Inference Methods for Message Endpoint Localization in Networks

by

Derek H. Justice

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2006

Doctoral Committee:

Professor Alfred O. Hero III, Chair
Professor Romesh Saigal
Professor Demosthenis Teneketzis
Associate Professor George Michailidis

ACKNOWLEDGEMENTS

I am grateful to my advisor, Prof. Hero, for providing guidance in this work while giving me the freedom and support to move at my own pace. I also appreciate the input and time of the rest of my committee: Profs. Michailidis, Saigal, and Teneketzis. This work benefited from interaction with collaborators from Wisconsin, Rice, and Applied Signal Technology at the yearly Network Tomography Workshop. In particular, I thank Mike Rabbat of Wisconsin for helpful discussions and his sharing of experimental data used in Chapter 4. Financial support was provided by a departmental graduate fellowship and an NSF grant under ITR contract CCR-0325571.

On a personal note, I thank my family for calling to check in on me and my friends for providing enjoyable diversions throughout the process of developing this material. Many thanks also go to my wonderful girlfriend, Dan Ruan, for her support and advice on some of the theorem proofs in Chapter 4. Lastly, and above all, I thank the Lord for giving me the health, motivation, and intelligence necessary to produce this work.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

CHAPTER I

Introduction

1.1 Network Models and Message Endpoint Localization

A network or graph is a mathematical structure that summarizes connections between some objects of interest. The objects, referred to as nodes or vertices in the network, may be anything from computers and telephones to humans and atoms. Connections between the objects, called links or edges, signify the existence of some relevant relationship between the two objects connected. Links in a telephone network might represent physical telephone lines; links in a network of humans (called a social network) might indicate the connected individuals have had lunch together in the last week; links in a chemical graph can indicate bonded atoms. The power of networks as tools for representing relationships is clearly immense. An extensive review of current network models and research in this area is given in [70]. In many applications, it is useful to have a means for comparing two networks. For example, the pharmaceutical industry frequently maintains databases of chemical graphs representing the molecular structures of known drugs. When designing a new medication, it is useful to compare the graphs in the database to some query structure in order to avoid patent infringements while getting ideas from existing medications. A distance metric for graphs is clearly fundamental to this operation. We develop such

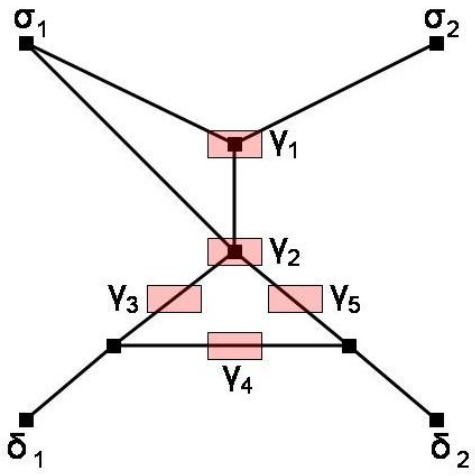


Figure 1.1: Diagram of the measurement apparatus on a sample network for the message endpoint localization problem. Probing sites are sources $\Sigma = \{\sigma_1, \sigma_2\}$ and destinations $\Delta = \{\delta_1, \delta_2\}$. A box on a link or node represents a sensor that indicates when a transmission path intercepts that link/node. We see γ_1 and γ_2 monitor nodes while γ_3, γ_4, and γ_5 monitor links.

a metric, analyze its properties, and give efficient computational approximations in this thesis.

Very often a network is organized for purposes of communication. That is, nodes in the network communicate by transmitting messages along links in the network. This purpose is certainly intrinsic to telephone and computer networks. A command and control structure is a natural example of a social network organized for communication. Here, some central figure of authority might issue orders to one of his closest lieutenants. These orders are then passed along by the subordinate to some appropriate foot soldiers for execution. Consider also a social network formed for the purposes of covertly distributing some product (such as weapons technology or a controlled substance). Suppliers of the product communicate with consumers through some number of middlemen. Similarly, one might be interested in large scale financial transactions initiated by some influential individuals and proceeding through a network of banks and brokers.

