1 Preliminaries

This chapter covers preliminary materials required to understand the presentation in the following chapters.

1.1 Background on Graphs

Unless stated otherwise, we model the network topology as a connected, undirected, simple graph $\mathcal{G}$. In the text that follows we introduce some basic notions from graph theory. We refer the reader to [1] for details.

1.1.1 Graph-Theoretic Definitions

DEFINITION 1.1 (Graph) *An undirected graph $\mathcal{G}$ is defined as a pair $(V(\mathcal{G}), L(\mathcal{G}))$, where $V(\mathcal{G})$ is the set of nodes in $\mathcal{G}$ and $L(\mathcal{G})$ the set of links. A graph is* simple *if it contains no self-loop (a link beginning/ending at the same node) or parallel links (multiple links between the same pair of nodes). The* degree of $\mathcal{G}$*, denoted by $|\mathcal{G}|$, is the number of nodes in the graph; the* order of $\mathcal{G}$*, denoted by $||\mathcal{G}||$, is the number of links in the graph.*

In the sequel, we will simplify denote $V(\mathcal{G})$ by V and $L(\mathcal{G})$ by L when the graph is clear from the context. We will use "node"/"vertex" and "link"/"edge" interchangeably.

DEFINITION 1.2 (Degree)
(a) *The degree of a node v, denoted by* $\deg(v)$*, is the number of links incident to v.*
(b) *The degree of a subgraph $\mathcal{D}$ of a graph $\mathcal{G}$, denoted by* $\deg(\mathcal{D})$*, is the number of links with one endpoint in $\mathcal{D}$ and one endpoint outside $\mathcal{D}$.*

DEFINITION 1.3 (Vertex/Edge Connectivity)
(a) *A connected graph $\mathcal{G} = (V, L)$ is said to be k-vertex-connected (or k-edge-connected) if $k \leq |V| - 1$ (or $k \leq |L| - 1$) and deleting any subset of up to $k - 1$ vertices (or edges) does not disconnect $\mathcal{G}$.*
(b) *The greatest integer k such that $\mathcal{G}$ is k-vertex-connected (or k-edge-connected) is the* vertex connectivity *(or* edge connectivity*) of $\mathcal{G}$.*

https://doi.org/10.1017/9781108377119.002 Published online by Cambridge University Press

DEFINITION 1.4 (Cut) *A vertex cut (or edge cut) in $\mathcal{G}$ is a set of nodes (or links) whose removal increases the number of connected components in $\mathcal{G}$. In particular,*
(a) *a cut vertex is a node that forms a vertex cut by itself;*
(b) *a* 2-vertex cut *is a set of two nodes $\{v_1, v_2\}$ such that v_1 or v_2 alone does not form a vertex cut, but together they form a vertex cut; and*
(c) *a* bridge *is a link that forms an edge cut by itself.*

DEFINITION 1.5 (Induced Subgraph) *An* induced subgraph *$\mathcal{G}'$ of $\mathcal{G}$ is a subgraph such that for any pair of vertices v and w in $\mathcal{G}'$, vw is an edge in $\mathcal{G}'$ if and only if vw is an edge in $\mathcal{G}$.*

DEFINITION 1.6 (k-Vertex/Edge-Connected Component)
(a) *A* k-vertex-connected component*, a.k.a.* k-connected component*, of $\mathcal{G}$ is a maximal subgraph of $\mathcal{G}$ that is either (1) k-vertex-connected or (2) a complete graph with up to k vertices. The case of $k = 2$ is also called a* biconnected component *and $k = 3$ a* triconnected component.
(b) *A* k-edge-connected component *of $\mathcal{G}$ is a maximal subgraph of $\mathcal{G}$ that is k-edge-connected.*

Definition 1.6 generalizes the notion of connected component. By this definition, a connected component is both a 1-connected component and a 1-edge-connected component.

1.1.2 Graph Algorithms

Solutions in the following chapters will leverage several existing graph algorithms, particularly graph decomposition algorithms that compute various types of connected components. For completeness, we outline in Algorithms 1 and 2 the basic steps of the biconnected and triconnected component decomposition algorithms in [2, 3].

