.

1.3.2 Multicast Tomography

In multicast tomography, a sequence of multicast probe packets is dispatched from a boundary source along a multicast tree. Successful receipt and transmission latency are recorded for each packet at each receiver. Maximum likelihood estimators for loss [26] and discretized delay distribution [33] are computed for logical edges under independence assumptions for loss and delay on edges. If the topology is not known, each logical source tree is recovered by recursive clustering in which vertices with the largest common path loss or delay weight are identified as siblings [29]; see Section 1.3.4 for further details. Recovery on the edge weights in a known multicast topology by fusing packet level measurement from trees is proposed in [24]. Multicast inference exploits the inherent correlation of multicast packets, each of which occurs once per edge, with copies propagated from each branch point. In order to avoid the requirement for multicast probing, in the work [31] probing based on sets of back-to-back unicast packets sent to pairs of receivers have been proposed. It has been shown that this unicast way of probing approximates multicast one since each unicast packet in a set, experiences similar performance on their common path portion.

1.3.3 Binary Covariance-Based Tomography

Let independent random variables $\mathcal{X} = \{x_e : e \in E\}$ be associated with each edge and set $X_P = \sum_{e \in P} x_e$ for any subpath P. Then using the independence assumption, the following equality holds for the covariance of two paths metric and the variance of their intersection

$$\mathsf{Cov}(X_P, X_{P'}) = \mathsf{Var}(X_{P \cap P'}) \tag{1.2}$$

Hence any unbiased estimator of the path covariance $\mathrm{Cov}(X_P, X_{P'})$ is an unbiased estimator of the additive path metric $\mathrm{Var}(X_{P''})$ where $P'' = P \cap P'$. This approach was first proposed for multicast probing in [30] where each packet corresponds to an instance of $\mathcal{X}$. A different interpretation considers each instance of $\mathcal{X}$ to represent the edge metric values in force during a time slot. Any flow of packets traversing edge e in a given time slot is assumed to experience performance governed by the same metric value. Hence the window-average performance experienced by distinct flows of packets (even unicast) will be correlated, and more so for larger packet set. Assuming the instances of X are drawn i.i.d. over different time slots, then (1.2) allows estimation of $\mathsf{Var}(X_P)$ from measurement of unicast packet on distinct end-to-end paths with intersection P; [34].

1.3.4 Tree Reconstruction from Binary Primitives

As briefly referred to in Section 1.3.2, an unknown logical tree can be estimated by recursive clustering on leaf vertices based on largest estimated metric on their common path from the root; the same method can be applied to covariance-based estimates of Section 1.3.3. Denote by $X_{k,i}$ the data associated with packet measurements along the path $\mathcal{P}_{k,i}$ and estimate $\widehat{m}(X_{k,i}, X_{k,j})$ of the aggregate metric the intersection $\mathcal{P}_{k,i} \cap \mathcal{P}_{k,j}$. The pair i, j of maximal $\widehat{m}$ are identified as siblings with common parent denoted $\{i, j\}$ while i and j are removed from further consideration. A merged measurement $X_{k,\{i,j\}}$ is then associated–in a metric dependent manner–with the parent, and the process performed recursively until the root is reached. In multicast loss inference $X_{k,i}$ comprises a bitmap indicating which packets reached i from k, while for covariance based estimators and general vertex clusters A and B, $X_{k,A\cup B}$ is a convex combination of $X_{k,A}$ and $X_{k,B}$, with coefficients chosen e.g. to minimize estimation variance [30]. Due to statistical node, a non-binary node in the underlying network is typically resolved as a set if binary nodes which may then be amalgamated by pruning edges of small inferred weight; see Section 1.3.6 below.

1.3.5 Subgraph Fusion

Prior works have investigated the problem of how to fuse topographically inferred subgraphs. [38] fuses quartets (2-source 2-receiver inferred subgraphs) which have been shown to be sufficient to characterize an M-source, N-receiver network from which they are derived [36]. However, information concerning metric edge weights is not exploited in this approach, and so the issue of consistency between different measurements of the same path does not arise. These papers were principally concerned with fusing measurements obtained with striped unicast probes, while our present work is largely agnostic on the measurement mechanism. The Occam system [37] exploits the idea of binary tree primitives to form source based trees which then fused through an optimization problem where a network compatible with the source trees and minimizer over the number of edges and host to host distances is selected. Most recently [23] proposed fusing source and destination based trees derived from binary tree primitives using passive traffic measurements. The key idea is that each path $\mathcal{P}_{u,v}$ from boundary vertex u to v occurs in both the source tree rooted at u and the receiver tree rooted at v. This allows placement of interior vertex on the path according to metric values. However, consistent placement requires equality of the total weight $W_{u,v}$ in each tree, a property that does not hold for estimated weights. Earlier work [28] used detailed timing and packet order information from multiple probe sources to infer overlaps between sourced based trees. Networking tools such as ping and traceroute tools can in principle be used to detect the presence of a given responsive router on distinct paths, although non-responsiveness in encrypted networked and ambiguity due to distinct interface address often limit the utility of this approach.

1.3.6 Tree Pruning

As discussed above, pruning of low weight edges has been proposed to reduce topological noise arising from statistical measurement variability [25]. If path weight to be preserved, the weight of pruned edges should be ascribed amongst the remaining edges. One approach is to recompute tomographic weights on the pruned topology. This is well suited to multicast based inference since edge weight estimators exist in non-binary trees [26] and networks [24]. However, this approach does not generalize to unicast-based network inference. Consider an internal vertex v through which the paths from sources s_1 and s_2 reach subset R of boundary receivers. The weights of the paths (s_1, v) and (s_2, v) can be individually estimated based on convex combinations of primitive binary estimates from the logical binary tree with vertices $\{\{s_1, v, r_1, r_2\} : r_1 \neq r_2 \in R\}$ and $\{\{s_2, v, r_1, r_2\} : r_1 \neq r_2 \in R\}$ respectively. However, these estimates will not in general yield equal weights for the edges $\{(v, r) : r \in R\}$. This motivates our approach to find weights that are extrinsically consistent (with the path weights) but close to the weights of surviving edges before pruning.

1.4 Problem Statements

The focus of this dissertation is to answer three main questions in the network tomography area summarized as follows:

(i) Under what conditions is the network identifiable in the sense that distinct values of network parameters (topology and edge metrics) can be distinguished in the limit of a large number of packet measurements?

(ii) What algorithms can identify the network parameters in this limit?

(iii) How are these algorithms best adapted to work with finite measurement data in the sense of being applicable and performing accurately?

The focus of **Section 2** is to answer the first and second questions above for a wide class of inference problems on networks with asymmetric paths between host pairs. Network level inference is performed by fusing source and destination based trees at each measurement host, characterized by their Path Correlation Data (PCD), namely the weight of the intersection of any two paths that share an origin or destination. Necessary and sufficient conditions for a network graph to be reconstructible from the PCD will be established together with an explicit reconstruction algorithm. Thus when the PCD are identifiable from path pair measurements, the full network is identifiable under the reconstruction conditions.

The practical network measurement may provision data with imperfect consistency. For example, input data may be provided in the form of weighted trees computed from packet measurement over time intervals that are not perfectly aligned, so that the metric of a path $\mathcal{P}(b, b')$ may be reported differently in the source and receiver trees which contains this path. Even with aligned intervals, deviations from the model and statistical fluctuations due to finitely many probe packets may result in inconsistency. In order to enable our proposed algorithm (presented in Section 2) to operate with such data, we propose to compute a PCD that is a least-squares fit to inconsistent tree data. This extension and associated error sensitivity analysis is the subject of **Section 3**.

The focus of **Section 4** is to provide a composite application of our methods to the problem of network inference through model-based simulations. We compare its performance in correctly identifying subset of network paths that experience performance degradation of a common internal edge.

2. GRAPH RECONSTRUCTION FROM PATH CORRELATION DATA *

2.1 Introduction

2.1.1 Network Model and Partial Network Graphs

We start with formal definition of communication network and the notion of partial network graphs. A communications network can be abstracted as a directed edge-weighted graph $G = (V, E, \mathcal{W})$ be with vertices V, edge set E, a single edge-based non-negative metric $\mathcal{W} : E \to \mathbb{R}_{\geq 0}$. Edges in E represent links between router identified with vertices, while the weights represent packet performance metrics associated with each edge. We do not assume edge symmetry: $(u, v) \in E$ does not imply $(v, u) \in E$, or weight symmetry: $w_{u,v}$ and $w_{v,u}$ need not be equal. A *partial network graph* $\mathcal{G} = (G, V_B, \mathcal{P})$ consists of the graph G together with a set V_B of boundary vertices and the set $\mathcal{P}$ of directed paths $P_{u,v}$ between *some* ordered pairs (u, v) of boundary vertices. We shall call u the *source* and v the *receiver* associated with the path $P_{u,v}$. An important example of a partial network graph is the *source tree* $\mathcal{T}_b^S$ of a boundary vertex $b \in V_B$. This is the minimal subgraph supporting the path set $\mathcal{P}(\mathcal{T}_b^S) = \{P_{b,v} \colon v \in V_B\}$. Correspondingly, the *receiver tree* $\mathcal{T}_b^R$ is induced by the path set $\{P_{v,b} \colon v \in V_B\}$. In the context of network tomography, the boundary nodes V_B act as the sources and sinks of measurement packets that traverse the network on the paths in $\mathcal{P}$. The remaining vertices $V_I = V \setminus V_B$ will be called the interior vertices. While the paths may be the smallest-weight paths through the network, in general we do not make this assumption.

The performance metrics are considered additive in the sense that the performance metric of the path $P_{u,v}$ is the sum $W_{u,v} = \sum_{e \in P_{u,v}} w_e$. The same notation extends to sums over subpaths that terminate at interior vertices. Examples of additive performance metrics include packet delay, and negative log transmission probability. We will assume one can measure the length of the following: the path between any two boundary nodes, the common part of the two paths from a boundary

*Reproduced with permission from "Graph Reconstruction From Path Correlation Data" by G. Berkolaiko, N.G. Duffield, M. Ettehad, and K. Manousakis, 2018. Inverse Problems, 35, 1-25, Copyright 2018 by IOP Publishing. All rights reserved.

node to any two other boundary nodes, as well as the common part of the paths to a boundary node from any two other boundary nodes. We abstract both these cases into a unified data model in which for every triple b, b_1 and b_2 of boundary vertices, we can measure, via covariances of the packet statistics, the metric of the intersection $P = \mathcal{P}(b, b_1) \cap \mathcal{P}(b, b_2)$ as well as to the metric of the intersection $P = \mathcal{P}(b_1, b) \cap \mathcal{P}(b_2, b)$. The results of such measurements we will denote by $\mathrm{PCD}(b \prec b_1, b_2)$ and $\mathrm{PCD}(b_1, b_2 \succ b)$ correspondingly. We remark that we use the symbols $\prec$ and $\succ$ here not as binary comparison operators but as pictograms meant to evoke the topological structure of the corresponding pair of paths. The totality of such measurements we call the PCD. Note that we will not, in general, assume that the paths are symmetric; the path $\mathcal{P}(b, b_1)$ may be different topologically from the path $\mathcal{P}(b_1, b)$. As a consequence, the values $\mathrm{PCD}(b \prec b_1, b_2)$ and $\mathrm{PCD}(b_1, b_2 \succ b)$ are in general different. We will assume that the function PCD is measured exactly; see next Section for a detailed discussion of possible sources of error and the developed technique for error-correction.

Under the fairly general conditions that will be in force in this work, the problem has a natural formulation in terms of trees. First, assume that for each $b, b_1, b_2 \in V_B$, the intersection $\mathcal{P}(b, b_1) \cap \mathcal{P}(b, b_2)$ is connected. Second, assume that the metric $\mathcal{D}$ is path increasing, i.e., $\mathcal{D}(\mathcal{P}) < \mathcal{D}(\mathcal{P}')$ for $\mathcal{P} \subsetneq \mathcal{P}'$. As shown in [29], the quantities $\{\mathrm{PCD}(b \prec b_1, b_2) : b_1, b_2 \in V_B\}$ give rise to an embedded logical weighted tree rooted at b (called “source tree”). The tree is computed iteratively by finding node pairs (b_1, b_2) of maximal $\mathrm{PCD}(b \prec b_1, b_2)$ and identifying each such pair with a branch point in the logical tree. Each logical link is assigned a weight equal to the difference between the values of PCD$(b \prec \cdot, \cdot)$ associated with its end points. Similarly, the quantities $\mathrm{PCD}(b_1, b_2 \succ b)$ give rise to an embedded logical tree with a single receiver b and the sources $V_B \setminus \{b\}$. Such trees are called “receiver trees”. Under the assumed conditions, our problem can be restated as how to recover the underlying weighted graph from the set of logical sources and receiver trees rooted at every $b \in V_B$.

The main contribution of this Section, Theorem 2.2.1, is to show that under natural conditions, a weighted directed graph $\mathcal{G}$ can be recovered knowing only the graph's Path Correlation Data (PCD). In summary, the conditions under which Theorem 2.2.1 holds are (i) each edge is traversed by at least one path in $\mathcal{P}$; (ii) each non-boundary vertex is *nontrivial* in the sense that in-degree and out-degree are not both equal 1; and (iii) each non-boundary vertex x is *nonseparable* in the sense that the set of paths $\mathcal{P}(x) \subset \mathcal{P}$ that pass through x cannot be partitioned into two or more subsets with non intersecting end point sets. Our result holds without the assumption of weight symmetry (defined as requiring the existence of the reverse of any edge in G, having the same weight) or path symmetry (defined as the paths in either direction between two boundary nodes traversing the same set of edges). We prove the correctness of our reconstruction algorithm (Algorithm 1) under the stated assumptions.

As stated our solution does not assume that link weights are symmetric. Neither do we assume that paths in either direction between two endpoints are symmetric: they are not required to traverse the same set of internal (i.e. non-boundary) vertices. This level of generality reflects networking practice, in which non-symmetric routing is employed for policy reasons including performance and revenue optimization [18]. However, in the cases where symmetric paths can be assumed *a priori*, this knowledge enlarges the set of reconstructible networks. We therefore pay special attention to this case, providing alternative definitions of the *nontrivial* and *nonseparable* vertices and a separate proof of the correctness of Algorithm 1 (which requires a one-line change). We also establish the correctness of a second, more customized Algorithm 2 which applies only in the case of symmetric weights and paths, and which is computationally less expensive than Algorithm 1 applied to this case.

2.1.2 An Example of a Communication Network

To illustrate the information available to an observer in our model, consider the network graph schematically shown in Fig. 2.1. It is assumed that the end-to-end measurements are possible among the boundary vertices $V_B = \{b_1, ..., b_6\}$. The three versions of the same graph shown contain information about the paths between the given source (b_1, b_2 and b_3, correspondingly) and the corresponding set of receivers $V_B \setminus \{b_i\}$ for $i = 1, 2, 3$; the links belonging to these paths are highlighted in thicker lines.

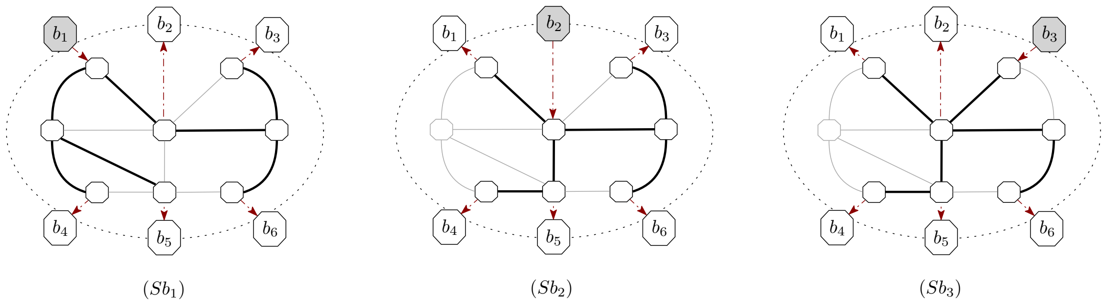


Figure 2.1: Alternative selections of the source and the corresponding routing paths (bold edges).

From the point of view of an external observer, the routing paths on the graph are hidden but can be reconstructed, to a certain extent, by the measurements with a fixed source and alternating receivers, represented in our setting by queries to the PCD function. As Fig. 2.2 shows, the trees reconstructed from PCD are *logical* trees where the edges represent the amalgamated versions of the actual physical edges. For example, the logical tree labeled (Sb_3) has a direct edge from b_3 to b_6, whereas the actual route, shown in (Sb_3) passes through an internal node. Since this node does not feature as a junction in the tree (Sb_3), it will not be detected from the PCD. Moreover each internal vertex in the original graph has multiple appearances in the logical trees with no identifying information attached to them. Correctly identifying multiple representations of the same internal node will be the central challenge of this work.

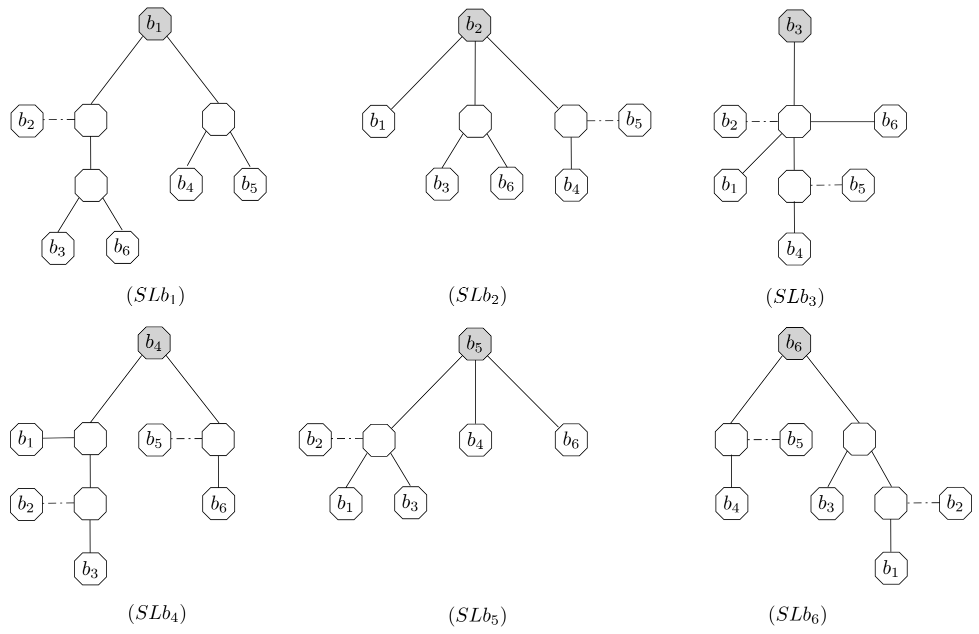


Figure 2.2: Representation of PCD on the network graph through the set of observed logical source trees for sources b_1 to b_6.

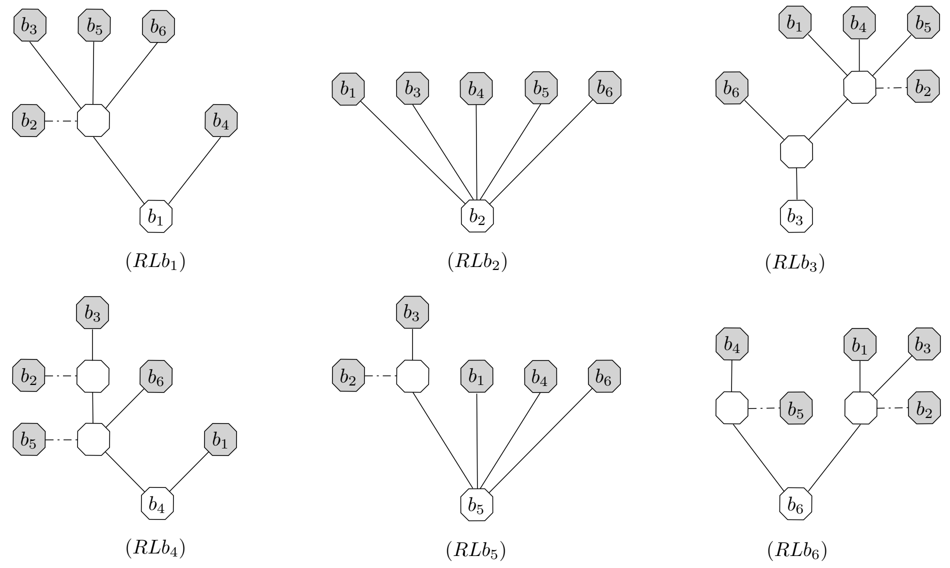


Figure 2.3: Representation of PCD on the network graph through the set of observed logical receiver trees for receivers b_1 to b_6.