In the context of communication networks such as those described above, a useful

bit of information is the source and destination of some transmission that is intercepted along its path through the network. The transmission is intercepted by some sensors in the network–these may be 'boxes' placed on telephone lines, informants in a social network, or cooperative brokers in a financial network (see Figure 1.1). This information would allow one to infer the identities of parties that are significantly altering the nature of their assets, determine central figures in a command structure, or discover the locations of some callers in a telephone network. Due to various legal, technological, and human shortcomings, the content of an intercepted message is very often insufficient for determining its source and destination. We might then turn to knowledge of the transmission's path through the network (as measured by where it was intercepted) to determine the source and destination. This approach is complicated still by the fact that we may not know the true structure of the network of interest. Even worse, our sensors might not be synchronized; in other words we may not know the chronological order in which a single message was intercepted by several sensors. In spite of these deficiencies of knowledge, there is hope for estimating the transmission source and destination if some training data is available. The estimates are further strenghtened by any prior information about the structure of the network and synchronization of sensors that might be available. The present thesis develops a suite of tools to tackle this problem.

1.2 Survey of Topics

Many estimators are defined with respect to some distance norm; consider for example the classic least squares estimator. We will be concerned with inference problems involving networks; the Euclidean norm as is, certainly does not apply to combinatorial structures like networks. Chapter 2 therefore develops a distance

metric for labeled networks known as the graph edit distance. As the name suggests, the graph edit distance is constructed from a set of edit operations that incur some cost. A general formulation for editing graphs is used to derive a graph edit distance that is proven to be a metric provided the cost function for individual edit operations is a metric. Then, a binary linear program is developed for computing this graph edit distance, and polynomial time methods for determining upper and lower bounds on the solution of the binary program are derived by applying solution methods for standard linear programming and the assignment problem. A recognition problem of comparing a sample input graph to a database of known prototype graphs in the context of a chemical information system is presented as an application. The costs associated with various edit operations are chosen by using a minimum normalized variance criterion applied to pairwise distances between nearest neighbors in the database of prototypes. The new metric is shown to perform quite well in comparison to existing metrics when applied to a database of chemical graphs.

After defining a useful distance metric for graphs, we move to the precise statement of the endpoint localization problem in Chapter 3. We consider the problem of estimating the endpoints (source and destination) of a transmission in a network based on partial measurement of the transmission path. Sensors placed at various points within the network provide the basis for endpoint estimation by indicating that a specific transmission has been intercepted at their assigned locations. During a training phase, test transmissions are made between various pairs of endpoints in the network and the sensors they activate are noted. Sensor activations corresponding to transmissions with unknown endpoints are also observed in a monitoring phase. A semidefinite programming relaxation is used in conjunction with the measurements and linear prior information to produce likely sample topologies given

the data. These samples are used to generate Monte Carlo approximations of the posterior distributions of source/destination pairs for measurements obtained in the monitoring phase. The posteriors allow for maximum a posteriori (MAP) estimation of the endpoints along with computation of some resolution measures. We illustrate the method using simulations of random topologies.

The endpoint estimation techniques of Chapter 3 assume a batch mode of operation and are therefore not suitable for online implementation. In Chapter 4, we present online techniques for estimating the source and destination of a suspect transmission through a network based on the activation pattern of sensors placed on network components. We utilize a hierarchical Bayesian model relating routing, tracking, and topological parameters. A controlled Markovian routing model is used in conjunction with a recursive EM algorithm to derive adaptive routing and tracking parameter estimates. Previously developed semidefinite programming methods are used to account for any prior topological information through Monte Carlo estimates of the topology parameters. We prove convergence of the routing and tracking parameter estimates and show that their asymptotic estimates are fixed points of an exact EM algorithm. We present and analyze approximate methods based on permutation clustering to reduce the complexity of sums that arise in the estimator formulas. A multiarmed bandit approach to the design problem of online probe scheduling is also presented. Finally, we illustrate the effectiveness of the new methods through a variety of tracking simulations involving real Internet data. We observe speedy performance and a high degree of accuracy.

Chapter 5 presents a theoretical development of an additional set of endpoint localization methods based on a Markov chain Monte Carlo (MCMC) algorithm. The Metropolis-Hastings algorithm is applied to develop estimators from routing

models used in previous chapters. Since MCMC methods are notoriously slow and computationally difficult, it is unlikely that these methods could compete with the speed of those in prior chapters (especially the online techniques of Chapter 4). However, it remains to be seen whether simulated experiments will reveal improved accuracy. We finally conclude in Chapter 6 with some summarizing remarks and suggestions for future research.