Algorithm 1 Biconnected Component Decomposition

Input: Connected graph $\mathcal{G}$
Output: All biconnected components in $\mathcal{G}$
1 Each vertex v in $\mathcal{G}$ is assigned a "pre" and a "low" number [2], where the "pre" number is obtained during the depth-first search while the "low" number is computed according to its connections to neighboring vertices;
2 Identify all cut vertices based on the "pre" and "low" values;
3 Store the visited edges in a stack via depth-first search;
4 When a cut vertex is discovered, pop the edges in the stack to get all edges in the same biconnected component;

The basic idea behind these algorithms is to find the cuts that define the components of interest. Intuitively, a biconnected component is a subgraph connected to the rest of the graph by cut vertices, and a triconnected component within a biconnected component is a subgraph connected to the rest by 2-vertex cuts. For instance, Fig. 1.1b shows

https://doi.org/10.1017/9781108377119.002 Published online by Cambridge University Press

Algorithm 2 Triconnected Component Decomposition

Input: A connected preprocessed graph $\mathcal{G}$
Output: All triconnected components in $\mathcal{G}$

Partition $\mathcal{G}$ into biconnected components $\mathcal{B}_1, \mathcal{B}_2, \ldots$, according to Algorithm 1;
foreach *biconnected component* $\mathcal{B}_i$ **do**
  Find a cycle $\mathcal{C}$ in $\mathcal{B}_i$;
  if $\mathcal{C}$ *does not exist* **then**
    $\mathcal{B}_i$ is a triconnected component;
  else
    $\mathcal{B}'_i = \mathcal{B}_i - \mathcal{C}$;
    Localize all 2-vertex-cuts (see Definition 3.5) by finding cycles in each connected component within $\mathcal{B}'_i$;
    Follow a depth-first search to store the edges belonging to the same triconnected component;

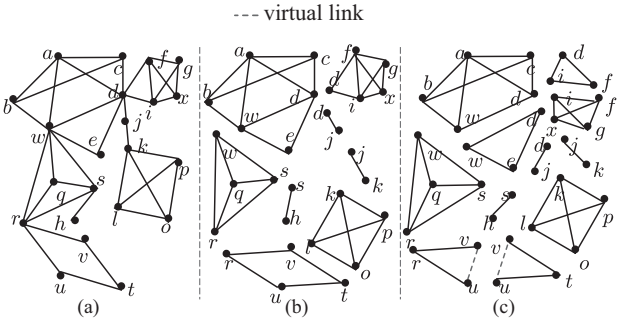

Figure 1.1 (a) Original graph. (b) Biconnected components. (c) Triconnected components. © 2014 IEEE. Reprinted, with permission, from [4].

the biconnected components of Fig. 1.1a, separated by cut vertices d, j, k, w, s, and r. Figure 1.1c shows the triconnected components, separated by the aforementioned cut vertices and 2-vertex cuts $\{w, d\}$, $\{f, i\}$, and $\{u, v\}$.

However, it is possible that not all the subgraphs separated by cut vertices and 2-vertex cuts are valid triconnected components. To avoid this issue, we need to preprocess the graph by adding *virtual links* as follows: if $\exists$ a 2-vertex cut whose vertices are not neighbors (e.g., $\{u, v\}$ in Fig. 1.1), connect them by a virtual link; repeat this on the resulting graph until no such cut exists. Note that the set of virtual links may not be unique (e.g., in Fig. 1.1, preprocessing may add a virtual link (u, v) or (r, t), but not both). The output of Algorithm 2 comprises all the triconnected components in the preprocessed graph. In the sequel, we assume that the graph is always preprocessed before a triconnected component decomposition, and the nodes that are cut vertices or part of 2-vertex cuts are called *separation vertices* (e.g., w, d, f, i, j, k, s, r, u, and v in Fig. 1.1 with an added virtual link (u, v)).

https://doi.org/10.1017/9781108377119.002 Published online by Cambridge University Press

1.2 Background on Linear Algebra

For additive constant link metrics (e.g., means/variances of delays or jitters, log-success rates), the metric on a path equals the summation of the metrics on the traversed links. Therefore, as shown in (2.1), the measurement system can be modeled by a system of linear equations $\mathbf{Rw} = \mathbf{c}$, where the coefficient matrix $\mathbf{R}$ largely determines the structure of the solution. To study the structure of the solution, we will use the following notions from linear algebra. We refer the reader to standard books on linear algebra such as [5] for more details.