The information form logical source trees (Fig. 2.2) is only enough for reconstruction of special class of network graph, namely the symmetric one where both the routing and edge weights are symmetric. When this is not the case, the measurements of the form PCD$(b_1, b_2 \succ b)$ will be essential to reconstruct *logical receiver trees* shown in Fig. 2.3. Those contain information about the paths with the selected receiver b and with the source set $V_B \setminus \{b\}$. To summarize, the main goal is to develop a practical algorithm to reconstruct the original graph from set of measurements information of the form PCD$(b_1 \prec b_2, b_3)$ and PCD$(b_2, b_3 \succ b_1)$ for $b_1, b_2, b_3 \in V_B$ and to establish necessary and sufficient conditions for the successful reconstruction.

2.2 Problem Statement and the Main Result

In this Section we set up our model and formulate our results. Informally, we have a graph with some metric assigned to directed edges and a subset of vertices that is declared to be the "boundary" (denoted by b_1, b_2 etc) where the observations are made. We assume there is a fixed path ("route") between each ordered pair of boundary vertices ("source" to "receiver"). We further assume that the metric is such that we can measure the length of any route and also the length of the common part of any two routes starting at the same source or any two routes coming into the same receiver. These assumptions and the reconstruction problem are made precise below.

Our standing assumptions concerning $\mathcal{P}$ are

(i) *Uniqueness*: for any given pair (u, v) in V_B, there is at most one path $P_{u,v} \in \mathcal{P}$ connecting them in that direction.

(ii) *No interior boundaries:* no $v \in V_B$ is an interior node of any path in $\mathcal{P}$. A partial network without this property can be modified by replacing such v by an interior node to which it is connected with a zero weight edge; see e.g. Figure 5 in [23]).

(iii) *Path consistency*: if two vertices u and v appear in two paths P_{b_1,b_2} and $P_{b'_1,b'_2}$ in the same order, then the subpaths connecting u and v in P_{b_1,b_2} and $P_{b'_1,b'_2}$ are identical (Note that path consistency implies tree consistency of [23]).

2.2.1 Problem Setup

Definition 1. A network graph $\mathcal{G} = (G, V_B, \mathcal{P})$ is an edge-weighted graph $G = (V, E, \mathcal{W})$ together with a set $V_B \subset V$ of *boundary vertices* and the paths $\mathcal{P}$ between them. In detail,

(i) V is a finite set of vertices,

(ii) $E \subset V \times V$ is the set of directed edges (no loops or multiple edges are allowed),

(iii) $\mathcal{W} : E \to \mathbb{R}_+$ are the edge weights,

(iv) V_B is an arbitrary subset of V

(v) there is a path $\mathcal{P}(b, b')$ between any pair of boundary vertices $b \neq b'$; each path $\mathcal{P}(b, b')$ is simple and assumed to be fixed for the duration of observation of the network. The paths are assumed to have the *tree consistency property*: for any b_1, b'_1, b_2 and b'_2 in V_B the intersection $\mathcal{P}(b_1, b'_1) \cap \mathcal{P}(b_2, b'_2)$ is connected. As a special example see Fig. 2.4 where b_1, b_2 are the same source.

Figure 2.4: An example of the violation of the tree consistency property. The paths $\mathcal{P}(b, b_1)$ and $\mathcal{P}(b, b_2)$ first diverge and then meet again.

We will assume that a path $\mathcal{P}(b, b')$ does not pass through any other boundary vertices. This is done for convenience only; a graph can be easily made to satisfy this condition by "drawing out" the boundary vertices from the bulk of the graph as shown in Fig. 2.5. The vertices that do not belong to the boundary we will call *internal vertices* and use the notation $V^I = V \setminus V_B$.

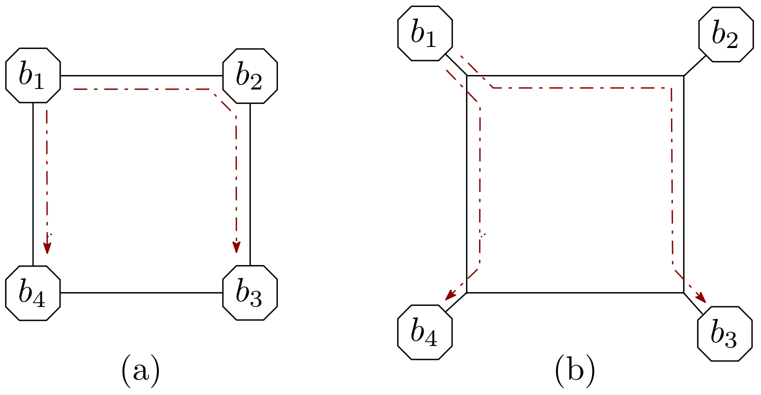


Figure 2.5: A graph (square with no diagonals) with the path $\mathcal{P}(b_1, b_3)$ going through b_2. The same graph with the boundary vertices drawn out.

We remark that we do not assume that the weights are symmetric: $\mathcal{W}_{x,y}$ is generally different from $\mathcal{W}_{y,x}$. We also do not need to assume that the paths are symmetric. However, since the latter case is important in applications and allows for a simplified reconstruction algorithm, we will devote some time to its separate treatment, in particular in Definition 7. Let us consider some implications of the tree consistency property. Consider two paths, $\mathcal{P}(b, b_1)$ and $\mathcal{P}(b, b_2)$ with some distinct $b, b_1, b_2 \in V_B$. The tree consistency property implies that the paths can be written as

$$\mathcal{P}(b, b_1) = [b, x_1, x_2, ..., x_j, y_1, y_2, ..., b_1] \tag{2.1}$$

$$\mathcal{P}(b, b_2) = [b, x_1, x_2, ..., x_j, z_1, z_2, ..., b_2], \tag{2.2}$$

where the vertex sets $\{y_1, y_2, ...\}$ and $\{z_1, z_2, ...\}$ are disjoint, see Fig. 2.6(a).

Similarly tree consistency property applied to paths $\mathcal{P}(b_1, b)$ and $\mathcal{P}(b_2, b)$ implies that

$$\mathcal{P}(b_1, b) = [b_1, y_1, y_2, ..., y_i, x_1, x_2, ..., b] \tag{2.3}$$

$$\mathcal{P}(b_2, b) = [b_2, z_1, z_2, ..., z_j, x_1, x_2, ..., b], \tag{2.4}$$

with $\{y_1, y_2, ..., y_m\}$ and $\{z_1, z_2, ..., z_n\}$ are disjoint, see Fig. 2.6(b).

Definition 2. The vertex x_j in equations (2.1)–(2.2) is called the $(b \prec b_1, b_2)$-junction. Note that the set $\{x_1, ..., x_j\}$ is allowed to be empty in which case b acts as the junction. The vertex x_1 in equations (2.3)–(2.4) is called the $(b_1, b_2 \succ b)$-junction.

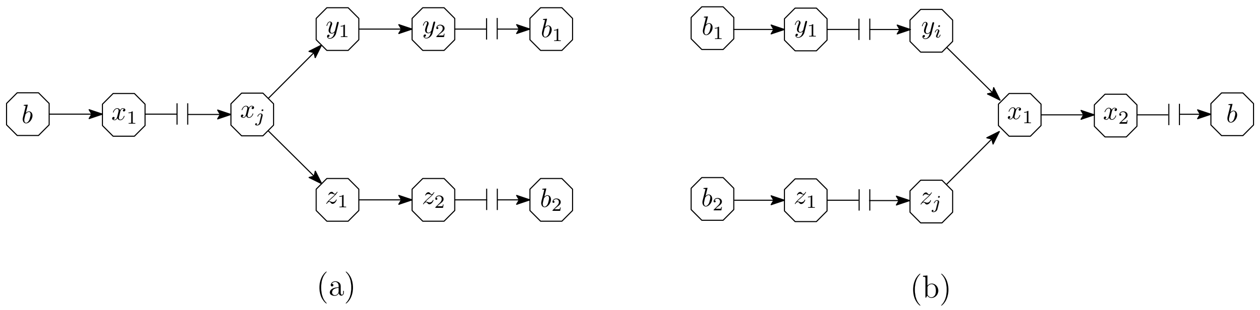


Figure 2.6: (a) $x_j = (b \prec b_1, b_2)$-junction, (b) $x_1 = (b_1, b_2 \succ b)$-junction.

We remark that we use symbols $\prec$ and $\succ$ not as relational operators but as separators in the list of 3 vertices which are pictorially similar to the path configurations in Figure 2.6. To specify the graph reconstruction problem we will be solving we need to define the set of *measurements* available to us. The length of a path is defined as the sum of the weights $\mathcal{W}$ of its edges; we will denote the length by $|\cdot|$. We consider a single vertex as a zero-length path; the length of an empty set is also set to be zero. This allows us to assign length to an intersection of two paths between boundary vertices which is either empty or a single vertex or a connected subpath.

The totality of the measurements we can make will be called the Path Correlation Data (PCD). In includes, for all $b, b_1, b_2 \in V_B$,

(i) the length $|\mathcal{P}(b, b_1)|$,

(ii) the length $|\mathcal{P}(b, b_1) \cap \mathcal{P}(b, b_2)|$, which we will denote by $\mathrm{PCD}(b \prec b_1, b_2)$,

(iii) the length $|\mathcal{P}(b_1, b) \cap \mathcal{P}(b_2, b)|$, which we will denote by $\mathrm{PCD}(b_1, b_2 \succ b)$.

Thus we can directly measure the distance from $b \in V_B$ to any $(b \prec b_1, b_2)$-junction, or from the $(b_1, b_2 \succ b)$-junction to b. We can also infer the distance from the $(b \prec b_1, b_2)$-junction to b_1, or from b_1 to the $(b_1, b_2 \succ b)$-junction, see Fig. 2.7.

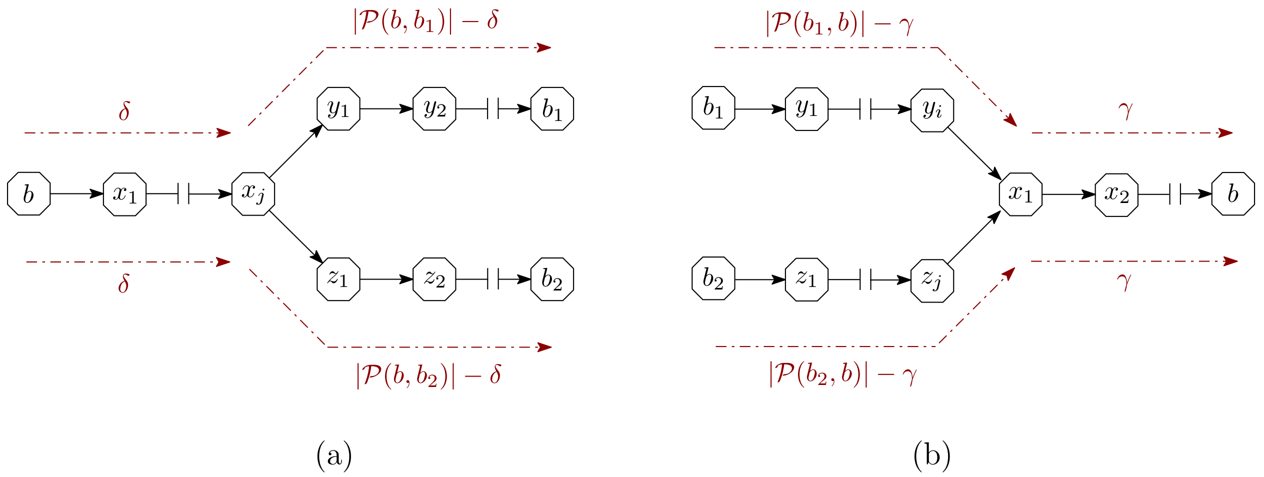

$|\mathcal{P}(b,b_1)| - \delta$
δ
y_1
y_2
b_1
b
x_1
x_j
δ
z_1
z_2
b_2
$|\mathcal{P}(b,b_2)| - \delta$
(a)
$|\mathcal{P}(b_1,b)| - \gamma$
b_1
y_1
y_i
γ
x_1
x_2
b
b_2
z_1
z_j
γ
$|\mathcal{P}(b_2,b)| - \gamma$
(b)

Figure 2.7: Various distances we can measure from Path Correlation Data (PCD). Here $\delta = \mathrm{PCD}(b, \prec b_1, b_2)$ and $\gamma = \mathrm{PCD}(b_1, b_2 \succ b)$.

Our principal question is thus: *Which network graphs $(\mathcal{G}, V_B, \mathcal{P})$ can be reconstructed from their path correlation data and how does one accomplish this?*

2.2.2 Some Obvious Necessary Conditions

Before we state our result and the associated reconstruction algorithm, let us consider examples that show some obvious necessary conditions we need to impose on the network graph $\mathcal{G} = (G, V_B, \mathcal{P})$ in order to be able to reconstruct it.

Example 1. Consider the network graphs in Fig. 2.8, with $V_B = \{u, v, w\}$ and with the routing paths indicated by dashed lines. None of the routing paths pass through the edge $e = (u, w)$ therefore the length of this edge cannot influence the PCD in any way. Conversely, the length of the edge e (and even its existence) cannot be inferred from the PCD.

Example 2. Consider the network graphs in Fig. 2.9 with boundary vertex set $V_B = \{u, w\}$. In

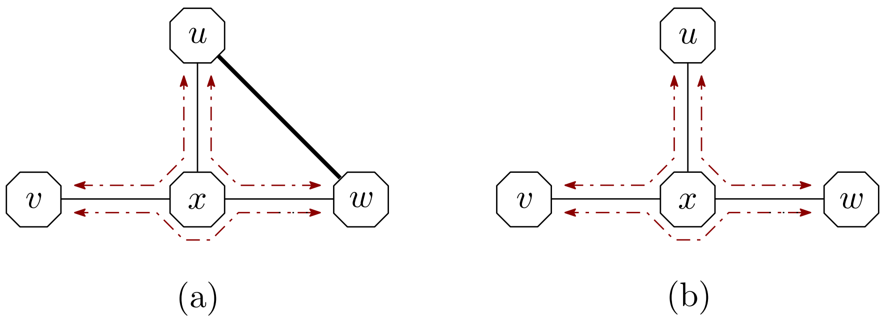


Figure 2.8: Failure to recover the edge $e = (u, w)$: the graphs (a) and (b) will produce the same PCD since none of the paths pass through the extra edge.

.

the left graph the length of the edge (x, u) will never appear in the PCD on its own, without being added to the length of the edge (x, w). Therefore, it will be impossible to reconstruct the location of the vertex x, and even detect it at all. This will be the case for any internal vertex of degree 2.

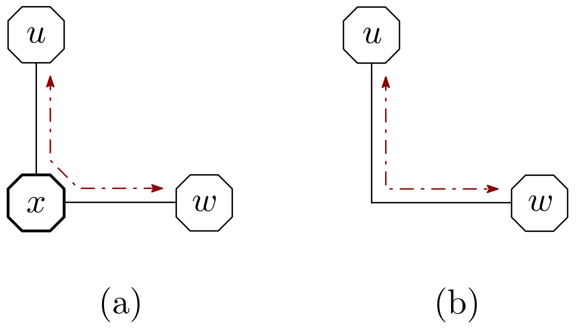
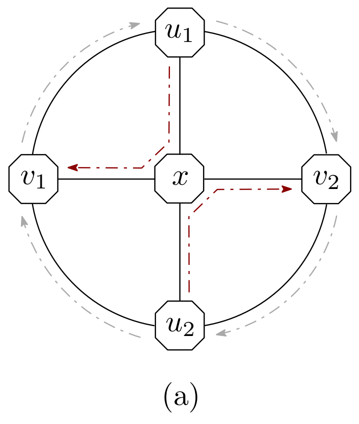


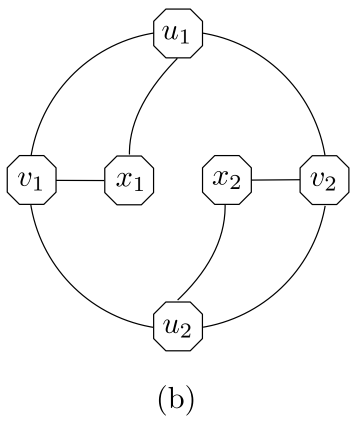
Figure 2.9: Failure to recover the internal vertex x: the graphs (a) and (b) will produce the same PCD as long as the sum of the lengths of (u, x) and (x, w) in the graph (a) is equal to the length of the edge (u, w) in the graph (b).

Example 3. Consider the network graphs in Fig. 2.10 with the boundary vertex set

$$V_B = \{u_1, v_1, u_2, v_2\}. \tag{2.5}$$

In Fig. 2.10(a) the paths $\mathcal{P}(u_1, v_1)$ and $\mathcal{P}(u_2, v_2)$ intersect at internal vertex x, while in Fig. 2.10(b) they do not have any vertices in common. However, the two graphs will produce the same PCD and will be indistinguishable.





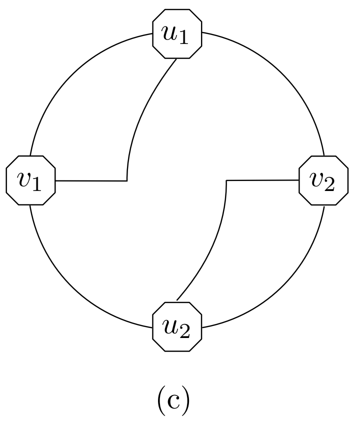
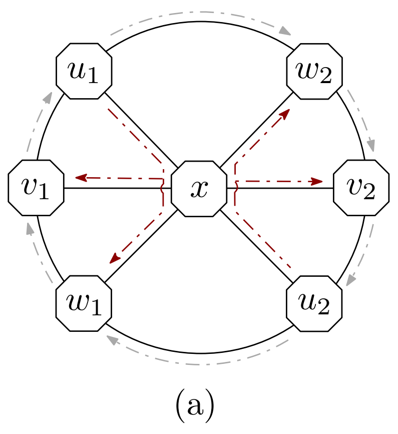


Figure 2.10: Failure to recover the integrity of the vertex x: the graphs (a), (b) and (c) will produce the same PCD since the path between u_1 and v_1 is in no way correlated to the path between u_2 and v_2.

The reader will undoubtedly observe that the vertices x_1 and x_2 in Fig. 2.10(b) will not be detected, and the graph in Fig. 2.10(c) is the "minimal" graph which will have the same PCD. By making the graph structure more complicated one can easily construct an example where x, x_1 and x_2 will act as junctions for some pairs of paths and thus will be detectable, see Fig. 2.11, but the two graphs are still not distinguishable from their PCD.