1.3 Contributions

This thesis contains a number of new developments. Although the graph edit distance has been known for some time [94], the edit grid development of Chapter 2 is novel, as is the proof of metric properties and the binary linear programming formulation. The development of the endpoint localization problem using linear priors and ordering distributions as in Chapter 3 is new, along with the adaptation of the semidefinite programming rounding method [31] for incorporating the linear prior. The online estimation methods of Chapter 4 are novel as well, along with the MCMC endpoint localization techniques of Chapter 5. The following publications have resulted from this thesis.

[45] D. Justice and A. Hero. "A binary linear programming formulation of the graph edit distance." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8), Aug. 2006.

[46] D. Justice and A. Hero. "Estimation of message source and destination from network intercepts." *IEEE Transactions on Information Forensics and Security*, 1(3), Sept. 2006.

[47] D. Justice and A. Hero. "Online methods for network endpoint localization." Technical report, Dept. of Electrical Engineering and Computer Science, University

of Michigan, Ann Arbor, MI, July 2006.

We work with the abstract graphical representation of a network throughout the thesis without appealing to specific properties of any certain type of network, and any models are introduced with a minimal number of assumptions. The techniques described are therefore developed at a sufficiently general level so as to make them applicable to a wide variety of problems in network monitoring and surveillance.

CHAPTER II

A Binary Linear Programming Formulation of the Graph Edit Distance

2.1 Introduction

Attributed graphs provide convenient structures for representing objects when relational properties are of interest. Such representations are frequently useful in applications ranging from computer-aided drug design to machine vision. A familiar machine vision problem is to recognize specific objects within an image. In this case, the image is processed to generate a representative graph based on structural characteristics, such a region adjacency graph or a line adjacency graph, and vertex attributes may be assigned according to characteristics of the region to which each vertex corresponds [76]. This representative graph is then compared to a database of prototype or model graphs in order to identify and classify the object of interest. Face identification [42] and symbol recognition [58] are among the problems in machine vision where graphs have been utilized recently. In this context, a reliable and speedy method for comparing graphs is important. Many heuristics and simplifications have been developed and employed for this purpose in a variety of applications [87].

Comparing graphs in the context of graph database searching has also found significant application in the pharmaceutical and agrochemical industries. The attributed graphs of interest are so called chemical graphs which are derived from chemical

structure diagrams. Graphical representations are of great utility here because of the *similar property principle*, which states that molecules with similar structures will exhibit similar chemical properties [44]. Thus databases of these chemical graphs are often searched by comparing with a query graph to aid in the design of new chemicals or medicines [23]. Various techniques have been designed for processing the graphs for structural features and generating bit strings (referred to as *fingerprints*) based on the presence or absence of such features [80]. Since the fingerprints can be rapidly compared, some pre-screening or clustering is often done based on these to eliminate all but the most similar graphs to a given input graph. The remaining graphs may then be compared to the query using a more sophisticated (and more computationally demanding) method. Distance metrics based on the maximum common subgraph are frequently used in this role [104, 81].

Although computing the maximum common subgraph (MCS) is no small task (indeed, it is an NP-Hard problem [29]), several graph distance metrics have been proposed that use the size of the MCS. One such metric is given in [13], with a slight modification presented in [101]. A different MCS-based metric is presented in [43] specifically for application to chemical graphs. An alternate metric that uses the MCS along with the minimum common supergraph has also been proposed [27]. For related structures, such as attributed trees, it is often possible to derive distance metrics based on the maximum common substructure that operate in polynomial time [92]. An intimate relationship between graph comparison and graph (or subgraph) isomorphism is readily apparent in these examples because the MCS defines subgraphs in the two graphs being compared that are isomorphic. Indeed, computing a graph distance metric often requires the computation of some sort of isomorphism (aka matching) between graphs.

An exact graph isomorphism defines a mapping between the nodes of two attributed graphs so that their structures (vertex attributes along with edges) exactly coincide. As one might expect, this is also a challenging computational problem in general although it has not been shown to be NP-Complete [29]. As with subgraph isomorphism, polynomial time algorithms are available for certain restricted classes of graphs [34]. Algorithms for general graph isomorphism that are shown to be quite speedy in practice are given in [61, 21]. Such algorithms typically take advantage of vertex attributes to efficiently prune a search tree constructed for finding an isomorphism. Graphs obtained from real objects are rarely isomorphic, however, so it is useful to consider inexact or error-correcting graph isomorphisms (ECGI) that allow for graphs to nearly (but not exactly) coincide [94]. As the name suggests, the lack of exact isomorphism can be caused by measurement noise or errors in a sample graph when compared to a model graph. On the other hand, when comparing two model graphs one might interpret such 'errors' as capturing the essential differences between the two graphs.