DEFINITION 1.7 (Linear Independence)
(a) *A set of vectors* $\mathbf{x}_1, \ldots, \mathbf{x}_n$ *are* linearly independent *of each other if none of them can be written as a linear combination of the other vectors, i.e.,* $a_1\mathbf{x}_1 + \cdots + a_n\mathbf{x}_n = \mathbf{0}$ *implies* $a_1 = \cdots = a_n = 0$.
(b) *A* basis *of a linear space is a maximal set of vectors in this space that are linearly independent of each other.*

DEFINITION 1.8 (rank)
(a) *The* rank *of a matrix* $\mathbf{R}$*, denoted by rank*($\mathbf{R}$)*, is the number of linearly independent rows (or columns).*
(b) *A matrix* $\mathbf{R}$ *has full column rank if all the columns are linearly independent of each other, i.e., rank*($\mathbf{R}$) *equals the number of columns.*

DEFINITION 1.9 (Null Space) *The* null space *of an* $m \times n$ *matrix* $\mathbf{R}$ *is the set of all* $n \times 1$ *vectors* $\mathbf{x}$ *satisfying* $\mathbf{Rx} = \mathbf{0}$.

Additionally, given a square matrix A, we will use A^{-1} to denote its inverse (assuming that A is invertible), $\det(A)$ to denote its determinant, and $\mathrm{Tr}(A)$ to denote its trace; see their standard definitions in [5].

1.3 Background on Parameter Estimation

For stochastic link metrics (e.g., realizations of delays, jitters, losses), network tomography aims at inferring the parameters $\boldsymbol{\theta} = (\theta_l)_{l \in L}$ (e.g., link success rates) that characterize the distributions of these link metrics from the observed realizations $\mathbf{x} = (x_t)_{t=1}^N$ of the corresponding path performance metrics. Therefore, the network tomography problem is cast as a parameter estimation problem in this case. Let $f(x; \boldsymbol{\theta})$ denote the conditional probability of observing x under parameter $\boldsymbol{\theta}$. We will leverage the following results from estimation theory. We refer the reader to a book on estimation theory, such as [6], for details.

https://doi.org/10.1017/9781108377119.002 Published online by Cambridge University Press

1.3.1 Fisher Information Matrix and the Cramér–Rao Bound

Let X have probability distribution $f(x;\boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \mathbb{R}^s$. Suppose that

1. X has common support for all values of $\boldsymbol{\theta}$, i.e., $\{x : f(x;\boldsymbol{\theta}) > 0\}$ is independent of $\boldsymbol{\theta}$.
2. The derivative $\partial f(x;\boldsymbol{\theta})/\partial\theta_i$ exists and is finite, $i = 1,\ldots,s$.

DEFINITION 1.10 *The* Fisher information matrix (FIM) *is the* $s \times s$ *matrix*

$$I(\boldsymbol{\theta}) = [I_{ij}(\boldsymbol{\theta})]_{i,j=1}^{s}, \tag{1.1}$$

where

$$I_{ij}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\theta}}\left[\frac{\partial}{\partial\theta_i}\log f(x;\boldsymbol{\theta}) \cdot \frac{\partial}{\partial\theta_j}\log f(x;\boldsymbol{\theta})\right]. \tag{1.2}$$

If $\log f(x;\boldsymbol{\theta})$ *has a second derivative, then*

$$I_{ij}(\boldsymbol{\theta}) = -\mathbb{E}\left[\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log f(x;\boldsymbol{\theta})\right]. \tag{1.3}$$

The significance of the FIM is due to the following lower bound on the covariance of an unbiased estimator of $\boldsymbol{\theta}$, called the *Cramér–Rao bound (CRB)*.

DEFINITION 1.11 *Let* $\boldsymbol{\mu}(X)$ *be an unbiased estimator of* $\boldsymbol{\theta}$ *given observations* X. *Then*

$$\mathrm{Cov}_{\boldsymbol{\theta}}(\boldsymbol{\mu}) \geq I(\boldsymbol{\theta})^{-1}, \tag{1.4}$$

where $\mathrm{Cov}_{\boldsymbol{\theta}}(\boldsymbol{\mu}) := \mathbb{E}[(\boldsymbol{\mu} - \boldsymbol{\theta})(\boldsymbol{\mu} - \boldsymbol{\theta})^T]$. *Note that this yields the following lower bound on the variance for the estimate of* θ_i*:*

$$\mathrm{Var}_{\boldsymbol{\theta}}(\mu_i) \geq (I(\boldsymbol{\theta})^{-1})_{ii}. \tag{1.5}$$

Estimator μ is said to be *efficient* when it is unbiased and satisfies (1.4) with equality.