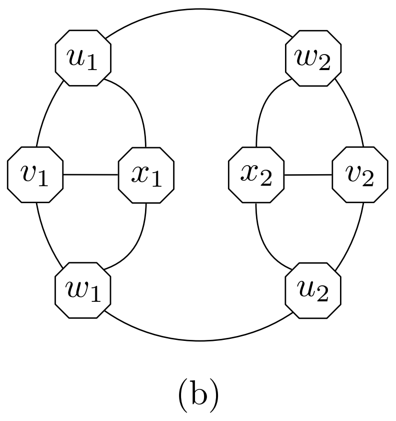


Figure 2.11: A more complicated example of failure to recover the integrity of the vertex x.

Thus the real problem is that in the left graph in Fig. 2.11 there are two families of paths going through the internal vertex x that do not interact in any way.

2.2.3 Statement of the Main Result

The main result of this Section is that the necessary conditions illustrated by examples in Section 2.2.2 are in fact sufficient for the reconstruction! We start by formalizing (and naming) the conditions we observed.

Definition 3. An edge is called *unused* if there is no path in $\mathcal{P}$ containing it.

We remark that if there are no unused edges in a network graph, each internal vertex has at least one incoming and at least one outgoing edge.

Definition 4. An internal vertex x is called *trivial* if it has only one incoming and only one outgoing edge (i.e. edges of the form (y_1, x) and (x, y_2) correspondingly).

We remark that if there are no unused edges, then there are at least two paths through every non-trivial internal vertex.

Definition 5. For an internal vertex $x \in V^I$, let $S_x \subset V_B$ to be the set of the sources and $R_x \subset V_B$ be the set of the receivers whose paths pass through x. More precisely,

$$S_x = \left\{b \in V_B : \exists \widehat{b} \in V_B,\ x \in \mathcal{P}(b, \widehat{b})\right\}$$
$$R_x = \left\{\widehat{b} \in V_B : \exists b \in V_B,\ x \in \mathcal{P}(b, \widehat{b})\right\}.$$

Definition 6. A vertex $x \in V^I$ is called *separable* if there are disjoint non-empty partitions $S_x = S_x^1 \cup S_x^2$ and $R_x = R_x^1 \cup R_x^2$ with the property that

$$b \in S_x^j,\ \widehat{b} \in R_x^{j'} \text{ with } j \neq j' \quad \Rightarrow \quad x \notin \mathcal{P}(b, \widehat{b}). \tag{2.6}$$

An example of a separable vertex is shown in Fig. 2.12. The partition sets here are $S^1 = \{b_1\}$, $S^2 = \{b_2\}$, $R^1 = \{b_3, b_4\}$ and $R^2 = \{b_5, b_6\}$.

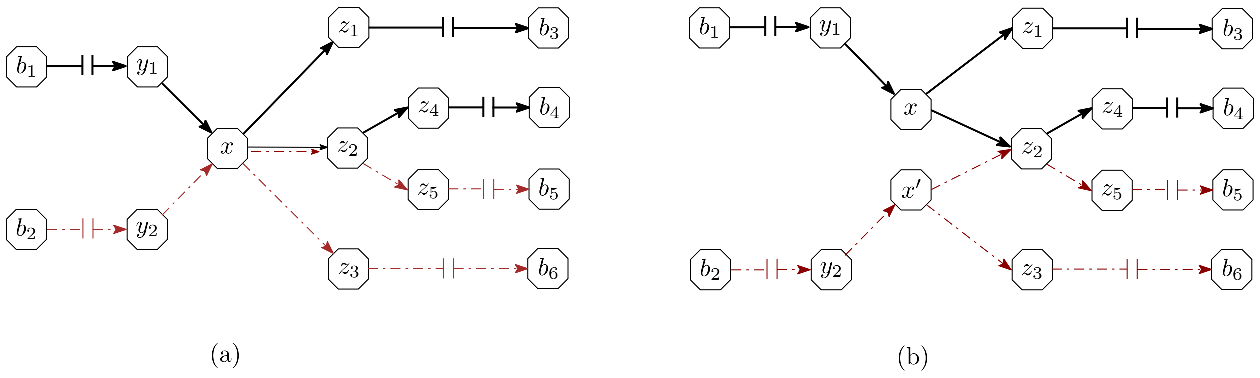


Figure 2.12: Paths through a separable vertex x and its separation into x_1 and x_2.

Finally, if the graph has symmetric routing we need to modify the conditions slightly but the resulting reconstructibility theorem will stay the same. Naturally, symmetric routing is an extra piece of information and more graphs are reconstructable in this case. The natural setting for the problem with symmetric routing is a non-directed graph, but since we still allow non-symmetric edge weights, we will keep the edges directed. As a result, edges come in pairs which correspond to undirected edges splitting into two directed ones. This is formalized in the definition of a "network graph with symmetric routing" below.

Definition 7. We will say that the network graph $\mathcal{N} = (G, V_B, \mathcal{P})$ has *symmetric routing* if

- for all $x, y \in V$, $(x, y) \in E$ implies $(y, x) \in E$ and
- for all $b, b' \in V_B$, the path $\mathcal{P}(b, b')$ is the reversal of the path $\mathcal{P}(b', b)$, namely

$$\mathcal{P}(b, b') = [b, x_1, x_2, \dots, x_j, b'] \quad \Rightarrow \quad \mathcal{P}(b', b) = [b', x_j, x_{j-1}, \dots, x_1, b]. \tag{2.7}$$

Definition 8. A vertex $x \in V^I$ in a network graph with symmetric routing is *trivial* if it has two (or less) adjacent vertices. A vertex $x \in V^I$ in a network graph with symmetric routing is *separable* if there is a disjoint partition $S_x = S_x^1 \cup S_x^2$ so that

$$b_1 \in S_x^1,\ b_2 \in S_x^2 \quad \Rightarrow \quad x \notin \mathcal{P}(b_1, b_2). \tag{2.8}$$

We can now state our Main Theorem. We stress that the statement of the theorem applies uniformly to network graphs with or without symmetric routing, the differences being absorbed by the definitions above. We will still need to provide two separate (but similar!) proofs.

Theorem 2.2.1 (Main Theorem)**.** Let $(G, V_B, \mathcal{P})$ be a network graph. If

1. no edge $e \in E$ is unused,
2. no $x \in V^I$ is trivial,
3. no $x \in V^I$ is separable,

then $(G, V_B, \mathcal{P})$ is uniquely reconstructable from its PCD. To put it more generally, in every class of network graphs with the same PCD, there is a unique network graph which satisfies the above conditions.

The theorem will be proved constructively, by presenting a reconstruction algorithm and verifying its result. The second part, which posits not only uniqueness but also the existence of the reconstructed graph, means, in practical terms, that even when given PCD from a graph that does not satisfy the conditions, the algorithm will terminate and produce a "nearby" result which does.

2.2.4 Comments on the Algorithm for Non-Symmetric Routing

The algorithm for the case of non-symmetric routing (Algorithm 1) works by discovering the internal vertices and reconstructing the routing paths in the format

$$\mathcal{R}(b, \widehat{b}) = [(b, 0),\ (x_1, \delta_1),\ (x_2, \delta_2), \ldots,\ (\widehat{b}, \delta)], \tag{2.9}$$

Algorithm 1 Reconstruction of the network graph $(G, V_B, \mathcal{P})$

```
1: for $b_1, b_2 \in V_B$ do                                        ▷ Initialization
2:     $\mathcal{R}(b_1, b_2) = [(b_1, 0), (b_2, |\mathcal{P}(b_1, b_2)|)]$
3: end for
4: for $b_1, b_2, b_3 \in V_B$ do                                   ▷ Main Loop
5:     create label $a$
6:     $\delta = \mathrm{PCD}(b_1 \prec b_2, b_3)$
7:     UPDATEPATH($\mathcal{R}(b_1, b_2), a, \delta$)
8:     if "symmetric routing" then $a' = a$ else create label $a'$
9:     $\delta' = |\mathcal{P}(b_2, b_1)| - \mathrm{PCD}(b_2, b_3 \succ b_1)$
10:    UPDATEPATH($\mathcal{R}(b_2, b_1), a', \delta'$)
11: end for
12: Read off the graph from reconstructed paths $\mathcal{R}$.       ▷ Return the result

13: function UPDATEPATH($\mathcal{R}(u, v), a, \delta$)              ▷ Recursive Function
14:     if $\exists\, (\cdot, \delta) \in \mathcal{R}(u, v)$ then return
15:     insert $(a, \delta)$ into $\mathcal{R}(u, v)$
16:     $\delta' = |\mathcal{P}(u, v)| - \delta$
17:     for $z \in V_B$ do
18:         if $\mathrm{PCD}(u \prec v, z) \geq \delta$ then UPDATEPATH($\mathcal{R}(u, z), a, \delta$)
19:         if $\mathrm{PCD}(u, z \succ v) \geq \delta'$ then UPDATEPATH($\mathcal{R}(z, v), a, |\mathcal{P}(z, v)| - \delta'$)
20:     end for
21: end function
```

where δ_i is the cumulative distance from b to x_j along the path (naturally, $\delta = |\mathcal{P}(b, \widehat{b})|$). Once every path is complete, the edges can be read off as pairs of consecutive vertices appearing in a path. The internal vertices are discovered as junctions. The main difficulty lies in identifying different junctions that correspond to the same vertex. This is done by a depth-first search in the function UPDATEPATH.

The following comments might be in order. In lines 5–6 a is the label for the vertex that is the $(b_1 \prec b_2, b_3)$-junction and δ is the distance from b_1 to a along the path $\mathcal{P}(b_1, b_2)$. In lines 8–9 a' is the label for the $(b_2, b_3 \succ b_1)$-junction and δ' is the distance from b_2 to a' along the path $\mathcal{P}(b_2, b_1)$. If we know *a priori* that the routing is symmetric, the $(b_1 \prec b_2, b_3)$-junction and the $(b_2, b_3 \succ b_1)$-junction are the same vertex and can receive the same label.

Line 14 checks if there is already a vertex distance δ from b_1 (if there is, label a is unused). Finally, the loop starting on line 17 looks for any other paths that the vertex with label a must belong to (see Fig. 2.13). Here we rely heavily on the tree consistency property.

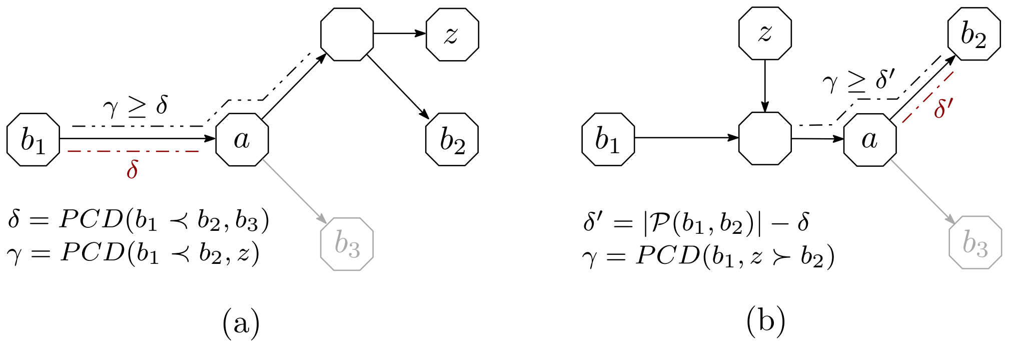

$\gamma \geq \delta$
b_1
a
z
b_2
b_3
δ
$\delta = PCD(b_1 \prec b_2, b_3)$
$\gamma = PCD(b_1 \prec b_2, z)$
(a)
z
b_2
$\gamma \geq \delta'$
δ'
b_1
a
b_3
$\delta' = |\mathcal{P}(b_1, b_2)| - \delta$
$\gamma = PCD(b_1, z \succ b_2)$
(b)

Figure 2.13: Insertion of a label a into the reconstructed paths $\mathcal{R}(b_1, z)$ and $\mathcal{R}(z, b_2)$ by lines 18 and 19 of Algorithm 1 correspondingly. The label a had been defined as the $(b_1 \prec b_2, b_3)$-junction prior to calling function UPDATEPATH with $u = b_1$ and $v = b_2$.

Some further code improvements are possible. Creating and then discarding unused labels can be avoided either by performing a check similar to line 14 in the main loop or, more elegantly, by making a an optional argument to the function UPDATEPATH and creating a label after line 14 if no a was supplied. Additionally, if the edge weights are symmetric, the call to PCD in line 9 can be avoided by using $\mathrm{PCD}(b_2, b_3 \succ b_1) = \mathrm{PCD}(b_1 \prec b_2, b_3) = \delta$.

Finally, a crude upper bound on complexity of the algorithm (in terms of label insertions into various $\mathcal{R}$) is $|V^I| \times |V_B|^2 \leq \frac{4}{27}|V|^3$ i.e. the product of number of internal vertices and the square number of boundary vertices of the graph.

2.3 Proof of the Reconstruction: Non-Symmetric Paths

The proof of Theorem 2.2.1 has three parts, with very similar arguments in each part. To facilitate the proof, we first state and prove an auxiliary lemma.

Lemma 2.3.1. Let x be an arbitrary *non-separable* internal vertex and let $A : V_B \times V_B \to \{\mathrm{T}, \mathrm{F}\}$

be a Boolean property (predicate) that is defined on the pairs (b, b') such that $x \in \mathcal{P}(b, b')$. Assume A is non-constant (i.e. true on some paths and false on some others). Define S_x^1 to be the set of the sources of the paths through x for which A is true and S_x^2 to be the set of the sources of the paths for which A is false. Define R_x^1 and R_x^2 analogously. More formally,

$$\begin{aligned} S_x^1 &= \left\{ b \in S_x : \exists b' \in R_x \left[x \in \mathcal{P}(b, b') \;\wedge\; A\,(b, b') \right] \right\}, \\ S_x^2 &= \left\{ b \in S_x : \exists b' \in R_x \left[x \in \mathcal{P}(b, b') \;\wedge\; \neg A\,(b, b') \right] \right\}, \\ R_x^1 &= \left\{ b' \in R_x : \exists b \in S_x \left[x \in \mathcal{P}(b, b') \;\wedge\; A\,(b, b') \right] \right\}, \\ R_x^2 &= \left\{ b' \in R_x : \exists b \in S_x \left[x \in \mathcal{P}(b, b') \;\wedge\; \neg A\,(b, b') \right] \right\}. \end{aligned} \tag{2.10}$$

Then $S_x^1 \cap S_x^2$ and $R_x^1 \cap R_x^2$ cannot both be empty. Above the notations $\wedge$ and $\neg$ stand for "and" and "not" logical conjunctions respectively.

Proof of Lemma 2.3.1. Since A is not always false, the sets S_x^1 and R_x^1 are non-empty; since A is not always true, S_x^2 and R_x^2 are non-empty. Furthermore, it is easy to see that

$$S_x^1 \cup S_x^2 = S_x \qquad \text{and} \qquad R_x^1 \cup R_x^2 = R_x. \tag{2.11}$$

Assume that $S_x^1 \cap S_x^2 = R_x^1 \cap R_x^2 = \emptyset$. Then we are in a position to use non-separability of the vertex x and to conclude that there is a path (without loss of generality) from $b_1 \in S_x^1$ to $b_2 \in R_x^2$. But this path either has property A or it does not. In the former case, $b_2 \in R_x^1$ and in the latter $b_1 \in S_x^2$, contradicting the assumption of disjointedness. $\square$

Proof of Theorem 2.2.1: unique reconstructability. We will now verify that, given the PCD from a graph that satisfies the conditions of the Theorem, the algorithm will produce the correct reconstruction. It is straightforward to check that the algorithm places a label for a vertex x only in the reconstructions of paths that actually contain x and with the right value of the cumulative distance δ. Thus it remains to show that

(i) every vertex x has at least one label created for it,

(ii) the reconstructed paths are not missing any vertices,

(iii) no more than one label is created for each vertex.

Then we will read off sequential pairs of vertices from the reconstructed paths to identify edges. Since there are no missing vertices in $\mathcal{R}$, the edges thus reconstructed will correspond to actual edges in E. By condition 1 of the theorem, every edge will appear in at least one $\mathcal{R}(u, v)$ and will therefore be reconstructed.

Let x be an arbitrary internal vertex. We would like to show the algorithm will create a label for x and place it in some reconstructed path containing x. Fix an arbitrary path through x and denote the edges that the path visits while going through x by $e_1 = (x_1, x)$ and $e_2 = (x, x_2)$. We will now use Lemma 2.3.1 with the property $A = A\,(b, b')$ defined as the statement "the path $\mathcal{P}(b, b')$ passes through *both* e_1 and e_2", or, in other words

$$A\,(b, b') = \text{“}\mathcal{P}(b, b') \text{ can be written as } [b, \ldots, x_1, x, x_2, \ldots, b']\text{”}. \tag{2.12}$$

There is at least one path on which A is true. Since x is non-trivial and each of its incident edges belongs to at least one simple path, there is at least one other path passing through x and *not* containing both e_1 and e_2. Therefore we can apply Lemma 2.3.1 and conclude that one of the pairs S_x^1 and S_x^2 or R_x^1 and R_x^2 are not disjoint.

Without loss of generality, consider a boundary vertex $b \in S_x^1 \cap S_x^2$. Let $\mathcal{P}_1(b, b_1)$ be a path containing both e_1 and e_2 and $\mathcal{P}_2(b, b_2)$ be a path that passes through x but does not contain both e_1 and e_2. Since both $\mathcal{P}_1$ and $\mathcal{P}_2$ contain b and x, they must coincide from b to x by the tree consistency property. Therefore, $\mathcal{P}_2$ contains e_1 and can not contain e_2. We conclude that $\mathcal{P}_1$ and $\mathcal{P}_2$ diverge exactly at x, see Fig. 2.14. In other words, x is the $(b \prec b_1, b_2)$-junction and a label will be created for it in the main loop. This label will be placed into $\mathcal{R}(b', b_1)$ unless there is already another label corresponding to x there.

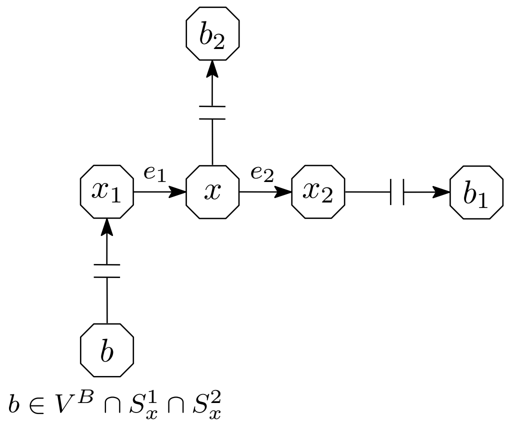


Figure 2.14: Two paths through a vertex x. The path $\mathcal{P}(b, b_1)$ contains both e_1 and e_2 and the $\mathcal{P}(b, b_2)$ does not.

We will now prove that each reconstructed path contains labels for all vertices composing the actual path. Assume the contrary: there is a path whose reconstruction does not contain a label for an internal vertex x. Define the property A by

$$A(b, b') = \text{“}\exists a(x) \in \mathcal{R}(b, b')\text{”}. \tag{2.13}$$

We have already proved that a label for x will be placed in *some* path; together with our assumption it means that A is non-constant and we can apply Lemma 2.3.1.

If there is a vertex $b \in S_x^1 \cap S_x^2$ then we can find $b_1', b_2' \in V_B$ such that $\mathcal{P}(b, b_1')$ and $\mathcal{P}(b, b_2')$ both pass through x but $\mathcal{R}(b, b_1')$ has a label corresponding to x and $\mathcal{R}(b, b_2')$ does not. By the tree consistency property, the two paths coincide from b to at least x, therefore PCD$(b \prec b_1', b_2') \geq \delta$, where δ is the distance from b to x along the path $\mathcal{P}(b, b_1')$. However, the label $a(x)$ was placed into $\mathcal{R}(b, b_1')$ by a call to UPDATEPATH with this δ. This would trigger the condition on line 18 with $w = b_2'$ and the same label would be placed into $\mathcal{R}(b, b_2')$, a contradiction. The case when $R_x^1 \cap R_x^2$ is non-empty is treated similarly but this time condition on line 19 ensures the transfer of the label.