Error-correcting graph matching attempts to compute a mapping between the vertices of two graphs so that they approximately coincide, realizing that the graphs may not be isomorphic. Many suboptimal approaches exist to tackle this problem [2, 96, 32, 97]. The adjacency matrix eigendecomposition approach of [96] gives fast suboptimal results, however it is only applicable to graphs having adjacency matrices with no repeated eigenvalues. Graphs with a low degree of connectivity will often have adjacency matrices with multiple zero eigenvalues. Heuristics are used in the graduated assignment type methods of [32, 97] to significantly reduce the exponential complexity of the original problem. These methods can be applied to very large graphs; however they require several tuning parameters to which the

performance of the algorithm is quite sensitive. Unfortunately, no systematic method for choosing these parameters is provided. The linear programming approach of [2] gives good results in a reasonable amount of time for graphs having the same number of vertices. The authors of [2] use the linear program to minimize a matrix norm similarity metric.

The graph edit distance (GED) is a convenient and logical graph distance metric that arises naturally in the context of error-correcting graph matching [94, 11, 12]. It can also be viewed as an extension of the string edit distance [100]. The basic idea is to define graph edit operations such as insertion or deletion of a node/vertex or relabeling of a vertex along with costs associated with each operation. The graph edit distance between two graphs is then just the cost associated with the least costly series of edit operations needed to make the two graphs isomorphic. The optimal error-correcting graph isomorphism can be defined as the resulting isomorphism after performing this optimal series of edits. Furthermore, it has been shown that the optimal ECGI under a certain graph edit cost function will find the MCS [11]. Enumeration procedures for computing such optimal matchings have been proposed [41, 62, 94]. These procedures are applicable for only small graphs. In [67, 84, 6], probabilistic models of the edit operations are proposed and used to develop MAP estimates of the optimal ECGI. It is not clear in all applications, however, what is the appropriate model to use for the edit probabilities. As with previous metrics, efficient algorithms have been developed for computing edit distances on trees with certain structures [107, 102, 49].

The graph edit distance is parameterized by a set of edit costs. The flexibility provided by these costs can be very useful in the context of a standard recognition problem described earlier of matching a sample input graph to a database of known

prototype graphs [75]. If chosen appropriately, the costs can capture the essential features that characterize differences among the prototype graphs. Recently, methods for choosing these costs that are best from a recognition point of view have been presented. In [69], the EM algorithm is applied to assumed Gaussian mixture models for edit events in order to choose costs that enforce similarity (or dissimilarity) between specific pairs of graphs in a training set. An application for matching images based on their shock graphs [86] uses the tree edit distance algorithm in [49] and chooses edit costs based on local shape differences within shock graphs corresponding to similar images. In a chemical graph recognition application, heuristics are used to choose the edit costs of a string edit distance between strings formed from the maximal paths between vertices in the graphs [36]. Related studies have also been done into the effectiveness of weighting the presence or absence of certain substructures differently when comparing fingerprints derived from chemical graphs [105].

In this chapter, we provide a formulation of the graph edit distance whereby error-correcting graph matching may be performed by solving a binary linear program (BLP–that is a linear program where all variables must take values from the set $\{0, 1\}$). We first present a general framework for computing the GED between attributed, unweighted graphs by treating them as subgraphs of a larger graph referred to as the edit grid. It is argued that the edit grid need only have as many vertices as the sum of the total number of vertices in the graphs being compared. We show that graph editing is equivalent to altering the state of the edit grid and prove that the GED derived in this way is a metric provided the cost function for individual edit operations is a metric. We then use the adjacency matrix representation to formulate a binary linear program to solve for the GED. Since solving a

BLP is NP-Hard [29], we show how to obtain upper and lower bounds for the GED in polynomial time by using solution techniques for standard linear programming and the assignment problem [72]. These bounds may be useful in the event that the problem is so large that solving the BLP is impractical.