1.3.2 Maximum Likelihood Estimator

The maximum likelihood estimator (MLE) $\hat{\boldsymbol{\theta}}^{\mathrm{MLE}}$ is a mapping from a given set of observations $\mathbf{x}$ to the parameter value that maximizes the likelihood of these observations, i.e.,

$$\hat{\boldsymbol{\theta}}^{\mathrm{MLE}}(\mathbf{x}) := \arg\max_{\boldsymbol{\theta}} f(\mathbf{x};\boldsymbol{\theta}). \tag{1.6}$$

The MLE plays an important role in estimation theory. Implicit in any application of FIM is the assumption that the adopted estimator is unbiased and achieves the CRB, and thus the CRB characterizes the estimation error. In this regard, the MLE has the

https://doi.org/10.1017/9781108377119.002 Published online by Cambridge University Press

property that if an efficient estimator exists for a parameter, then the MLE of that parameter is efficient [6]. Moreover, although the MLE may not be unbiased and thus not efficient for finite sample sizes, the MLE is asymptotically efficient under mild regularity conditions [7]; i.e., its expectation converges to the true parameter at a rate approximating that of the CRB. Therefore, the variance of the MLE will approximate the CRB as the number of observations becomes large.

1.4 Background on Routing Mechanisms

With the exception of Chapter 8, we assume that all the end-to-end measurements are taken via *unicast*; i.e., each probing packet is forwarded without replication from a single source to a single destination. Given a network topology $\mathcal{G}$ and the set of monitors M, the set of paths that can be measured (a.k.a. measurement paths) P is determined by the mechanism used to route probes between monitors.

Depending on the flexibility of routing, we classify routing mechanisms into the following types: (1) uncontrollable routing (UR), where P contains paths between monitors specified by the underlying routing protocol of the network; (2) controllable cycle-free routing (CFR), where P contains all simple (e.g., cycle-free) paths between monitors; (3) controllable cycle-based routing (CBR), where P contains every path between monitors that does not contain repeated links (but repeated nodes, i.e., cycles, are allowed); and (4) arbitrarily controllable routing (ACR), where P contains any path between monitors.

Remark While ACR dominates the others in terms of the flexibility of measurements, each of these cases can represent practical constraints in some network environments. Specifically, ACR models the ideal case when strict source routing [8] or its equivalence (e.g., software-defined networking (SDN) routing [9]) is available for setting up measurement paths, CBR models the constraints when monitoring all-optical networks [10], CFR models the constraints when setting up measurement paths by multiprotocol label switching (MPLS) [11] or virtual private networks (VPNs) over IP [12], and UR models the most common scenario when only the default routing paths (e.g., least-cost paths) for data packets are monitored.

References

[1] R. Diestel, *Graph Theory*. Heidelberg: Springer, 2005.

[2] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 146–160, June 1972.

[3] J. E. Hopcroft and R. E. Tarjan, "Dividing a graph into triconnected components," *SIAM Journal on Computing*, vol. 2, pp. 135–158, 1973.

[4] L. Ma, T. He, K. K. Leung, A. Swami and D. Towsley, "Inferring Link Metrics From End-To-End Path Measurements: Identifiability and Monitor Placement," in *IEEE/ACM Transactions on Networking*, vol. 22, no. 4, pp. 1351–1368, Aug. 2014.

https://doi.org/10.1017/9781108377119.002 Published online by Cambridge University Press

[5] G. H. Golub and C. F. Van-Loan, *Matrix Computations*. Baltimore: Johns Hopkins University Press, 1996.
[6] H. L. V. Trees, *Detection, Estimation, and Modulation Theory*. Hoboken, NJ: John Wiley & Sons, 2004.
[7] H. E. Daniels, "The asymptotic efficiency of a maximum likelihood estimator," in *Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, 1961, pp. 151–163.
[8] "DARPA Internet Program Protocol Specification," www.ietf.org/rfc/rfc0791.txt.
[9] "OpenFlow switch specification," Open Networking Foundation, Version 1.4.0, October 2013.
[10] S. S. Ahuja, S. Ramasubramanian, and M. Krunz, "SRLG failure localization in optical networks," *IEEE/ACM Transactions on Networking*, vol. 19, no. 4, pp. 989–999, August 2011.
[11] www.ietf.org/rfc/rfc3031.txt.
[12] A. Kumar, R. Rastogi, A. Silberschatz, and B. Yener, "Algorithms for provisioning virtual private networks in the hose model," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 565–578, August 2002.

https://doi.org/10.1017/9781108377119.002 Published online by Cambridge University Press