The last step of the proof of reconstruction is to check that only one label is created for any vertex. Assume the contrary, there is a vertex x which has at least two different labels corresponding to it appearing in different reconstructed paths. Fix one of the labels $a(x)$ and define property A by

$$A(b, b') = \text{“}a(x) \in \mathcal{R}(b, b')\text{”}. \tag{2.14}$$

This definition of A is very similar to eq. (2.13), but here $a(x)$ is some fixed label, whereas in (2.13) it was *any* label of x. By Lemma 2.3.1 and without loss of generality, there is a vertex $b \in S_x^1 \cap S_x^2$. This means that there is a path $\mathcal{P}(b, b_1)$ with the label $a(x)$ and a path $\mathcal{P}(b, b_2)$ with a label $a'(x)$ different from $a(x)$ (we remark that we have already shown that each reconstructed path contains labels for all its vertices). As before, the two paths coincide from b to at least x, therefore PCD$(b \prec b_1', b_2') \geq \delta$, where δ is the distance from b to x along the path $\mathcal{P}(b, b_1')$. During the execution of the algorithm, one of the labels was placed in its respective path first. But then the condition on line 18 would be triggered and the same label would be copied to the other path, before the other label is created. We have thus arrived to a contradiction. $\square$

Proof of Theorem 2.2.1: reconstruction of a non-compliant graph. We will now consider the output of the algorithm in case the PCD was created by the network graph $(G, V_B, \mathcal{P})$ that violates some of the conditions of the theorem. We will show that there is a “compliant” network graph that has the same PCD and which will therefore serve as the output of the algorithm. We start from the network graph $\mathcal{G} = (G, V_B, \mathcal{P})$ and apply the following “cleaning” operations to them (the order is important): (1) remove all unused edges, (2) split each separable vertex into 2 or more non-separable vertices[†], (3) remove each trivial vertex by merging its incident edges into one edge. The only adjustments needed to the paths $\mathcal{P}$ is to choose the correct copies of the split vertices. The set of boundary vertices remains the same.

An example of performing these operations is shown in Fig. 2.15. In particular, the internal vertex

[†]See Remark 2.3.2 below for a detailed discussion of why such a splitting exists.

x fails the non-separability condition, which is seen by defining the disjoint partitions

$$S = S^1 \cup S^2 = \{b_1, b_3\} \cup \{b_2\}$$
$$R = R^1 \cup R^2 = \{b_2\} \cup \{b_1, b_3\}$$

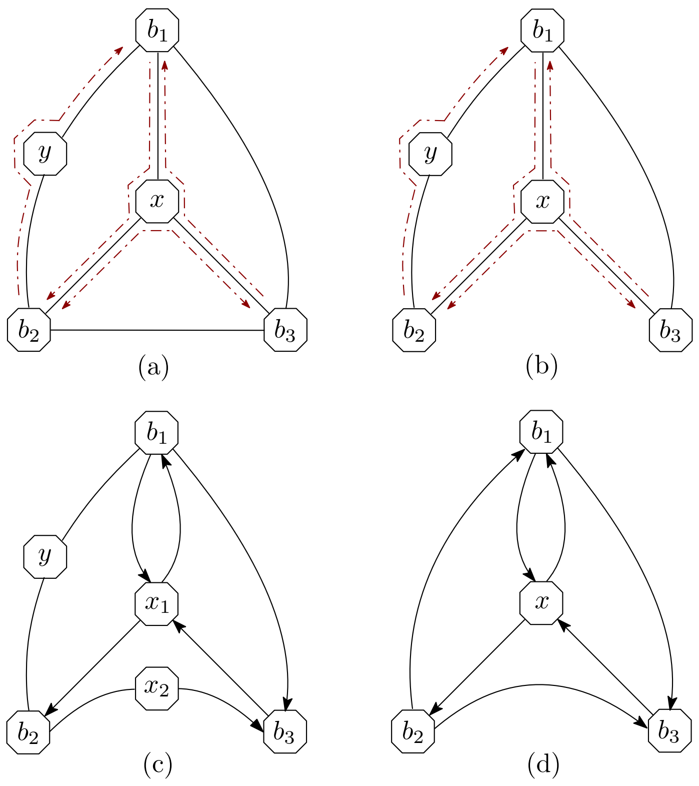


Figure 2.15: Cleaning of a graph: (a) the original graph (b) removing unused edge $e = (b_2, b_3)$ (c) splitting the vertex x into x_1 and x_2, (d) removing trivial vertices y and x_2. The edges that have no direction marked on them run in both directions.

We remark that splitting a separable vertex may require duplicating some incoming or outgoing edges, see Fig. 2.12, in which case the weights get duplicated too. The edges will be duplicated only if both resulting edges are present in some paths; thus no unused edges will be created.

Suppose an internal vertex got split into x_1 and x_2. It follows from the definition of the separable vertex that if $x \in \mathcal{P}(b, b_1)$ and $x \in \mathcal{P}(b, b_2)$ before the split, then after the split either both paths contain x_1 or both of them contain x_2. Therefore the length of each intersection of the form $\mathcal{P}(b, b_1) \cap \mathcal{P}(b, b_2)$ remains unchanged after a split. The same applies to any pair of paths $x \in \mathcal{P}(b_1, b)$ and $x \in \mathcal{P}(b_2, b)$. We conclude that the PCD remains unchanged by the operations above.

We denote the network graph so obtained by $\mathcal{G}^C$ (for "compliant" or "cleaned"). It is easy to see that $\mathcal{G}^C$ satisfies the conditions of the theorem and will therefore be reconstructed from its PCD by the algorithm. However, the PCD of $\mathcal{G}^C$ is the same as the PCD of $\mathcal{G}$. Therefore, given the PCD of $\mathcal{G}$, the algorithm will return the network graph $\mathcal{G}^C$. This logic is illustrated by Fig. 2.16.

$$\begin{tikzcd}
\boxed{\mathcal{G}} \arrow[r, "\text{measurement on network}"] \arrow[d, "C"'] & \boxed{\text{PCD}} \arrow[d, phantom, "\parallel"] \\
\boxed{\mathcal{G}^C} \arrow[r, "\text{measurement on network}"] & \boxed{\text{PCD}}
\end{tikzcd}$$

Figure 2.16: Flow of the proof of the reconstruction of a non-compliant graph: C is the operations that produce the corresponding compliant network graph $\mathcal{G}^C$, the Path Correlation Data of both graphs is the same.

.

$\square$

Remark 2.3.2. In the proof above we implicitly assumed that there exists a unique maximal splitting of a vertex into non-separable parts. We outline the proof of this fact. Fix x and define equivalence relations $\overset{S}{\sim}$ and $\overset{R}{\sim}$ on S_x and R_x correspondingly by letting

$$b_1 \overset{S}{\sim} b_2 \qquad \text{if} \qquad \exists\, \widehat{b} :\ x \in \mathcal{P}(b_1, \widehat{b}) \text{ and } x \in \mathcal{P}(b_2, \widehat{b}), \tag{2.15}$$

$$\widehat{b}_1 \overset{R}{\sim} \widehat{b}_2 \qquad \text{if} \qquad \exists\, b :\ x \in \mathcal{P}(b, \widehat{b}_1) \text{ and } x \in \mathcal{P}(b, \widehat{b}_2), \tag{2.16}$$

and closing each one by transitivity. It is easy to see that they are "dual" in the following sense: if $x \in \mathcal{P}(b_1, \widehat{b}_1)$ and $x \in \mathcal{P}(b_2, \widehat{b}_2)$ then

$$b_1 \overset{S}{\sim} b_2 \quad \Leftrightarrow \quad \widehat{b}_1 \overset{R}{\sim} \widehat{b}_2. \tag{2.17}$$

As a consequence, there is a natural one-to-one correspondence between the equivalence classes of $\overset{S}{\sim}$ and the equivalence classes of $\overset{R}{\sim}$. These equivalence classes provide the finest separation of the vertex x. To be more precise, the disjoint partitions

$$S_x = [b_1]_S \cup [b_2]_S \cup \ldots \cup [b_m]_S, \tag{2.18}$$

$$R_x = [\widehat{b}_1]_R \cup [\widehat{b}_2]_R \cup \ldots \cup [\widehat{b}_m]_R, \tag{2.19}$$

satisfy condition (2.8). Here $[\cdot]_S$ and $[\cdot]_R$ denote equivalence classes with respect to $\overset{S}{\sim}$ and $\overset{R}{\sim}$ and $x \in \mathcal{P}(b_j, \widehat{b}_j)$ for every j. Furthermore, if the nonempty partitions $S_x = S_x^1 \cup S_x^2$ and $R_x = R_x^1 \cup R_x^2$ satisfy Definition 6, then for any $b \in S_x$, either $[b]_S \subset S_x^1$ or $[b]_S \subset S_x^2$, and similarly for R.

2.4 Symmetric Paths

Reconstruction of network graph with the prior knowledge that the routing is symmetric (the edge weights may not be symmetric) has the advantage of being able to reconstruct a wider class of network graphs. In the following we present the proof of unique reconstruction of network graph with symmetric routing if the conditions of the main theorem are satisfied. The only differences on the required conditions for exact recovery are contained in Definition 8 for the non-triviality and non-separability of internal vertex. We start with an appropriate modification of Lemma 2.3.1.

Lemma 2.4.1. Let x be an arbitrary *non-separable* internal vertex and let B be a non-constant *symmetric* Boolean property (predicate) that is defined on the pairs (b, b') such that $x \in \mathcal{P}(b, b')$ and $x \in \mathcal{P}(b', b)$. Define S_x^1 to be the set of the sources of the paths through x for which B is true and S_x^2 to be the set of the sources of the paths for which B is false. More formally,

$$\begin{aligned} S_x^1 &= \Big\{ b \in S_x : \exists b_1 \in V_B \Big[x \in \mathcal{P}(b, b_1) \,\wedge\, B\,(b, b_1) \Big] \Big\}\,, \\ S_x^2 &= \Big\{ b \in S_x : \exists b_2 \in V_B \Big[x \in \mathcal{P}(b, b_2) \,\wedge\, \neg B\,(b, b_2) \Big] \Big\} \end{aligned} \tag{2.20}$$

Then $S_x^1 \cap S_x^2$ cannot be empty.

Proof of Lemma 2.4.1. For symmetric routing the receiver R_x and the source set $S_x = S_x^1 \cup S_x^2$ coincide. Assume that $S_x^1 \cap S_x^2 = \emptyset$. Then using the non-separability of vertex x, there is a path from $b_1 \in S_x^1$ to $b_2 \in S_x^2$. But this path either has property B or it does not.

(i) In the former case, we use symmetry of $B(b_1, b_2)$ to conclude $b_2 \in S_x^1$.

(ii) In the latter case $B(b_1, b_2)$ is false, thereby $b_1 \in S_x^2$,

either way, contradicting the assumption of disjointedness. □

Now we are in the position to apply Lemma 2.4.1 to prove the unique reconstruction of network graph with symmetric routing. While the proof steps are very similar to the ones discussed in proving Theorem 2.2.1, the main difference here is to ensure that the Boolean function B is symmetric.

2.4.1 Proof of the Reconstruction: Symmetric Paths

We first show that every vertex x has at least one label created for it. For an internal vertex x, fix an arbitrary path $\mathcal{P}_0$ through x and denote the vertices that the path visits before and after getting to x by x_1 and x_2. Denote by

$$\Lambda_x := \big\{ (x_1, x), (x, x_1), (x, x_2), (x_2, x) \big\} \subseteq E$$

the subset of edges contains x and either x_1 or x_2 as the other end vertex. Define the property $B = B(b, b')$ as the statement

$$B\,(b, b') = \text{“}\mathcal{P}(b, b') \text{ satisfies } |\mathcal{P}(b, b') \cap \Lambda_x| = 2\text{”}, \tag{2.21}$$

in other words, B is true if the path can be written as a sequence

$$\mathcal{P}(b, b') = [b, \ldots, x_1, x, x_2, \ldots, b'] \quad \text{or} \quad \mathcal{P}(b, b') = [b, \ldots, x_2, x, x_1, \ldots, b'].$$

Due to the symmetric routing assumption on the network graph, function B will be true for the reverse path $\mathcal{P}(b', b)$ as well. Because of our choice of x_1 and x_2, there are at least two paths on which B is true, namely $\mathcal{P}_0$ and its reversal. Since vertex x is non-trivial, there exists vertex x_3 adjacent to x and a path $\mathcal{P}'$ passing through (x, x_3) or (x_3, x) on which B is false. Therefore we can apply Lemma 2.4.1 and conclude that S_x^1 and S_x^2 are not disjoint.

Consider a boundary vertex $b \in S_x^1 \cap S_x^2$, and let $\mathcal{P}_1(b, b_1)$ and $\mathcal{P}_2(b, b_2)$ be the two paths where B is true and false respectively. By the tree consistency property, the two paths follow the same set of edges before diverging exactly at x because B is false on $\mathcal{P}_2$. As a result x is the $(b \prec b_1, b_2)$-junction and a label will be created for it in the main loop of Algorithm 1. This label will be placed in both reconstructed paths $\mathcal{R}(b, b_1)$ and $\mathcal{R}(b_1, b)$ unless there is already another label corresponding to x there.

Next we show that the reconstruction paths are not missing any vertices. Fix an internal vertex x and define the property

$$B\,(b, b') = \text{“}\exists\ a(x) \text{ such that } a(x) \in \mathcal{R}(b, b') \text{ and } a(x) \in \mathcal{R}(b', b)\text{”}, \tag{2.22}$$

which is clearly symmetric by its definition. We emphasis that $\mathcal{R}(b, b')$ and $\mathcal{R}(b', b)$ are required to contain the same label for x. We already proved that a label will be created for x, and when the first label is created, it will be placed into a path and its reversal. Therefore B is true on some paths. We will prove that this B holds for all paths (b, b') containing x. This will establish not only that the reconstructed paths are not missing any vertices, but also that the labels in $\mathcal{R}(b', b)$ are the same as in $\mathcal{R}(b, b')$.

Assume the contrary: B fails for some path. We apply Lemma 2.4.1 and let $b \in S_x^1 \cap S_x^2$. Then there exist $b_1, b_2 \in R_x$ so that a representation $\widehat{a}(x)$ exists in both $\mathcal{R}(b, b_1)$ and $\mathcal{R}(b_1, b)$ while B is false on (b, b_2). The latter implies that $\widehat{a}(x)$ cannot be present in both reconstructed paths $\mathcal{R}(b, b_2)$ or $\mathcal{R}(b_2, b)$. This can be summarized as follows: $\exists\, \widehat{a}(x)$ such that

$$\widehat{a}(x) \in \mathcal{R}(b, b_1) \;\text{ and }\; \widehat{a}(x) \in \mathcal{R}(b_1, b) \;\text{ and }\; \Big(\widehat{a}(x) \notin \mathcal{R}(b, b_2) \;\text{ or }\; \widehat{a}(x) \notin \mathcal{R}(b_2, b)\Big). \tag{2.23}$$

Now without loss of generality, assume that $\widehat{a}(x) \notin \mathcal{R}(b_2, b)$. The same arguments as in the proof of Theorem 2.2.1 guarantee that PCD$(b_1, b_2 \succ b) \geq \delta'$, where δ' is the distance from x to b. Therefore the insertion of the representation $\widehat{a}(x)$ into $\mathcal{R}(b_2, b)$ will be attempted by triggering the condition in line 19 for $z = b_2$ within the call of the function UPDATEPATH$(\mathcal{R}(b_1, b), \widehat{a}, \cdot)$. Since $\widehat{a}(x) \notin \mathcal{R}(b_2, b)$, we conclude that there is another label $a'(x) \in \mathcal{R}(b_2, b)$, placed there earlier by a call to UPDATEPATH$(\mathcal{R}(b_2, b), a', \cdot)$. But then the condition in line 19 will be triggered with $z = b_1$ and the label a' will be placed into $\mathcal{R}(b_2, b)$, in contradiction to our assumptions.

The last step of the proof is to show that no more than one label is created for each vertex. For a fixed label $a(x)$ of vertex x, define the property

$$B\,(b, b') = \text{“}a(x) \in \mathcal{R}(b, b')\text{”}. \tag{2.24}$$

which is symmetric since we already proved that property (2.22) holds for all paths containing x. By Lemma 2.4.1 there exists $b \in S_x^1 \cap S_x^2$ where the representation $a(x) \in \mathcal{R}(b, b_1)$ and $a'(x) \in \mathcal{R}(b, b_2)$ are distinct. But the two paths pass same set of edges at least up to vertex x, i.e. PCD$(b \prec b_1, b_2) \geq \delta$, where δ is the distance from b to x along the path $\mathcal{P}(b, b_1)$. The execution of algorithm places one of the labels in its respective path first. As a result, before any new label for x is created, condition on line 18 will have been triggered and copied the label to other paths. This is contradiction to the assumption of existence of two different labels of x in the set of reconstructed paths. This finish the proof.

Finally, for graphs with symmetric routing where the exact reconstruction conditions are not met, similar discussion as in Theorem 2.2.1 proves the uniqueness of the reconstruction result. The main point here is that cleaning operations preserve the symmetric routing on the graph. Moreover the compliant graph $\mathcal{G}^C$ satisfies the exact reconstruction conditions with same PCD as the one for original graph $\mathcal{G}$.

2.4.2 Specialized Algorithm for Symmetric Routing

We have shown that the reconstruction Algorithm 1 is universal: it covers both the general non-symmetric network and also graphs with symmetric routing. However, having the prior information that the routing on the network is symmetric makes it possible to call the recursive function UPDATEPATH less. This is due to the symmetric routing property that if an internal vertex x is inserted in reconstruction path $\mathcal{P}(u, v)$ for $u, v \in V_B$ then this vertex should also be inserted in the reverse path $\mathcal{P}(v, u)$ with appropriate distance from root v (this is not generally true for non-symmetric routing case). To take advantage of this feature we propose Algorithm 2 as a specialized version of the reconstruction algorithm for the graphs with symmetric routing.

Compared to the Algorithm 1, the new algorithm calls the recursive function twice less (as can be seen by comparing the main loops of the two algorithms). The change in the number of calls of reconstruction function is compensated by adding line 14 in the new algorithm where the label of internal vertex which is inserted in the given path will be inserted in the reverse one as well. It should be pointed out that although less calls of main reconstruction function makes the algorithm computationally more effective, from the point of view of mathematical complexity (in terms of insertions of a vertex into a reconstructed path), the two algorithms can be considered the same.

Algorithm 2 Reconstruction of a network graph with symmetric routing

```
 1: for $b_1, b_2 \in V_B$ do                                          ▷ Initialization
 2:     $\mathcal{R}(b_1, b_2) = [(b_1, 0), (b_2, |\mathcal{P}(b_1, b_2)|)]$
 3: end for
 4: for $b_1, b_2, b_3 \in V_B$ do                                     ▷ Main Loop
 5:     create label $a$
 6:     $\delta = \mathrm{PCD}(b_1 \prec b_2, b_3)$
 7:     $\delta' = \mathrm{PCD}(b_2, b_3 \succ b_1)$
 8:     UPDATEPATH($\mathcal{R}(b_1, b_2), a, \delta, \delta'$)
 9: end for
10: Read off the graph from reconstructed paths $\mathcal{R}$.          ▷ Return the result

11: function UPDATEPATH($\mathcal{R}(u, v), a, \delta, \delta'$)          ▷ Recursive Function
12:     if $\exists\, (\cdot, \delta) \in \mathcal{R}(u, v)$ then return
13:     insert $(a, \delta)$ into $\mathcal{R}(u, v)$
14:     insert $(a, |\mathcal{P}(v, u)| - \delta')$ into $\mathcal{R}(v, u)$
15:     for $z \in V_B$ do
16:         if $\mathrm{PCD}(u \prec v, z) \geq \delta$ then UPDATEPATH($\mathcal{R}(u, z), a, \delta, \delta'$)
17:         if $\mathrm{PCD}(v \prec u, z) \geq \gamma'$ then UPDATEPATH($\mathcal{R}(v, z), a, |\mathcal{P}(v, u)| - \delta', |\mathcal{P}(u, v)| - \delta$)
18:     end for
19: end function
```

2.4.3 Reconstruction Example

In the following example, we will show how having the prior information of symmetric routing will help to uniquely reconstruct the network graph shown in Fig. 2.17(a). The selected routing among the boundary vertices $V_B = \{b_1, b_2, b_3\}$ follows the sequence $\mathcal{P}(b_i, b_j) = [b_i, x_i, x_j, b_j]$ for $i, j = 1, 2, 3$. From the measurement point of view, the PCD on the graph is represented as the set of observed logical trees in Fig. 2.18.