We also present a recognition problem [75] that demonstrates the utility of the new method in the context of a chemical information system. Suppose there is a database of prototype chemical graphs to which a sample graph is to be compared as described earlier. The experiment proceeds in two stages: edit cost selection followed by recognition of a perturbed prototype graph via a minimum distance classifier. We provide a method for choosing the edit costs that is purely nonparametric and is based on the assumption (or prior information) that the graphs in the database should be uniformly distributed. The edit costs are chosen as those that minimize the normalized variance of pairwise distances between nearest neighbor prototypes, thereby uniformly distributing them in the metric space of graphs define by the GED. Note that a metric which uniformly distributes nearest neighbors in the database essentially equalizes the probability of classification error with a minimum distance classifier, thereby minimizing the worst case error. This method is similar to the use of spherical packings for error-correcting code design, where the distances between all nearest-neighbor code points are the same [20]. Also, providing such homogeneous sets of graphs is desirable in chemical applications for certain structure-activity experiments [23]. This computation involves matching all pairs of prototypes in a neighborhood and tabulating the edits between them. These are provided as inputs to a single convex program to solve for the optimal edit costs. This is one possible method for choosing edit costs; other methods might certainly be concocted to accommodate whatever prior information about the data at hand is

available.

We test our algorithm on a database of 135 chemical graphs derived from a set of similar biochemical molecules [25]. Our GED metric is compared with the MCS based metrics proposed in [43, 13]. Indeed, the similarity of molecules in this database indicates it is a good candidate for our method of edit cost selection. We first compute the optimal edit costs as previously described and show that our metric equipped with these costs more uniformly distributes the prototype graphs than either MCS metric. The recognition problem is investigated next by generating sample graphs through random perturbations on the prototype graphs; thus we consider a scenario where the ECGI is used to 'fix' errors between sample and model graphs. Each sample graph is matched to every prototype in the database in an effort to recognize which prototype was perturbed to create the sample graph, and a classification ambiguity index is computed. The GED metric is found to perform better with respect to certain types of edit and worse with respect to others than the MCS metrics. However, when random edits are applied, the GED typically performs better.

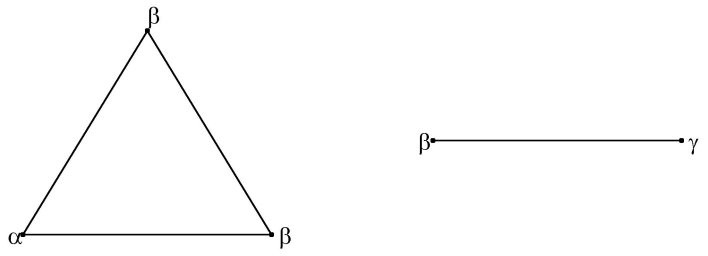
This chapter is organized as follows. Section II presents the bulk of the theory. Within Section II, we first present the general framework for computing the GED and prove that it results in a metric provided the edit cost function is a metric. This is followed by the development of the binary linear program to compute the graph edit distance along with a description of polynomial-time solutions for upper and lower bounds on the GED. Finally a description of edit cost selection for a graph recognition problem is given, and it is shown that the resulting problem is a convex program. Section III presents the results of the graph recognition problem applied to a database of chemical graphs, and Section IV provides some concluding remarks.

2.2 Theory

We first introduce a framework for edits on the set of unweighted, undirected graphs with vertex attributes based on relabeling of graph elements (vertices and edges). Suppose we wish to find the graph edit distance between graphs G_0 and G_1. The graph to be edited G_0 is embedded in a labeled complete graph G_Ω referred to as the 'edit grid.' Vertices and edges in G_Ω may possess the special null label indicating the element is not part of the embedded graph; such an element is referred to as 'virtual' and allows for insertion and deletion edits by simply swapping a null label for a non-null label or vice versa. The state of the edit grid is altered by relabeling its elements until the graph G_1 surfaces somewhere on the grid. Assuming a cost metric on the set of labels (including the null label), we prove the existence of a set of graph edits with minimum cost that occur in one transition of the edit grid state. We also show that the graph edit distance implied by this cost is a metric on the set of graphs.

Next, we consider the adjacency matrix representation in order to develop the binary linear programming formulation of the graph edit optimization as a practical implementation of the general framework. We show how to use this formulation to obtain upper and lower bounds on the graph edit distance in polynomial time.

Finally, we offer a minimum variance method for choosing a cost metric. This metric is appropriate for a graph recognition problem wherein an input graph is compared to a database of prototypes. It is based on the assumption that the prototype graphs should be roughly uniformly distributed in the metric space described by the graph edit distance.