If there is no information on the symmetry of routing, then we can not conclude that $a_1 = a_4$, $a_2 = a_5$ and $a_3 = a_6$. The resulting reconstruction will be the graph appearing in Fig. 2.17(b).

We remark that such geometry is fairly realistic if we, for example, consider the vertices b_j to be internet service providers (ISPs) who are eager to push the traffic addressed outside their network to other ISPs as soon as possible. Still, the reconstruction does not match the graph we had originally.

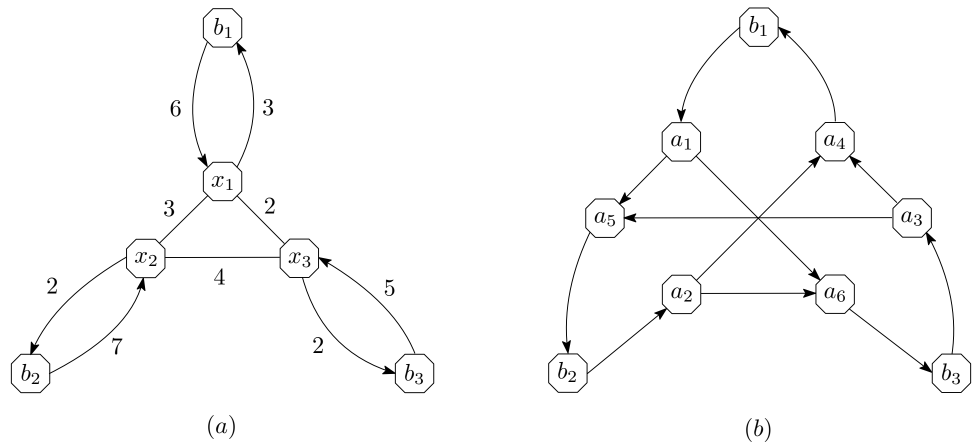


Figure 2.17: (a) An example of a network graph. To avoid clutter, the edges with no specified direction are assumed to go in both directions with the same weight, (b) reconstructed network

However, the reconstruction *is possible* if we know a priori that the routes are symmetric, as they are in this example. Then the paths going out of a given source b and the paths going to b acting as the receiver have exactly the same topology. This additional information allows us to identify $a_1 = a_4$, $a_2 = a_5$ and $a_3 = a_6$ in Fig. 2.17(b) and thus recover the original graph. Since symmetric routing networks appear in applications, we expect the optimized Algorithm 2 to be a practical value.

2.4.4 The Effect of Symmetric Edge Weights

Symmetric network graph for which both the routing on the graph and edge weights are symmetric can be considered as a special class of general networks. For this class of networks, similar to the graphs with symmetric routing (see eq. 2.7), same set of edges are passed in the two reverse paths $\mathcal{P}_1(u, v)$ and $\mathcal{P}_2(v, u)$. Additionally, the symmetric edge weights property implies that for any vertex $x \in \mathcal{P}_1$, then $|\mathcal{P}(v, x)| = |\mathcal{P}(u, v)| - |\mathcal{P}(u, x)|$. In other words, path $\mathcal{P}_2$ is fully identified by the available information form the path $\mathcal{P}_1$. This property of symmetric networks implies that PCD$(v, w \succ u) =$ PCD$(u \prec v, w)$, and as a result the information from PCD encoded in the form of $(u \prec v, w)$-junctions can be applied to identify the information for the $(v, w \succ u)$-junctions. From the **reconstruction** point of view, same conditions are required for exact reconstruction of graphs with symmetric edge weights as discussed in previous Sections and thereby Algorithm 2

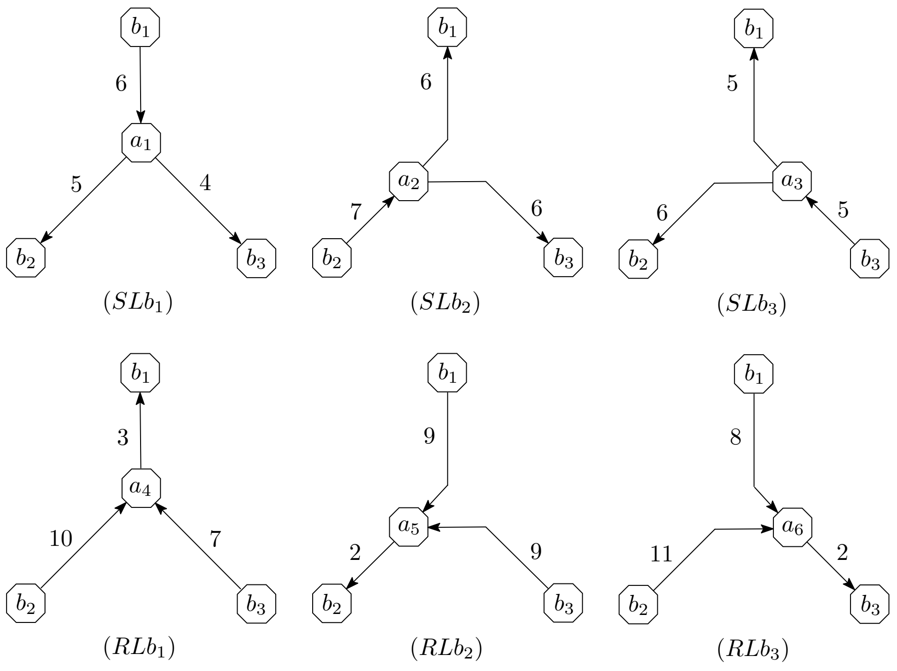


Figure 2.18: Representation of PCD on the network example through set of logical source and receiver trees.

can be applied for reconstruction purposes.

3. CONSISTENT MERGING AND PRUNING OF SUBGRAPHS

Practical network measurement may provision data with imperfect consistency. For example, input data may be provided in the form of weighted trees computed from packet measurement over time intervals that are not perfectly aligned, so that the metric of a path $\mathcal{P}(b, b')$ may be reported differently in the source tree from b and the receiver tree to b'. Even with aligned intervals, deviations from the model and statistical fluctuations due to finitely many probe packets may result in inconsistency. To enable our proposed algorithm to operate with such data, we propose to compute a PCD that is a least-squares fit to inconsistent tree data.

3.1 The Challenge of Subgraph Weight Inconsistency

A key challenge for fusion derives from the *weight inconsistencies* between different inferred subgraphs of the network. We say that two partial network graph $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$ have inconsistent weights if there is a pair of vertices $u, v \in V_B^{(1)} \cap V_B^{(2)}$ such that $P_{u,v} \in \mathcal{P}^{(1)}$ and $P_{u,v} \in \mathcal{P}^{(2)}$, yet the corresponding path weights are unequal: $W_{u,v}^{(1)} \neq W_{u,v}^{(2)}$. Causes of inconsistencies include:

(C$_1$) **Statistical variation:** when inference of two partial network graphs yields different weight estimates for edges in their intersection.

(C$_2$) **Imperfect temporal alignment:** some inference methods use time series of path performance metrics over a sequence of time slots. However, the slots for different time series may not be synchronized among different hosts due to variable transit time or clock skew. Although packet sequence numbers can be used to coordinate slot alignment at different hosts, these may not be available for measurements of background traffic.

(C$_3$) **Deviations from a model:** violation of assumptions, such as edge independence, cause inconsistent apportioning of path performance amongst the path's constituent edges.

Two main consequences of weight inconsistency are as follows:

3.1.1 Difficulties in Merging Weighted Graphs

Lack of consistent path weights prevents application of fusing algorithm stated in Section 2. To illustrate this, consider in Figure 3.1 two weighted simple binary trees $\mathcal{T}_{b_1}^S$ and $\mathcal{T}_{b_2}^R$ that have been inferred by primitive inference using distinct packet time series involving the same three boundary vertices b_1, b_2 and b_3 with central vertices c and c' respectively. $\mathcal{T}_{b_1}^S$ is a source tree rooted at vertex b_1 while $\mathcal{T}_{b_2}^R$ is a receiver tree rooted at vertex b_2. The trees have the common directed path $P_{1,2}$ which we wish to merge. However, unless the respective paths weights $w_1 + w_2$ and $v_1 + v_2$ in $\mathcal{T}_{b_1}^S$ and $\mathcal{T}_{b_2}^R$ are equal, it is not immediately clear how to assign interior vertices and edge weights in a merged graph. A naive approach is to assign to the path some combination of the available weights, such as the arithmetic mean. However, changing the paths weights necessitates changing the weights in the constituents edges, which may impact the consistency of other paths that contain those edges, and so on. Instead, what is needed is a principled approach to removing all path inconsistencies by coordinated adjustment of their constituent edge weights.

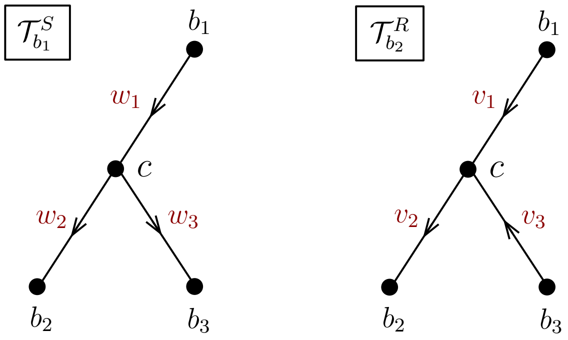


Figure 3.1: Source tree $\mathcal{T}_{b_1}^S$ rooted at vertex b_1, receiver tree $\mathcal{T}_{b_2}^R$ rooted at vertex b_2 both having version of direct path P_{b_1,b_2}.

3.1.2 Topological Noise and Edge Pruning

Now suppose the weights on path $P_{1,2}$ *are* consistent, i.e., $w_1 + w_2 = v_1 + v_2$. Then the trees $\mathcal{T}_{b_1}^S$ and $\mathcal{T}_{b_2}^R$ may be merged, forming the graph $\mathcal{G}$ in Figure 3.2. Without loss of generality we have assumed $v_1 < w_1$ and hence $v_2 > w_2$. To merge, the central vertices c from $\mathcal{T}^S_{b_1}$ and c' from $\mathcal{T}^R_{b_2}$ are inserted in the path joining b_1 and b_2 according to the edge weights in their respective topologies; see e.g. [23]. This results in a directed edge (c, c') of weight $\delta = w_1 - v_1 = v_2 - w_2$. The distinction between vertices c and c' may represent asymmetric routing in the underlying network. However, estimation errors of the type (C_1) will be manifest in the form of extraneous small weight edges. In order to simplify the inferred topology, such edges are *pruned* i.e. removed from the topology and their endpoints identified. Criteria for pruning include (a) an edge weight being less that a threshold performance metric values of interest, and (b) an edge weight being statistically indistinguishable from zero, e.g., on account of being less than some multiple of its estimated standard deviation. Pruning the edge (c, c') from topology $\mathcal{G}$ gives rise to the pruned topology $\widetilde{\mathcal{G}}$ in Figure 3.2. However, pruning introduces new inconsistencies since the weight $v_1 + w_2$ on path $P_{1,2}$ is less than the measured weight $w_1 + w_2 = v_1 + v_2$. Naive approaches to restoring consistency, such as allocating a total weight $w_1 + w_2$ in proportion to the weights v_1 on edge (b_1, c) and w_2 in edge (c, b_2) in $\widetilde{\mathcal{G}}$ will cause weight inconsistencies with other paths containing those edges. Again, we seek a principled way of redistributing the weights of pruned edges while maintaining measured path weights.

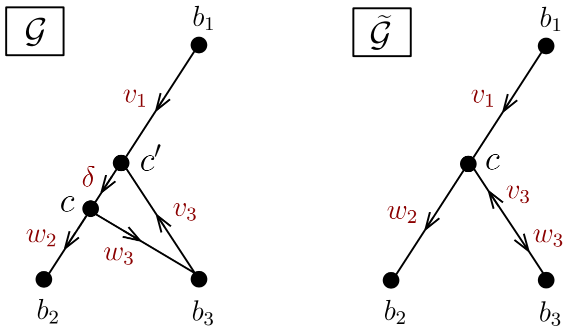


Figure 3.2: Left: merged graph $\mathcal{G}$ under equal path weights $w_1 + w_2 = v_1 + v_2$ for directed path $P_{1,2}$. With no loss of generality assume $v_1 < w_1$ and hence $v_2 > w_2$. Right: Pruned graph $\widetilde{\mathcal{G}}$ after removal of edge (c, c') from $\mathcal{G}$.

3.2 Problem Statement

Motivated by the problems inherent in graph merging and graph pruning described above, we abstract two variant problems in treating inconsistency, as follows:

3.2.1 Extrinsic Consistency

Let $\mathcal{G} = (G, V_B, \mathcal{P})$ be a partial network graph. A *target path weight set* $\mathcal{Z}$ is positive function on a path subset $\widetilde{\mathcal{P}} \subseteq \mathcal{P}$. We call edge weights $\widetilde{\mathcal{W}}$ on E *extrinsically consistent* with $\mathcal{Z}$ if $Z_{u,v} = \widetilde{W}_{u,v}$ whenever $(u, v) \in \widetilde{\mathcal{P}}$. Given original edge weights $\mathcal{W}$ and target path weights $\mathcal{Z}$, our problem is to find edge weights $\widetilde{\mathcal{W}}$ that are (a) extrinsically consistent with the target path weights $\mathcal{Z}$; (b) close to $\mathcal{W}$ in a sense to be defined, and (c) readily computable.

3.2.2 Intrinsic Consistency

Let $\{\mathcal{G}^{(1)}, \dots \mathcal{G}^{(k)}\}$ be a set of inferred partial network subgraphs with boundary sets $V_B^{(i)} \subset V_B = \bigcup_i V_B^{(i)}$, and path sets $\mathcal{P}^{(i)}$ joining ordered pairs of vertices in $V_B^{(i)}$. Apart from the boundary points, the graph elements are distinct with no identification between internal vertices $V^{(i)} \setminus V_B^{(i)}$ and edges $E^{(i)}$ from different subgraphs $\mathcal{G}^{(i)}$. We call a set of weights $\{\widetilde{\mathcal{W}}^{(1)}, \dots, \widetilde{\mathcal{W}}^{(k)}\}$ on the graphs $\{\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(k)}\}$ *intrinsically consistent* if for any ordered pair $(u, v) \in V_B$ and all i for which there is a path $\mathcal{P}_{u,v}^{(i)} \in \mathcal{P}^{(i)}$ connecting u and v, the path weight $\widetilde{\mathcal{W}}_{u,v}^{(i)} = \sum_{e \in \mathcal{P}_{u,v}^{(i)}} \widetilde{w}_e^{(i)}$ is independent of i. Given the original set of weights $\{\mathcal{W}^{(i)}\}$ on $\{\mathcal{G}^{(1)}, \dots, \mathcal{G}^{(k)}\}$, our problem is to find a set of weights $\{\widetilde{\mathcal{W}}^{(i)}\}$ that are (a) intrinsically consistent; (b) close to $\{\mathcal{W}^{(i)}\}$ in a sense to be defined, and (c) readily computable.

3.2.3 Contribution and Outline of this Section

The contributions of this Section are summarized as follows:

1. We formulate and solve the problems of producing extrinsically or intrinsically consistent weights in a least-squares framework. We seek to minimize the square differences $||\widetilde{\mathcal{W}} - \mathcal{W}||^2$ of the solution and given edge weights, under consistency and positivity conditions for $\widetilde{\mathcal{W}}$. In each case, the solutions are expressed in term of the Moore-Penrose pseudo-inverse of

a generalized routing matrix that expresses the linear constraints between path and edge weights. While on the one hand this is a standard approach to constrained optimization, we must make a careful examination of invertibility properties in order to provide a computable solution and avoid singularities. While several approaches exist in the literature to ensure positivity conditions, we are able to take a simpler approach informed by the network context that small or negative weights are uninteresting and may be set to zero and thereafter ignored.

2. We show how our solution for extrinsic consistency applies to the problem of pruning topological noise.

3. We show how our solution for intrinsic consistency applies to the problem of preparing inconsistent inferred trees for fusion in networks with asymmetric routing.

3.3 Extrinsic Consistency

Let $\mathcal{G} = (G, V_B, \mathcal{P})$ be a partial network graph and $\mathcal{Z} : \mathcal{P} \to \mathbb{R}_{\geq 0}$ be a target path weight set. If $\mathcal{Z}$ is specified on only a subset of $\mathcal{P}$, we can extend it to all of $\mathcal{P}$ by assigning $Z_{u,v} = W_{u,v}$ for all other pairs $(u, v) \in \mathcal{P}$. We seek a set of weights $\widetilde{\mathcal{W}}$ on E which is extrinsically consistent with $\mathcal{Z}$ as a solution to the constrained optimization problem

$$\min_{\widetilde{\mathcal{W}}} \|\mathcal{W} - \widetilde{\mathcal{W}}\|^2 \quad \text{such that} \tag{3.1}$$

$$\sum_{e \in P_{u,v}} \widetilde{w}_e = Z_{u,v}, \ \forall (u,v) \in \mathcal{P} \tag{3.2}$$

$$\widetilde{w}_e \geq 0, \ \forall e \in E \tag{3.3}$$

The positivity constraint (3.3) originates in the interpretation that positive edge weights w_e are associated with performance impairment, while $w_e = 0$ indicates no impairment on edge e. We focus first on the optimization problem (3.1)-(3.2) returning to the positivity constraint in Section 3.5.

Theorem 3.3.1. Rewrite equation (3.2) as

$$A\widetilde{\mathcal{W}} = \mathcal{Z}, \tag{3.4}$$

where $A = \left(A_{(u,v),e}\right)_{u,v\in\mathcal{P},e\in E}$ denotes the incidence matrix of edges in paths,

$$A_{(u,v),e} = \begin{cases} 1, & \text{if } e \in P_{u,v}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.5}$$

Then the least squares solution of (3.2) which minimizes (3.1) is given by

$$\widetilde{\mathcal{W}} = \mathcal{W} + A'(\mathcal{Z} - A\mathcal{W}), \tag{3.6}$$

where A' is the Moore-Penrose pseudo-inverse of A.

Proof of Theorem 3.3.1. Introducing new notation

$$\delta = \widetilde{\mathcal{W}} - \mathcal{W} \quad \text{and} \quad \eta = \mathcal{Z} - A\mathcal{W},$$

in (3.4) and (3.1), we seek solutions to the linear equation $A\delta = \eta$ of minimal ℓ_2 norm $\|\delta\|^2$. It is well known [35], that the least squares solution of minimal norm is given by $\delta = A'\eta$ where A' is the Moore-Penrose pseudo-inverse of A. $\square$

Remark 3.3.2. We stress that there is no guarantee that (3.4) is consistent (possess at least one solution). If it is not, the least squares solution is the best fit with respect to ℓ_2 norm. More precisely, the solution $\widetilde{\mathcal{W}}$ in (3.6) is the vector which minimizes $\|\mathcal{Z} - A\widetilde{\mathcal{W}}\|^2$ and, if there is more than one minimizer, $\widetilde{\mathcal{W}}$ also minimizes $\|\mathcal{W} - \widetilde{\mathcal{W}}\|^2$.

On the practical level, the external constraint is often known only approximately. In this situation, it is natural to accept an approximate solution to the constraint.

Remark 3.3.3. For numerical computation of the Moore-Penrose pseudo-inverse, there exist simple and stable prescriptions. For example, if AA^T is invertible which is equivalent to $Aw = z$ being solvable for any z, then A' is given by the formula

$$A' = A^T(AA^T)^{-1}. \tag{3.7}$$

If AA^T is not invertible, formula (3.7) can still be used via regularization or by representing the real symmetric matrix AA^T as $0_K \oplus B$, where K is the null-space of AA^T and B is the restriction of AA^T to the orthogonal complement of $K^{\perp}$, and interpreting $(AA^T)^{-1}$ as $0_K \oplus B^{-1}$.

An important special case of Theorem 3.3.1 is when the underlying graph is a tree. In this case (3.4) will have one or more solutions for every $\mathcal{Z}$. More precisely, let G be a directed tree with root vertex b and leaf set $\mathcal{L}_b = V_B \setminus \{b\}$. The following result applies equally to a *source tree*, where there is a unique simple path $P_{b,u}$ for each $u \in \mathcal{L}_b$ and to a *receiver tree* consisting of unique simple paths $P_{u,b}$, with $u \in \mathcal{L}_b$.

Corollary 3.3.4. Let G be a directed tree and let A be the incidence matrix of edges over the set of simple paths to (or from) the root vertex ρ from (or to) the leaf vertices. Then A has the full row rank and the solution that minimizes (3.1) is given by (3.6) with A' computable using (3.7).

Proof of Corollary 3.3.4. It suffices to show that A has linearly independent rows. Indeed, this implies that the null-space of A^T is zero and, by the rank-nullity theorem, the range of A is the entire space ($A\widetilde{\mathcal{W}} = \mathcal{Z}$ has a solution for any $\mathcal{Z}$). Also, for any $\mathcal{W} \neq 0$, $0 < \|A^T\mathcal{W}\|^2 = \langle \mathcal{W}, AA^T\mathcal{W} \rangle$ and thus $AA^T\mathcal{W} \neq 0$ and the matrix AA^T is invertible. Since by our construction a path $P(b, b')$ does not pass through any other boundary vertices, every path is uniquely identified by the leaf it goes to (or from). Thereby every row of matrix A has entry 1 corresponding to the leaf edge. This entry must be 0 in any other row in A. Therefore the rows are linearly independent. $\square$

3.3.1 Edge Weight Adjustment After Pruning

To show how the general framework of Section 3.3 applies to pruning we address the question: how should the weight of pruned edges be assigned to remaining edges in order to preserve end-to-end path weights. Consider a partial network graph G and denote by A the incidence matrix of edges over paths, equation (3.5). Pruning an edge amounts to deleting the corresponding column from A and contracting the edge in the underlying graph. All paths in $\mathcal{P}$ remain connected upon identification of the endpoints of the deleted edge and no further adjustment of A is needed.

Denote by $\tilde{E} \subset E$ the reduced set of edges and by $\tilde{A}$ the corresponding incidence matrix of the pruned network. Given original edge weights $\mathcal{W} = \{x_i : i \in E\}$ we seek edge weights $\widetilde{\mathcal{W}} = \{\tilde{w}_i : i \in \tilde{E}\}$ of the pruned network that reproduce the same total weight on all paths in $\mathcal{P}$,

$$\tilde{A}\widetilde{\mathcal{W}} = A\mathcal{W}, \tag{3.8}$$

and minimize the square distance on the remaining edges,

$$\|\widetilde{\mathcal{W}} - \mathcal{W}_{\tilde{E}}\|_2^2 = \sum_{i \in \tilde{E}} |\tilde{w}_i - w_i|^2, \tag{3.9}$$

where $\mathcal{W}_{\tilde{E}}$ denotes the restriction of $\mathcal{W}$ to $\tilde{E}$.

Corollary 3.3.5. The least squares solution $\widetilde{\mathcal{W}}$ of (3.8) which minimizes (3.9) is given by

$$\widetilde{\mathcal{W}} = \mathcal{W}_{\tilde{E}} + \tilde{A}' \left(A\mathcal{W} - \tilde{A}\mathcal{W}_{\tilde{E}} \right) \tag{3.10}$$

where $\tilde{A}'$ is the Moore-Penrose pseudo-inverse of $\tilde{A}$. If the graph $\tilde{G}$ obtained after pruning is a tree, the matrix $\tilde{A}$ has full row rank.

Proof of Corollary 3.3.5. Use Theorem 3.3.1 with $\mathcal{Z} = A\mathcal{W}$. Full row rank of $\tilde{A}$ follows from Corollary 3.3.4. $\square$

Remark 3.3.6. The result in Corollary 3.3.5 does not guarantee that all elements of $\widetilde{W}$ are positive. In subsection 3.5, we discuss different approaches that can be applied to impose this sign constraint.

Note that if there is only limited amount of pruning involved, we can guarantee that system (3.8) does have at least one feasible solution even if the graph is not a tree. The following lemma sketch the sufficient conditions on the existence of solution to system (3.8).

Lemma 3.3.7. Suppose the graph G has no looping edges and a directed edge (u, v) has one of the following properties:

1. u is not a boundary vertex and the weight of (u, v) is strictly smaller than the weight of any other edges originating from u, or

2. v is not a boundary vertex and the weight of (u, v) is strictly smaller than the weight of any other edge terminating at v,

then pruning (u, v) results in a consistent system (3.8).

Proof of Lemma 3.3.7. Denote the weight of (u, v) by δ. Assume the first condition is satisfied (the proof of the second case is similar). We contract (u, v), and adjust weights by subtracting δ from the weight of every edge originating from u and adding δ to the weight of every edge terminating at u. It is easy to see that the weight of every path remains the same. Indeed, if the path P visits the vertex u, P must contain exactly one edge of the form (u', u) and exactly one edge of the form (u, v') (here we make use of the fact that there are no loops at vertex u). Then

$$\widetilde{W}_{u',u} = W_{u',u} + \delta, \quad \widetilde{W}_{u,v'} = W_{u,v'} - \delta, \quad \widetilde{W}_{u',u} + \widetilde{W}_{u,v'} = W_{u',u} + W_{u,v'}.$$

The above reasoning also applies to the case $v' = v$ since we can view the contracted edge (u, v) as an edge with $\widetilde{W}_{u.v} = 0$. We have thus constructed one solution to (3.8) and therefore it is consistent. $\square$

Since a reasonable pruning scenario is pruning edges of the smallest weight, one can attempt an iterative application of the above Lemma. However, there can be problems of topological nature. The following example clarifies the discussion above.

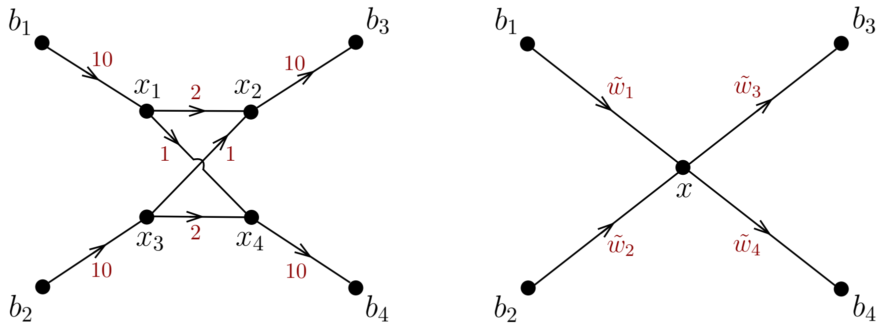


Figure 3.3: Inconsistency occurs by pruning the edges. Left: the original partial graph, right: the pruned representation.

Example 4. Consider the partial network graph with source b_1, b_2 and receivers vertices b_3, b_4 in Figure 3.3(left), and the result of pruning four shortest edges as shown in Figure 3.3(right).

The linear system in equation (3.8) consists of four equations,

$$\tilde{w}_1 + \tilde{w}_3 = 22, \qquad \tilde{w}_2 + \tilde{w}_3 = 21,$$

$$\tilde{w}_1 + \tilde{w}_4 = 21, \qquad \tilde{w}_2 + \tilde{w}_4 = 22.$$

Eliminating $\tilde{w}_1$ from the first column and $\tilde{w}_2$ from the second column shows the system is inconsistent ($\tilde{w}_3 - \tilde{w}_4 = \pm 1$). Applying the result of Corollary 3.3.5 will assign new weights $\tilde{w}_i = 10.75$ for $i = 1, \ldots, 4$. While the consistency from end-to-end point of view fails (e.g. $W_{b_1,b_4} \neq \widetilde{W}_{b_1,b_4}$), the weights $\widetilde{\mathcal{W}}$ are optimal in ℓ_2 sense.

3.4 Intrinsic Consistency in Tree-Based Network Inference

In this Section we address the question of intrinsic consistency (see Section 3.2) in the setting of our intended application, when the graphs $\mathcal{G}^{(i)}$ are trees. More precisely, we assume to be given the set V_B and, for any $b \in V_B$, two directed trees, a source tree $\mathcal{T}_b^S = (V_b^S, E_b^S, \mathcal{W}_b^S)$ and a receiver tree $\mathcal{T}_b^R = (V_b^R, E_b^R, \mathcal{W}_b^R)$. In each tree, the root is b and the leaf set is identified with $V_B \setminus \{b\} =: \mathcal{L}_b$. All edges are directed away from b in the source tree $\mathcal{T}_b^S$ and towards b in the receiver tree. In applications these trees have been inferred (with some subsequent pruning) from packet measurements on an underlying network, with the only known parameter of the network being the set V_B.

Note that $u \in \mathcal{L}_v$ iff $v \in \mathcal{L}_u$. Each edge i in E_v^S possesses an inferred weight $w_{v,i}^S \geq 0$ and likewise each edge i in E_v^R possesses a inferred weight $w_{v,i}^R \geq 0$. We emphasize that all trees are distinct with no edge or internal vertices in common. For each $u \neq v \in V_B$ let $P_{v,u}^S$ denote the unique path that connects v to u in $\mathcal{T}_v^S$. Similarly, let $P_{v,u}^R$ denote the unique path connecting v to u in the receiver tree $\mathcal{T}_u^R$ rooted at u. If the given trees are source and receiver trees generated from the same underlying network, the paths must be identical. In particular, their weights must be the same.

Let $E^* = \bigcup_{v \in V_B} E_v^R \cup E_v^S$ denote the set of all edges over all source and receiver trees and we write $\mathcal{W} \in \mathbb{R}^{E^*}$ for the vector of their weights, $\mathcal{W} = \{w_{v,i}^S, v \in V_B, i \in E_v^S\} \cup \{w_{v,i}^R, v \in V_B, i \in E_v^R\}$. We seek to determine the vector of weights $\widetilde{\mathcal{W}} = \{\tilde{w}_{v,i}^S : v \in V_B,\ i \in E_v^S\} \cup \{\tilde{w}_{v,i}^R : v \in V_B,\ i \in E_v^R\} \in \mathbb{R}^{E^*}$ that minimize the square difference:

$$\|\mathcal{W} - \widetilde{\mathcal{W}}\|_2^2 = \sum_{v \in V_B} \left(\sum_{i \in E_v^S} \left(w_{v,i}^S - \tilde{w}_{v,i}^S\right)^2 + \sum_{i \in E_v^R} \left(w_{v,i}^R - \tilde{w}_{v,i}^R\right)^2 \right) \tag{3.11}$$

subject to the common path consistency constraint,

$$\forall_{v \in V_B},\ \forall_{u \in L_v} : \sum_{i \in P_{v,u}^S} \tilde{w}_{v,i}^S = \sum_{i' \in P_{v,u}^R} \tilde{w}_{u,i'}^R. \tag{3.12}$$

Let Q denote the set of ordered pairs of distinct boundary vertices and let A denote the $|Q| \times |E^*|$

signed incidence matrix of edges over Q, defined by

$$A_{(v,u),i} = \begin{cases} +1 & \text{if } i \in P^R_{v,u}, \\ -1 & \text{if } i \in P^S_{v,u}. \end{cases}$$

With this notation $(A\mathcal{W})_{(v,u)}$ is the asymmetry between the total weight on the paths from v to u on the receiver tree with root u and the source tree with root v. The constraint (3.12) is written succinctly as $A\widetilde{\mathcal{W}} = 0$.

Theorem 3.4.1. The matrix A has full row rank (therefore AA^T is invertible) and the solution to the constrained optimization (3.11), (3.12) is given by

$$\widetilde{\mathcal{W}} = \mathcal{W} - A^T(AA^T)^{-1}A\mathcal{W}, \tag{3.13}$$

with the following error bound

$$\|\widetilde{\mathcal{W}} - \mathcal{W}\|_2^2 \le \|A\mathcal{W}\|_2^2/2. \tag{3.14}$$

This error bound is tight for tree network graphs.

Remark 3.4.2. This suggests a possible heuristic for pruning the tree with weights computed using Theorem 3.4.1: prune the maximal set of edges of smallest weight, whose sum of square weights does not exceed $\|A\mathcal{W}\|_2^2/2$.

Proof of Theorem 3.4.1. Writing $\widetilde{\mathcal{W}} = \mathcal{W} + \delta$, then we seek solutions δ to the linear equation $A\delta = -A\mathcal{W}$ of minimal ℓ_2 norm. The form (3.13) then follows if AA^T is invertible, as we now establish. Observe we can write $A = (A_R, -A_S)$ (joining the corresponding rows) where A_R and A_S are the incidence matrices of the edges over paths in receiver trees and (reversed) paths in source trees. Under an appropriate ordering of the pairs in Q the matrix A_R decomposes into sum $A_R = \oplus_{v \in V_b} A_{R,v}$ (similarly, $A_S = \oplus_{v \in V_b} A_{S,v}$ which we will need later). Since each $A_{R,v}$ has independent rows, therefore A_R and, by extension, A have independent rows. Now referring to Corollary 3.3.4, invertibility of AA^T follows.

To prove the error estimate, equation (3.14), we write $\delta = \widetilde{\mathcal{W}} - \mathcal{W} = -A^T(AA^T)^{-1}A\mathcal{W}$ and hence

$$\begin{aligned}
\|\delta\|_2^2 &= \langle A^T(AA^T)^{-1}A\mathcal{W}, A^T(AA^T)^{-1}A\mathcal{W}\rangle \\
&= \langle (AA^T)(AA^T)^{-1}A\mathcal{W}, (AA^T)^{-1}A\mathcal{W}\rangle = \langle A\mathcal{W}, (AA^T)^{-1}A\mathcal{W}\rangle \\
&\leq \|A\mathcal{W}\|_2^2 \, \|(AA^T)^{-1}\|
\end{aligned}$$

using the spectral norm for matrices. The result follows if we can show $\|(AA^T)^{-1}\| \leq 1/2$ or, equivalently, $\|AA^T\| \geq 2$. The latter follows from $AA^T = A_R A_R^T + A_S A_S^T$ if $\langle y, A_R A_R^T y\rangle \geq \|y\|^2$ and $\langle y, A_S A_S^T y\rangle \geq \|y\|^2$ for all vectors $y \in \mathbb{R}^Q$. It suffices to show this for receiver trees. Observe $\langle y, A_R A_R^T y\rangle = \sum_{v \in B} \langle y_v, A_{R,v} A_{R,v}^T y_v\rangle$ where y_v is projection of $y \in \mathbb{R}^Q$ onto $\mathbb{R}^{\mathcal{L}_v}$. We now represent

$$A_{R,v} A_{R,v}^T = \sum_{i \in E_v^R} A_{R,v,i} A_{R,v,i}^T,$$

where $A_{R,v,i}$ is the i-th column of the matrix $A_{R,v}$. If i corresponds to an edge ending in a leaf u, the matrix $A_{R,v,i} A_{R,v,i}^T$ has a single 1 on the diagonal, in the position $((v,u),(v,u))$. Therefore, the sum of $A_{R,v,i} A_{R,v,i}^T$ over leaf edges i produces an identity matrix, while the rest of the summands are clearly positive semidefinite. We conclude that $A_{R,v,i} A_{R,v,i}^T \geq \mathbb{I}$, therefore $\langle y_v, A_{R,v} A_{R,v}^T y_v\rangle \geq \|y_v\|^2$ and so $\langle y, A_R A_R^T y\rangle \geq \sum_{v \in V_B} \|y_v\|^2 = \|y\|^2$.

We now show that the bound is tight for tree network graphs. Observe that this class of networks has the property that the routing paths are symmetric and the receiver tree $\mathcal{T}_v^R$ and source tree $\mathcal{T}_v^S$ are isomorphic at each $v \in V_B$. Denote by $S := A_S A_S^T$ and $R := A_R A_R^T$, these two matrices act on vectors indexed by pair of distinct boundary vertices. Due to structure of matrix A, $S_{(u_1,v_1)(u_2,v_2)} = 0$ if $u_1 \neq u_2$ and $S_{(u,v_1)(u,v_2)} = |P_{u,v_1} \cap P_{u,v_2}|$ i.e. number of directed edges in common between the two paths P_{u,v_1} and P_{u,v_2}. If the routing is symmetric (this is the case on a tree network), then $R_{(v_1,u_1)(v_2,u_2)} = S_{(u_1,v_1)(u_2,v_2)}$ which implies $R = J^{-1}SJ$ with J-permutation matrix sending $(u,v) \mapsto (v,u)$. Matrix J satisfies $J_{(u_1,v_1)(u_2,v_2)} = \delta_{(u_1,v_2)}\delta_{(v_1,u_2)}$ with the property that $J^{-1} = J^T = J$. Since $AA^T = S + R$, therefore, it is sufficient to find z such that

$$Sz = z, \quad \text{and} \quad SJz = Jz$$

If v_1 and v_2 are siblings in $\mathcal{T}_u^S$, then

$$S_{(u,v_1)(u,v_1)} = S_{(u,v_2)(u,v_2)} = 1 + S_{(u,v_1)(u,v_2)} = 1 + S_{(u,v_2)(u,v_1)}$$

Moreover, $\forall w \neq v_1, v_2$

$$S_{(u,w)(u,v_1)} = S_{(u,w)(u,v_2)}$$

We conclude that $S - \mathbb{I}$ has identical columns (u,v_1) and (u,v_2) for any v_1 and v_2 siblings in $\mathcal{T}_u^S$. Let now u_1, u_2 and v_1, v_2 be two distinct pairs of siblings (always possible to find these two set of distinct siblings in a tree with $|V_B| \geq 4$). Define vector z with

$$\begin{aligned} z_{(u_1,v_1)} &= +1, \quad z_{(u_2,v_1)} = -1 \\ z_{(u_1,v_2)} &= -1, \quad z_{(u_2,v_2)} = +1. \end{aligned}$$

and all other $z_{(w_1,w_2)} = 0$. Since columns (u_1,v_1) and (u_1,v_2) of matrix $S - \mathbb{I}$ are identical and columns (u_2,v_1) and (u_2,v_2) are also identical, we get $(S - \mathbb{I})z = 0$, i.e. $Sz = z$.

But $\tilde{z} = Jz$ has the same structure

$$\begin{aligned} \tilde{z}_{(v_1,u_1)} &= +1, \quad \tilde{z}_{(v_1,u_2)} = -1 \\ \tilde{z}_{(v_2,u_1)} &= -1, \quad \tilde{z}_{(v_2,u_2)} = +1. \end{aligned}$$

Using properties of the columns $(v_1,u_1),(v_1,u_2)$ and $(v_2,u_1),(v_2,u_2)$ of matrix $S - \mathbb{I}$, we get

$(S - \mathbb{I})\tilde{z} = 0$, i.e. $SJz = Jz$ as desired.