$G_0(V_0, E_0, l_0)$ $G_1(V_1, E_1, l_1)$

Figure 2.1: Example undirected unweighted graphs with vertex attributes. The attribute alphabet is given by $\Sigma = \{\alpha, \beta, \gamma\}$.

2.2.1 Editing Graphs and the Graph Edit Distance

Let $G_0(V_0, E_0, l_0)$ be an undirected graph to be edited where V_0 is a finite set of vertices, $E_0 \subseteq V_0 \times V_0$ is a set of unweighted edges, and $l_0 : V_0 \to \Sigma$ is a labeling function that assigns a label from the alphabet Σ to each vertex. We assume there is at most one edge between any pair of vertices. The vertex labels in Σ capture the attribute information. We define the label $\phi \notin \Sigma$ as ϕ is a special vertex label whose purpose will be introduced shortly. These assumptions are made implicitly for every graph in this chapter so that we need not mention them again. Some example graphs are shown in Figure 2.1.

Let $\Omega = \{\omega_i\}_{i=1}^{N}$ denote a set of vertices; accordingly $\Omega \times \Omega$ is the set of undirected edges connecting all pairs of vertices in Ω. We refer to the complete graph $G_\Omega(\Omega, \Omega \times \Omega, l_\Omega)$ as the *edit grid*. N, the number of vertices in the edit grid, may be as large as necessary. We will argue later that for computing the graph edit distance between $G_0(V_0, E_0, l_0)$ and $G_1(V_1, E_1, l_1)$ N need be no larger than $|V_0| + |V_1|$. An example edit grid with five vertices is shown in Figure 2.2.

For the purposes of editing, we let the graph $G_0(V_0, E_0, l_0)$ be situated on the edit grid, i.e. $V_0 \subset \Omega$ and $E_0 \subset \Omega \times \Omega$; equivalently, G_0 is a subgraph of G_Ω. Vertices in V_0 are assigned the appropriate label from Σ determined by the labeling function l_0, that

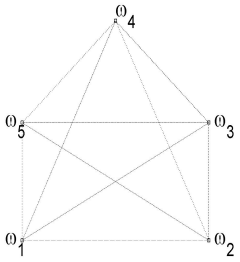


Figure 2.2: Example edit grid $G_\Omega(\Omega, \Omega \times \Omega, l_\Omega)$ with five vertices.

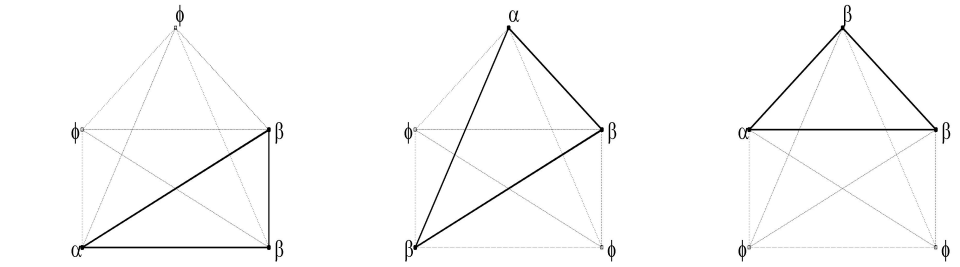


Figure 2.3: Isomorphisms of the graph G_0 on the edit grid G_Ω. Vertex labels are noted; dotted lines indicate virtual edges (label 0) while solid lines indicate real edges (label 1). The vertex numbering in Figure 2.2 is used therefore the standard placement is shown in A.

is $l_\Omega(\omega_i) = l_0(v_i)$ for all $\omega_i \in V_0$. Vertices in $\Omega - V_0$ are assigned the vertex null label ϕ so $l_\Omega(\omega_i) = \phi$ for all $\omega_i \in \Omega - V_0$. The null label indicates a virtual vertex that may be made 'real' during editing by changing its label to something in Σ. Since edges are unweighted, they take labels from the set $\{0, 1\}$ where 1 indicates a real edge and 0 (the edge null label) indicates a virtual edge. Accordingly, $l_\Omega(\omega_i, \omega_j) = 1$ for all edges $(\omega_i, \omega_j) \in E_0$ and $l_\Omega(\omega_i, \omega_j) = 0$ for all edges $(\omega_i, \omega_j) \in (\Omega \times \Omega) - E_0$. When the graph G_0 is placed on the first $|V_0|$ vertices of G_Ω (i.e. $\omega_i = v_i$ for $i = 1, 2, \ldots, |V_0|$), we refer to this as the *standard placement*. Some placements of the graph G_0 from Figure 2.1 on the edit grid of Figure 2.2 are shown in Figure 2.3. These are clearly isomorphisms of G_0 on the edit grid.