The above proof on tightness of bound holds for tree network graphs with $|V_B| \geq 4$. So, the only nontrivial tree which is not covered is 3-star one. Constructing incident matrix A for the 3-star tree as discussed in Example 5, then $z = (+1, -1, -1, +1, +1, -1)^T$ is the eigenvector of matrix AA^T with corresponding eigenvalue 2. $\square$

Example 5. Consider a 3-star network graph with boundary vertices $V_B = \{a, b, c\}$. For each of the boundary vertices we construct its source and receiver trees and label by 1 the edge which appear in two paths, and $2, 3$ the edges incidents to boundary vertices in alphabetical order in each of the 6 trees. We order the set Q of pairs of distinct boundary vertices as $\{ab, ac, ba, bc, ca, cb\}$. We order the elements of the six sets of edges E_a^R, E_b^R, E_c^R, E_a^S, E_b^S, E_c^S according to their label. The signed incidence matrix is then given by

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \end{pmatrix},$$

which results in

$$\left(AA^T\right)^{-1} = \frac{1}{90} \begin{pmatrix} 26 & -7 & -1 & 2 & 2 & -7 \\ -7 & 26 & 2 & -7 & -1 & 2 \\ -1 & 2 & 26 & -7 & -7 & 2 \\ 2 & -7 & -7 & 26 & 2 & -1 \\ 2 & -1 & -7 & 2 & 26 & -7 \\ -7 & 2 & 2 & -1 & -7 & 26 \end{pmatrix}.$$

We consider an example in which all edge weights are 1 except $w^R_{a,1}$ which is $1+\varepsilon$. Thus $\mathcal{W}$ is the vector whose first entry is $1+\varepsilon$ and all other entries 1. The path weights are 2 except for those in the receiver tree rooted at a, for which the path weights are $2+\varepsilon$. Then using (3.13) we compute the consistent edge weights

$$\widetilde{\mathcal{W}} = 1 + \frac{\varepsilon}{90}\big[52,-19,-19,4,-1,5,4,-1,5,2,1,1,14,19,-5,14,19,-5\big]^T.$$

We confirm that, in keeping with Theorem 3.4.1 above, 2 is the smallest eigenvalue of AA^T.

3.5 Positivity Constraint Optimization

The results in Corollary 3.3.5 and Theorem 3.4.1 do not guarantee that all elements of consistent weights are positive. An ad-hoc approach to ensuring positivity would be to prune the edges corresponding to negative $\tilde{w}$ from the graph(s) and recompute new $\widetilde{\mathcal{W}}$, iterating until no further negative entries occur in x. In numerical experiments a solution satisfying the positivity constraint was achieved in several iterations. There is no guarantee that such naive approach would result in an optimal or nearly optimal solution.

Apart from this naive approach, a more systematic way is to use, for example, path-following method (also known as barrier or interior-point method) for convex quadratic problems which modifies the objective function by adding a nonlinear penalty term with a small coupling coefficient. Let us re-formulate our constrained optimization problem as

$$x^* = \operatorname{argmin}_{x\in\mathbb{R}^n} \frac{1}{2}x^T Hx + x^T c \qquad \text{subject to:} \quad Bx = b \quad \text{and} \quad x \geq 0 \tag{3.15}$$

where $H \in \mathbb{R}^{n\times n}$ is positive semidefinite and $B \in \mathbb{R}^{m\times n}$ has full row rank. Introducing the *log-barrier term* modulated by a barrier parameter $\mu \geq 0$ we get

$$x^* = \operatorname{argmin}_{x\in\mathbb{R}^n} \frac{1}{2}x^T Hx + x^T c - \mu\sum_{i=1}^{n}\ln(x_i) \qquad \text{subject to:} \quad Bx = b \tag{3.16}$$

The idea of the algorithm is that solutions of (3.16) converge to solutions of (3.15) as $\mu \to 0$. It has been shown that for an appropriately chosen starting point, $\mathcal{O}(\sqrt{n}\log\frac{n}{\varepsilon})$ iterations are required to be ε-close to the optimal solution [27].

Returning to the our problem of imposing the positivity constrains in (3.8)–(3.9) and (3.11)–(3.12), we need to consider two cases: the matrix A has full row rank or not.

In the context of pruning a tree (Corollary 3.3.4 and Corollary 3.3.5) or ensuring intrinsic consistency in a set of trees (Theorem 3.4.1), we are guaranteed that A has full row rank. In the former case, the standard quadratic optimization problem can be used by setting $H := 2\mathbb{I}$, $c := -2\mathcal{W}_{\tilde{E}}$, $B := \tilde{A}$ and $b := A\mathcal{W}$. In the latter case, the standard quadratic optimization problem can be used by setting $H := 2\mathbb{I}$, $c := -2\mathcal{W}$, $B := A$ and $b := 0$.

With the view of pruning a full reconstructed graph which may not be a tree (Theorem 3.3.1), we explain modifications necessary for a matrix $A \in \mathbb{R}^{m\times n}$ which is not of full row rank. In order to circumvent this rank-deficiency, in general, the matrix A can be reduced to full rank matrix via QR factorization or Gaussian elimination with column pivoting [40]. Applying QR factorization applied to the consistent system $A\mathcal{W} = \mathcal{Z}$ obtains an $m \times m$ orthogonal matrix Q such that

$$QA = \begin{pmatrix} \bar{A} \\ 0 \end{pmatrix}, \quad Q\mathcal{Z} = \begin{pmatrix} \bar{\mathcal{Z}} \\ 0 \end{pmatrix} \tag{3.17}$$

where $\bar{A}$ and $\bar{\mathcal{Z}}$ have the same number of rows and $\bar{A}$ has full row rank. Through this construction of matrix $\bar{A}$, the systems $A\mathcal{W} = \mathcal{Z}$ and $\bar{A}\mathcal{W} = \bar{\mathcal{Z}}$ are equivalent; that is, any vector $\mathcal{W}$ that satisfies one of these equations also satisfies the other. For detailed example on dealing with rank deficient matrix A and application of QR factorization, see [40].

3.5.1 Imposing Sign Constraint from Implementation Point of View

Below we briefly discuss how interior-point methods will be applied to find the minimizer of modified version of constrained quadratic optimization problem

$$x^* = \operatorname{argmin}_{x\in\mathbb{R}^n} \frac{1}{2}x^T Hx + x^T c - \mu \sum_{i=1}^{n} \ln(x_i) \qquad \text{subject to:} \quad Bx = b,$$

appears in subsection 3.5 where $H \in \mathbb{R}^{n\times n}$ is positive semidefinite, $B \in \mathbb{R}^{m\times n}$ has full row rank, and parameter $\mu \geq 0$ is log-barrier parameter (see [22] for detailed discussion). Notice that in order to ensure that the objective function above is well defined, it is required that $x > 0$. The standard steps for solving this quadratic optimization problem as it will be discussed below, consists of (i) forming the Lagrangian, (ii) form the optimality condition, and finally (iii) apply iterative algorithm, for example, primal-dual path-following method. For the Lagrange multiplier $\lambda \in \mathbb{R}^m$, the *Lagrangian* is defined as

$$L(\alpha, \lambda, \mu) := \frac{1}{2}x^T Hx + c^T x - \mu \sum_{i=1}^{n} \ln(x_i) - \lambda^T (Bx - b) \tag{3.18}$$

The conditions for a stationary point of Lagrangian with respect to x and λ satisfy

$$\begin{aligned} Bx - b &= 0 \\ B^T\lambda - \mu X^{-1}e - Hx - c &= 0 \end{aligned} \tag{3.19}$$

for $x > 0$, where $X := \text{diag}\{x_1, \dots, x_n\}$, and e is vector of size n with unit entries. If we let $s = \mu X^{-1}e$ to be a vector of size n, then $x > 0$ implies that $s > 0$ and equation (3.19) can be written as

$$\begin{aligned} Bx - b &= 0 \\ B^T\lambda + s - Hx - c &= 0 \\ Xs &= \mu e \end{aligned} \tag{3.20}$$

for $x > 0$ and $s > 0$ which form the *optimality condition*. Now in order to apply *primal-dual path-following method*, let $w_k := \{x_k, \lambda_k, s_k\}$ be such that x_k strictly feasible for the quadratic optimization problem ($x_k \in \mathbb{R}^n$ satisfies the constraints). The increment $\delta_w := \{\delta_x, \delta_\lambda, \delta_s\}$ should be constructed such that the next iterate $w_{k+1} = w_k + \delta_w$ remains strictly feasible and approaches the central path. If w_k were satisfy equation (3.20) with $\mu = \mu_{k+1}$, neglecting second-order term in equation (3.20), we would have

$$\begin{aligned} -H\delta_x + B^T\delta_\lambda + \delta_s &= 0 \\ B\delta_x &= 0 \\ S\delta_x + X\delta_s &= \mu_{k+1}e - Xs_k \end{aligned} \tag{3.21}$$

where $\Delta X := \mathrm{diag}\{(\delta_x)_1, \ldots, (\delta_x)_n\}$ and $S := \mathrm{diag}\{(s_k)_1, \ldots, (s_k)_n\}$. The solution to equation (3.21) then can be obtained as

$$\begin{aligned} \delta_\lambda &= \Lambda y \\ \delta_x &= \Gamma X B^T \delta_\lambda - y \\ \delta_s &= H\delta_x - B^T\delta_\lambda \end{aligned} \tag{3.22}$$

where $\Gamma := (S + XH)^{-1}$, $\Lambda := (B\Gamma X B^T)^{-1}B$, and $y := \Gamma(Xs_k - \mu_{k+1}e)$. In order to show that matrices Γ and Λ in equation (3.22) are well defined, observe that since $x_k > 0$ and $s_k > 0$, matrices X and S are positive definite. Therefore, the positive definite property of $X^{-1}S + H$ implies that the pseudo-inverse of matrix $S + XH = X(X^{-1}S + H)$ exists. Applying the fact that B is full row rank, then

$$B\Gamma X B^T = B\big(X^{-1}S + H\big)^{-1}B^T$$

is also positive definite and hence nonsingular. As a result, this implies that δ_w has a unique solution. Once δ_w is calculated from equation (3.22), for sufficiently small step size α_k, the updated solution $w_{k+1} = w_k + \alpha_k\delta_w$ remains strictly feasible. In practice an appropriate choice of parameter

μ_{k+1} is

$$\mu_{k+1} = \frac{x_k^T s_k}{n + \rho} \tag{3.23}$$

with $\rho \geq \sqrt{n}$. This would lead to an iteration complexity mentioned in subsection 3.5 for an ε-close solution to the optimal one.

4. NETWORK TOMOGRAPHY ALGORITHM AND SIMULATIONS

This Section provides a composite application of our methods stated in previous Sections to the problem of network inference in five stages: (a) merging primitive binary trees at a single root (b) removing noise from the resulting binary trees by pruning (extrinsic consistency); (c) rendering trees with different roots intrinsically consistent; (d) merging trees using the methods of Section 2; (e) removing noise from the merged network by pruning (extrinsic consistency). Furthermore, the composite application in a model-based simulation will be evaluated. We compare its performance in correctly identifying subset of network paths that experience performance degradation of a common internal edge. The baseline methods are naive pruning and a non-optimal averaging method.

In the first step, we generate a set of network graphs (see Section 4.1), and in each graph simulate the end-to-end packet measurements (details in Section 4.2). The next step is to use the simulated data to infer the logical source and receiver trees rooted at every boundary vertex $b \in V_B$, by applying the recursive clustering approach reviewed in Section 1.3.4. Low weight edges in the resulting binary local logical trees are pruned producing non-binary internal vertices. In order to keep the set of weights extrinsically consistent with the end-to-end measurements, the result of Corollary 3.3.5 is applied on each pruned tree separately. The entire set of trees is then made intrinsically consistent using the results of Section 3.4. This internal consistency is required for fusion of inferred local trees following the graph reconstruction algorithm in Section 2. The last step before evaluating the inferred graph, is to prune edges with small weight due to topological noises raises from the nature of non-exact measurements. This will be achieved by applying the result of Corollary 3.3.5 along with positivity constraint.

4.1 Generating Network Graph

We first describe a framework on constructing set of random network graphs $\mathcal{G} = \{\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(N)}\}$ with each $\mathcal{G}^{(k)} = (G^{(k)}, V_B^{(k)}, \mathcal{P}^{(k)})$ has desirable characteristics.

4.1.1 Random Graphs

In the first part, we construct random graphs along with their corresponding set of paths among the boundary vertex set through the following steps. For each $k \in [N]$,

1. In the first step, a random graph which will be called *underlying graph* $\widehat{G} = (\widehat{V}, \widehat{E}, \widehat{\mathcal{W}})$ is constructed with $|\widehat{V}| = m$ arbitrary and each vertex $u \in \widehat{V}$ has degree deg$(u) = d$ connecting to its randomly selected neighbors. The set $\widehat{\mathcal{W}}$ is generally asymmetric with respect to edge direction in set $\widehat{E}$.

2. Among set $\widehat{V}$, n vertices will be selected randomly which represent the boundary set $V_B = \{b_1, \ldots, b_n\}$.

3. The routing between selected boundary vertices b_i and b_j, i.e. P_{b_i,b_j} with $i \neq j$, follows the shortest path with respect to the weights $\widehat{\mathcal{W}}$. This then form set $\mathcal{P}$.

Note that the assigned weight set $\widehat{\mathcal{W}}$ above is only applied to generate the set of paths $\mathcal{P}$, and the link's performance characteristic for end-to-end measurements will be assigned in the data generation step below (see Section 4.2).

An example of application of the three steps discussed above is shown in Figure 4.1 for graph with six boundary vertices (only the source and receiver trees at vertex b_1 are plotted). It should be noted that in the construction above, if the assigned weight set $\widehat{\mathcal{W}}$ is chosen to be symmetric on edge set $\widehat{E}$, then the communication paths in constructed network graph $\mathcal{G}$ will be symmetric, otherwise asymmetric routing will be obtained.

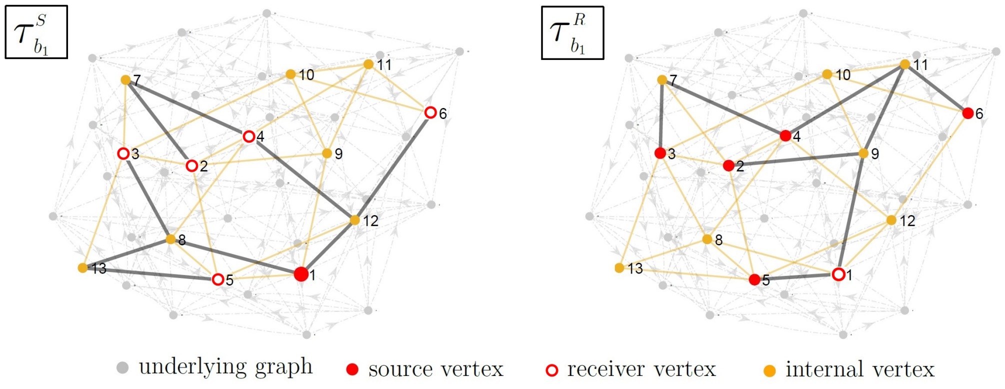


Figure 4.1: Example on randomly generated source $\mathcal{T}_{b_1}^S$ and receiver $\mathcal{T}_{b_1}^R$ trees at vertex b_1.

4.2 Generating Measurements

In this part we will discuss the process of generating end-to-end measurements corresponding to set $\mathcal{P}$ constructed above for network graph $\mathcal{G}$. Although there are various ways to generate this set of measurements, we will follow model-based unicast data acquisition framework described as follows. For graph $\mathcal{G} = (G, V_B, \mathcal{P})$, each directed edge $i \in E$ will experience alternative lossless and lossy states so that a packet transversing the edge will be dropped with probability zero and $p_l^{(i)} \in \mathcal{L}$ respectively, where $\mathcal{L}$ is a set determines the possible edge loss probability. For the experiment with total time T, fraction of packets received successfully for selected source and receiver vertex will be recorded over successive averaging windows of length t_a, i.e. intervals $((k-1)t_a, kt_a)_{k=1}^N$ with $Nt_a = T$. The time that each directed edge i being in successive lossless or lossy states will be equal to $k_s^{(i)} t_a$ and $k_\ell^{(i)} t_a$ with integers $k_s^{(i)}$ and $k_l^{(i)}$ drawn from Poisson distributions with per-determined parameters γ_s and γ_l respectively. The end-to-end measurements for P_{b_i,b_j} with $b_i, b_j \in V_B$ and $b_i \neq b_j$ then will be time series corresponding to variation of the fraction of successfully received packets over time. The schematic of the above process is shown in Figure 4.2 for the network graph with three boundary vertices.

Finally, the edge weight set $\mathcal{W}$ for the network graph $\mathcal{G}$ is constructed by averaging over the edges performance (e.g. empirical variance or mean of packets loss rate) through the generating measurements process overt time T. In other words, for an edge $i \in E$, the averaging is over all

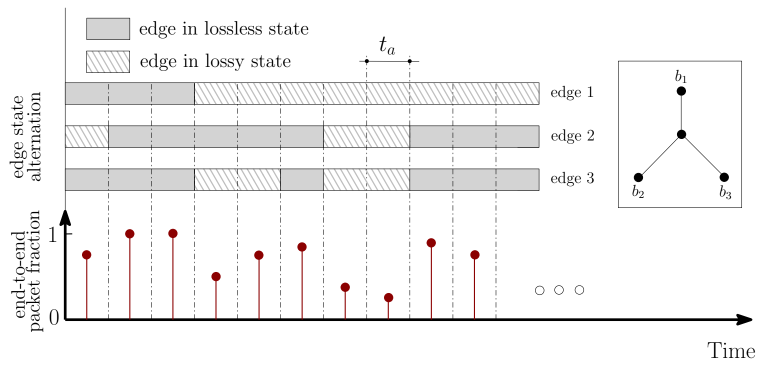


Figure 4.2: Alternation of edges state and measured end-to-end averaged packet fraction over time.

paths $P \in \mathcal{P}$ so that $i \in P$.

4.3 Performance Metric

We now detail metric we use to evaluate the closeness of the inferred network $\widehat{\mathcal{G}}$ to the true network $\mathcal{G}$. Many metrics have been developed for error tolerant graph matching, including graph edit distance [39] and maximal subgraph matching [32]. Closer to the present work, [29] used a notion of receiver matching in tree, namely, that an internal vertex in the true spanning tree associated with a boundary point is well estimate if there is a vertex in the corresponding inferred tree with the same receiver set.

The motivation for our work is to attribute common origins of performance degradation on paths, not limited to paths sharing an endpoint. To this end, we specify a network distance metric that captures the difference in the extent to which individual edges influence the performance of boundary-to-boundary transmission.

4.3.1 Weighted Path Intersection Metric

Given a network graph $\mathcal{G} = (G, V_B, \mathcal{P})$, for each directed edge $e \in E$, let $\mathcal{V}(e)$ denote the set of ordered pairs of boundary vertices $(u, v) \in V_B \times V_B$ such that e is an edge in the path $P_{u,v}$ from u to v. We refer to edges e and e' as equivalent in $\mathcal{V}(e) = \mathcal{V}(e')$. Given another network graph

$\mathcal{G}' = (G', V_B, \mathcal{P}')$ with the identical boundary set V_B, define

$$\mathcal{E}(\mathcal{V}(e), \mathcal{G}') := \bigcap_{(u,v)\in\mathcal{V}(e)} P'_{u,v} \setminus \bigcup_{(u,v)\notin\mathcal{V}(e)} P'_{u,v}. \tag{4.1}$$

We stress that the set $\mathcal{V}(e)$ is determined with respect to the graph $\mathcal{G}$ whereas the paths $P'_{u,v}$ on the right-hand side are taken from the graph $\mathcal{G}'$. Thus the meaning of $\mathcal{E}(\mathcal{V}(e), \mathcal{G}')$ is the set of edges in $\mathcal{G}'$ that are equivalent (from the point of view of boundary-to-boundary transmission) to the edge e of $\mathcal{G}$. Specifically, they lie in the intersection of paths from $\mathcal{P}'$ between endpoint pairs $\mathcal{V}(e)$, but in no other paths in $\mathcal{P}'$. In general, the set $\mathcal{E}(\mathcal{V}(e), \mathcal{G}')$ may be empty, but if $\mathcal{G}' = \mathcal{G}$, the set $\mathcal{E}(\mathcal{V}(e), \mathcal{G})$ is simply the set of edges equivalent to e as per definition above.