Here it is appropriate to provide a quick note on indexing notation used throughout this chapter. Superscript indices index elements within a vector while subscript indices index the entire vector (such as when it occurs in a sequence). For example,

i	ρ^i	η_A^i	η_B^i	η_C^i	π_A^i	π_B^i	π_C^i
1	ω_1	α	β	ϕ	1	4	5
2	ω_2	β	ϕ	ϕ	2	1	4
3	ω_3	β	β	β	3	3	3
4	ω_4	ϕ	α	β	4	5	2
5	ω_5	ϕ	ϕ	α	5	2	1
6	(ω_1, ω_2)	1	0	0	6	8	15
7	(ω_1, ω_3)	1	1	0	7	13	14
8	(ω_1, ω_4)	0	1	0	8	15	12
9	(ω_1, ω_5)	0	0	0	9	11	9
10	(ω_2, ω_3)	1	0	0	10	7	13
11	(ω_2, ω_4)	0	0	0	11	9	11
12	(ω_2, ω_5)	0	0	0	12	6	8
13	(ω_3, ω_4)	0	1	1	13	14	10
14	(ω_3, ω_5)	0	0	1	14	10	7
15	(ω_4, ω_5)	0	0	1	15	12	6

Table 2.1: Element orderings, state vectors, and corresponding state vector permutations for the isomorphisms of G_0 on the edit grid as shown in Figure 2.3. The vector of edit grid graph elements is denoted by ρ, the state vectors are denoted by η_A, η_B, and η_C, and the state vector permutations are denoted by π_A, π_B, and π_C for the corresponding isomorphism in Figure 2.3. Note that the numbering of the edit grid vertices ω_i shown in Figure 2.2 is used.

x_5^2 refers to the second element in the x_5 vector (fifth vector in a sequence of $\{x_k\}$). Similar indexing schemes are adopted for matrices: A_1^{34} refers to the $(3, 4)$ element in matrix A_1. Also, a single superscript index on a matrix indexes the entire row, so that A_1^3 denotes the third row of matrix A_1.

Let $\eta \in (\Sigma \cup \phi)^N \times \{0, 1\}^{\frac{1}{2}(N^2-N)}$ denote the state vector of the edit grid. We assign an ordering to the graph elements (vertices and edges) of the edit grid so that the i^{th} element of η contains the label of the i^{th} element of the edit grid (i.e. $\eta^i = l_\Omega(\rho^i)$ for $\rho^i \in \Omega \cup (\Omega \times \Omega)$). For example, the element orderings and state vectors for the graphs in Figure 2.3 are shown in Table 2.1.

We perform a finite sequence of graph edits to transform the graph $G_0(V_0, E_0, l_0)$ situated on the edit grid $G_\Omega(\Omega, \Omega \times \Omega, l_\Omega)$ into the graph $G_1(V_1, E_1, l_1)$ (such that $V_1 \subset \Omega$ and $E_1 \subset \Omega \times \Omega$). Vertex edits consist of insertion, deletion, or relabeling to some other symbol in Σ. Edge edits consist of insertion or deletion. Using the null labels introduced above, we may interpret all graph edits as relabeling of real and virtual

elements. For example, changing the label of a virtual edge from 0 to 1 corresponds to the insertion of that edge into the graph $G(V, E, l)$. Similarly, relabeling an edge from 1 to 0 amounts to deleting that edge. Vertex insertion or deletion is a bit more complex in that it also typically involves edge edits; however there is a natural decomposition of the vertex edit that is consistent with this framework. Consider a vertex deletion whereby a vertex is removed from the graph along with all edges adjacent to that vertex. We may delete the vertex by changing its label $\sigma \in \Sigma$ to ϕ and relabeling all edges adjacent to it from 1 to 0. Vertex insertion may involve attaching the new vertex to the existing graph via an edge. Again this process is easily decomposed by relabeling a virtual vertex from ϕ to some desired label $\sigma \in \Sigma$ and changing the label of an appropriate virtual edge from 0 to 1. Thus it suffices to consider the transforming of edge and vertex labels as the fundamental operation for editing.