Let W be a weight function from the set of subsets of $E(G)$ to non-negative integers; let W' be a similar function on the graph $\mathcal{G}'$. We define

$$Q(\mathcal{G}, \mathcal{G}') = \frac{\sum_{e\in E} \left| W_{\mathcal{E}(\mathcal{V}(e),\mathcal{G})} - W'_{\mathcal{E}(\mathcal{V}(e),\mathcal{G}')} \right|}{\sum_{e\in E} W_{\mathcal{E}(\mathcal{V}(e),\mathcal{G})}}, \tag{4.2}$$

where in both summation only one e from each equivalence class is used.

The relative error in estimating $W_{\mathcal{E}(\mathcal{V}(e),\mathcal{G})}$ is $\left| W_{\mathcal{E}(\mathcal{V}(e),\mathcal{G})} - W'_{\mathcal{E}(\mathcal{V}(e),\mathcal{G}')} \right| / W_{\mathcal{E}(\mathcal{V}(e),\mathcal{G})}$. Thus Q is the average over equivalence classes of edges $e \in E$ of the relative error in estimating $W_{\mathcal{E}(\mathcal{V}(e),\mathcal{G})}$, weighted by $W_{\mathcal{E}(\mathcal{V}(e),\mathcal{G})}$ i.e., Q is more sensitive to errors in estimating large weights $W_{\mathcal{E}(\mathcal{V}(e),\mathcal{G})}$ than for small ones.

The metric assumes that the weights in $\mathcal{G}$ and $\mathcal{G}'$ to be comparable, e.g., because W' estimates W. This is not the case for covariance based estimation as described in Section 1. One approach to this is to replace W with weights comparable to W', e.g. by computing edge estimator covariances from a model of packet performance. We here propose two approaches that can be used directly with any weights W and W'.

(TM$_1$) Set $W'_\emptyset = 0$ but take $W'_\mathcal{E} = W_\mathcal{E} = \sum_{e \in \mathcal{E}} w_e$ otherwise. In other words, we penalize e by its weight if there no edge e' in $\mathcal{G}'$ functionality equivalent to it.

(TM$_2$) As in TM$_1$ but with $W_\mathcal{E} = 1$ for all nonempty subpaths P.

4.4 Numerical Results

In this subsection performance of the reconstruction framework will be evaluated on set of randomly generated network graphs discussed above. The evaluation metric will be based on (TM$_1$) and (TM$_2$). In all numerical simulations, we chose the parameters of Poisson distribution which determine the number of averaging windows an edge experiences in lossy or lossless states to be $\gamma_l = \gamma_s = 10$ (the initial state of each edge is chosen randomly). Moreover the loss probability of edge i in lossy state is chosen randomly from $p_l^{(i)} \in \mathcal{L} = \{0.05, 0.10\}$ if edge i experiences loss state. The number of packets sent form source $b_i \in V_B$ to the set of receiver boundary vertices $V_B \setminus \{b_i\}$ in each averaging window t_a will be fixed and equal to 10^3 in all simulations. Thereby, number of averaging windows will control the total number of packets sent in unicast mode.

Figure 4.3 shows the error plot of topological inference over the number of averaging windows for two selected samples of network graph with $|V_B| = 6$ and $|V_B| = 12$ (for each number of averaging windows 50 experiments are conducted). The convergence to full inference accuracy over the number of averaging windows is clear. For the larger network graph (right), higher number of averaging windows is required to achieve same level of inference accuracy happens for six boundary vertex graph (left). The reason is that as the size of network graph increases, the variance of estimators in tree level identification increases. Hence mistaken pairing of non-sibling vertices, or erroneous inclusion or exclusion of vertices in a group is more likely to occur. For same experiment over the network, topological metric TM$_1$ reports slightly better algorithmic performance compared to TM$_2$ metric and the difference of these two metrics reduce (converges to zero) over the inference accuracy. Figure 4.4(left) shows the error plot of topological inference over the fraction of edges experiencing lossy state for 20 randomly selected networks graphs with six boundary vertices. For each network graph and selected fraction of lossy edges, 50 experiments

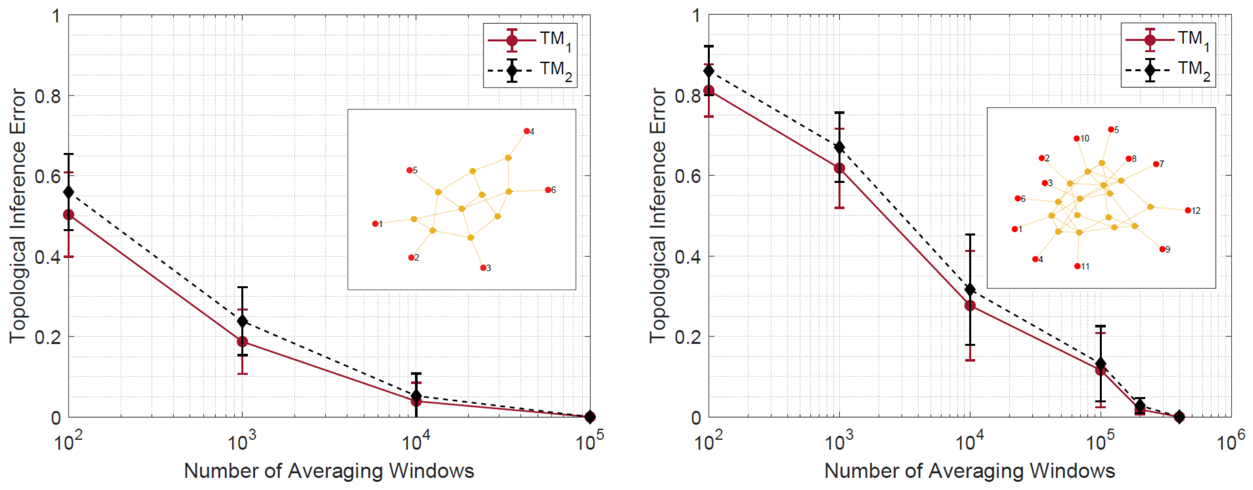


Figure 4.3: Dependence of topological inference error on the number of averaging windows evaluated based on metric TM$_1$ and TM$_2$. Left: network graphs graph with $|V_B| = 6$ and right: $|V_B| = 12$.

are conducted where edges which experience both lossless and lossy states are selected uniformly in random. Obviously, those edges which are not selected as lossy ones stay in lossless state over the whole experiment. The plot shows that once the fraction of lossy edges increases, the average of inference accuracy approaches to the selected lossy fraction while uncertainty in inference of identifiable subgraph (part of graph constructed by lossy edges) increases for low fraction of lossy edges. The difference of topological inference metrics TM$_1$ and TM$_2$ decreases for larger fraction of lossy edges while the two metrics start deviating from each other when the fraction of lossy edges is less than 0.7. In the numerical simulations, an edge $i \in E'$ belonging to the reconstructed network graph $\mathcal{G}'$ will be pruned if $w'_i/w'_{\max} \leq \delta$ with $\delta \in [0, \delta_{\max}]$ is called pruning factor with $\delta_{\max} \leq 1$.

In Figure 4.4(right), the dependence of density of pruned edges on pruning factor ($\delta_{\max} = 0.2$) for different values of number of averaging windows (one sample is selected) is plotted. The result shows that once the number of averaging windows increases, then small value of pruning factor is sufficient to prune relatively large number of edges with small weights due to topological noise while achieving the same number of pruned edges requires large value of pruning factor

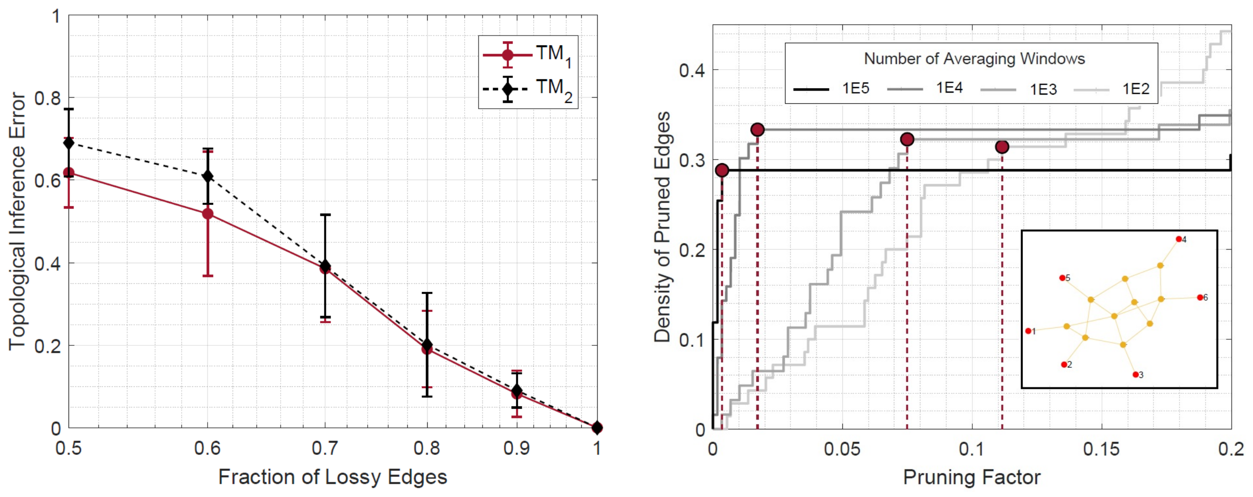


Figure 4.4: Left: Dependence of topological inference error on fraction of lossy edges. Right: Sample on dependence of density of pruned edges on pruning factor for different number of averaging windows. The vertical dashed lines are the selected pruning factor for the corresponding experiment.

for inference based on low number of averaging windows. This is in line with the fact that in the limiting case when the edge weights approaches to the exact value, then the pruning factor required to pruned edges exist due to topological noise approaches to zero. Letting $(\delta_k)_{k=1}^m$ with $\delta_m \leq \delta_{\max}$ to be increasingly ordered point of discontinuities for the plot of dependence of density of pruned edges over pruning factor (e.g. see Figure 4.4(right) for selected number of averaging window), then an ad-hoc approach to choose the pruning factor $\delta^* = \delta_{k^*}$ in each experiment is to find $k^* = \operatorname{argmax}_{k\in[m-1]}(\delta_{k+1} - \delta_k)$. Another possible heuristic approach to prune the inferred trees is to follow Remark 3.4.2.

5. SUMMARY

In this thesis we considered the problem of reconstructing a network graph from the measurements of PCD: the common length of any two paths sharing an origin or a destination.

We first established necessary and sufficient conditions for a network graph to be reconstructible and describe the reconstruction algorithm. It is shown that when the reconstruction is attempted on a graph violating those conditions, the result of our algorithm is a minimal graph fitting the available data. More precisely, among the network graphs that have the same PCD, the reconstructed graph has the minimal number of links, no vertices of degree 2 and the minimal average degree among vertices of degree 3 or more. Our framework is general and no assumption is made on the symmetricity of routing or edge weights of original graph is required. However, for graphs with symmetric routing less restrictive conditions are shown to apply and a more specialized algorithm is presented.

We next extended the method of graph reconstruction to PCD in which sampling and measurement noise leads to inconsistencies in path weights reported for different paths. The two main types of inconsistencies due to the nature of non-exact measurements are formulated in the form of extrinsic and intrinsic types, and solved in a least-squares framework. In each case, the solutions are expressed in term of the Moore-Penrose pseudo-inverse of a generalized routing matrix which express the linear constraints between path and edge weights. This extension provided a unified framework on consistency merging and pruning subgraphs which reduce the problem in network inference through the following five steps: (a) merging primitive binary trees at a single root (b) removing topological noise from the resulting binary trees by pruning (extrinsic consistency); (c) rendering trees with different roots intrinsically consistent; (d) merging trees using the methods of [3]; and (e) removing topological noise from the merged network by pruning (extrinsic consistency).

Finally, we discussed the application of prescribed five composite steps on inference of randomly generated network graphs using model based data. Although the proposed metric on inference accuracy is general in the sense that it can capture both fully topological base and weighted based accuracy, but in this work we focused on its topological version.

REFERENCES

[1] N. Alon, Y. Caro, I. Krasikov, and Y. Roditty. Combinatorial reconstruction problems. *J. Combin. Theory Ser. B*, 47(2):153–161, 1989.

[2] V. Arya and D. Veitch. Sparsity without the complexity: Loss localisation using tree measurements. In Robert Bestak, Lukas Kencl, Li Erran Li, Joerg Widmer, and Hao Yin, editors, *NETWORKING 2012*, pages 289–303, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[3] M. I. Belishev. Recent progress in the boundary control method. *Inverse Problems*, 23(5):R1–R67, 2007.

[4] M. I. Belishev and N. Wada. On revealing graph cycles via boundary measurements. *Inverse Problems*, 25(10):105011, 21, 2009.

[5] M. Ettehad, N. G. Duffield, and G. Berkolaiko. Optimizing consistent merging and pruning of subgraphs in network tomography. *ArXiv e-prints*, page arXiv:1908.03519, August 2019.

[6] L. Borcea, V. Druskin, A. V. Mamonov, and F. Guevara Vasquez. Pyramidal resistor networks for electrical impedance tomography with partial boundary measurements. *Inverse Problems*, 26(10):105009, 36, 2010.

[7] R. Caceres, N. G. Duffield, J. Horowitz, and D. F. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions on Information Theory*, 45(7):2462–2480, Nov 1999.

[8] M. Cheney, D. Isaacson, and J. C. Newell. Electrical impedance tomography. *SIAM Review*, 41(1):85–101, 1999.

[9] D. B. Chua, E. D. Kolaczyk, and M. Crovella. Network kriging. *IEEE Journal on Selected Areas in Communications*, 24(12):2263–2272, Dec 2006.

[10] F. J. Chung, A. C. Gilbert, J. G. Hoskins, and J. C. Schotland. Optical tomography on graphs. *Inverse Problems*, 33(5):055016, 21, 2017.

[11] N. G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley. Multicast topology inference from measured end-to-end loss. *IEEE Transactions on Information Theory*, 48(1):26–45, Jan 2002.

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

[13] N. G. Duffield and Francesco Lo Presti. Network tomography from measured end-to-end delay covariance. *IEEE/ACM Trans. Netw.*, 12(6):978–992, 2004.

[14] N. G. Duffield, F. L. Presti, V. Paxson, and D. F. Towsley. Network loss tomography using striped unicast probes. *IEEE/ACM Trans. Netw.*, 14(4):697–710, 2006.

[15] S. N. Evans and D. Lanoue. Recovering a tree from the lengths of subtrees spanned by a randomly chosen sequence of leaves. *Adv. in Appl. Math.*, 96:39–75, 2018.

[16] G. Liang and B. Yu. Maximum pseudo likelihood estimation in network tomography. *IEEE Transactions on Signal Processing*, 51(8):2043–2053, Aug 2003.

[17] H. Singhal and G. Michailidis. Identifiability of flow distributions from link measurements with applications to computer networks. *Inverse Problems*, 23(5):1821–1849, 2007.

[18] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of hot-potato routing in IP networks. *SIGMETRICS Perform. Eval. Rev.*, 32(1):307–319, June 2004.

[19] Y. Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the American Statistical Association*, 91(433):365–377, 1996.

[20] B. Yu, J. Cao, D. Davis, and S. Vander Wiel. Time-varying network tomography: router link data. In *2000 IEEE International Symposium on Information Theory (Cat. No.00CH37060)*, pages 79–, 2000.

[21] Y. Zhang, M. Roughan, N. G. Duffield, and A. Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. In *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '03, pages 206–217, New York, NY, USA, 2003. ACM.

[22] A. Antoniou and W. S. Lu. Practical optimization: Algorithms and engineering applications. *Springer*, 2007.

[23] G. Berkolaiko, N. G. Duffield, M. Ettehad, and K. Manousakis. Graph reconstruction from path correlation data. *Inverse Problems*, 35:1–25, 2018.

[24] T. Bu, N. G. Duffield, F. L. Presti, and D. F. Towsley. Network tomography on general topologies. In *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2002, June 15-19, 2002, Marina Del Rey, California, USA*, pages 21–30, 2002.

[25] R. Caceres, N. G. Duffield, J. Horowitz, F. L. Presti, and D. Towsley. Loss-based inference of multicast network topology. In *Proceedings of the 38th IEEE Conference on Decision and Control (Cat. No.99CH36304)*, volume 3, pages 3065–3070, Dec 1999.

[26] R. Caceres, N. G. Duffield, J. Horowitz, and D. F. Towsley. Multicast-based inference of network-internal loss characteristics. *IEEE Transactions on Information Theory*, 45(7):2462–2480, Nov 1999.

[27] X. Cai, G. Wang, and Z. Zhang. Complexity analysis and numerical implementation of primal-dual interior-point methods for convex quadratic optimization based on a finite barrier. *Numerical Algorithms*, 62:289–306, 2013.

[28] M. Coates, M. Rabbat, and R. Nowak. Merging logical topologies using end-to-end measurements. In *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, IMC '03, pages 192–203, New York, NY, USA, 2003. ACM.

[29] N. G. Duffield, J. Horowitz, F. L. Presti, and D. Towsley. Multicast topology inference from measured end-to-end loss. *IEEE Transactions on Information Theory*, 48(1):26–45, Jan 2002.

[30] N. G. Duffield and F. L. Presti. Multicast inference of packet delay variance at interior network links. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, volume 3, pages 1351–1360, March 2000.

[31] N. G. Duffield, F. L. Presti, V. Paxson, and D. Towsley. Inferring link loss using striped unicast probes. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, volume 2, pages 915–923, April 2001.

[32] R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximal cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1168–1180, Nov 1989.

[33] F. Lopresti, N. G. Duffield, J. Horowitz, and D. Towsley. Multicast-based inference of network-internal delay distributions. *IEEE/ACM Trans. Netw.*, 10(6):761–775, Dec. 2002.

[34] H. X. Nguyen and P. Thiran. Network loss inference with second order statistics of end-to-end flows. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 227–240, New York, NY, USA, 2007. ACM.

[35] R. Penrose. On best approximate solution of linear matrix equations. *Proceedings of the Cambridge Philosophical Society*, 52:17–19, 1956.

[36] M. Rabbat, R. Nowak, and M. Coates. Multiple source, multiple destination network tomography. In *IEEE INFOCOM 2004*, volume 3, pages 1628–1639, March 2004.

[37] A. Sabnis, R. K. Sitaraman, and D. Towsley. OCCAM: An Optimization-Based Approach to Network Inference. *ArXiv e-prints*, page arXiv:1806.03542, June 2018.

[38] P. Sattari, M. Kurant, A. Anandkumar, A. Markopoulou, and M. G. Rabbat. Active learning of multiple source multiple destination topologies. *IEEE Transactions on Signal Processing*, 62(8):1926–1937, April 2014.

[39] L. G. Shapiro and R. M. Haralick. Structural descriptions and inexact matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(5):504–519, Sept 1981.

[40] S. J. Wright. Primal-dual interior-point method. SIAM Philadelphia, 1997.