Edits essentially serve to alter the state of the edit grid. Thus we may specify a sequence of edits by noting the sequence of edit grid state vectors $\{\eta_k\}_{k=0}^{M}$ resulting from these edits. Suppose we wish to transform a graph G_0 into a graph G_1 by performing edit operations. Assume at this point that the initial state of the edit grid η_0 contains G_0 in its standard placement. We must have the final state η_M be such that it describes G_1 situated in some fashion on the edit grid. Thus if Γ_1 is the set of state vectors corresponding to all isomorphisms of G_1 on the edit grid, we must have $\eta_M \in \Gamma_1$. Two different state sequences for transforming the example G_0 into the example G_1 of Figure 2.1 are shown in Figure 2.4.

The set of all isomorphisms of a graph G_n on the edit grid, Γ_n, may be defined in terms of the standard placement of G_n denoted by η_n and Π–the set of all permutation

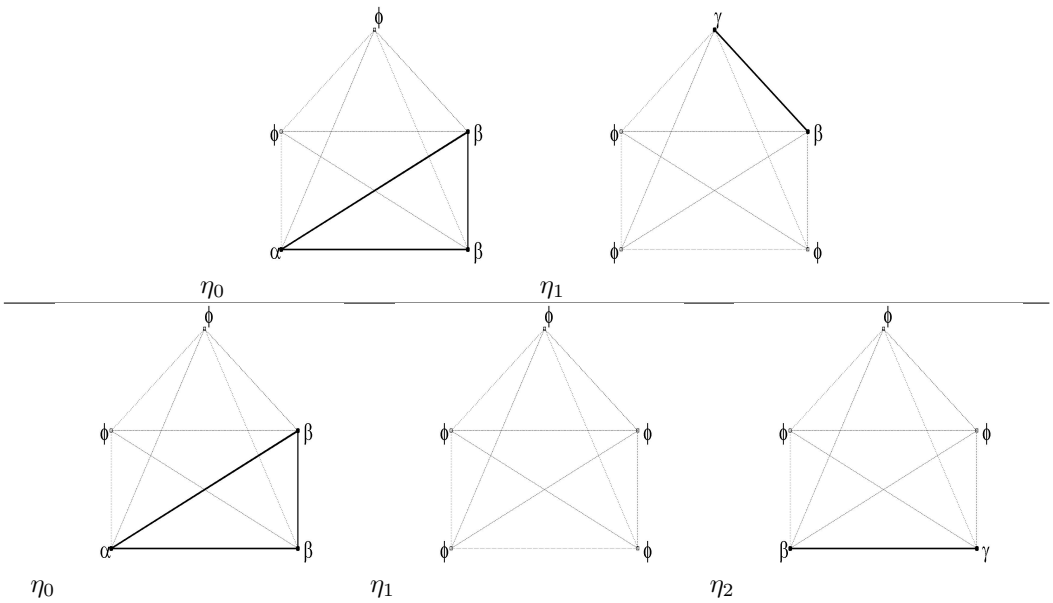


Figure 2.4: Two different edit grid state sequences for transforming G_0 of Figure 2.1 into G_1. The upper sequence requires only one state transition, while the lower sequence requires two. In both cases, the initial state is the standard placement of G_0, and the final state represents an isomorphism of G_1 on the edit grid. Using the vertex numbering scheme of Fig. 2.2, the following non-trivial edits are made in the single transition of the upper sequence: $(\omega_1, \alpha \to \phi)$, $(\omega_2, \beta \to \phi)$, $(\omega_4, \phi \to \gamma)$, $((\omega_1, \omega_2), 1 \to 0)$, $((\omega_1, \omega_3), 1 \to 0)$, $((\omega_2, \omega_3), 1 \to 0)$, and $((\omega_3, \omega_4), 0 \to 1)$. In the first transition of the lower sequence we have $(\omega_1, \alpha \to \phi)$, $(\omega_2, \beta \to \phi)$, $(\omega_3, \beta \to \phi)$, $((\omega_1, \omega_2), 1 \to 0)$, $((\omega_1, \omega_3), 1 \to 0)$, $((\omega_2, \omega_3), 1 \to 0)$, and in the second transition of the lower sequence: $(\omega_1, \phi \to \beta)$, $(\omega_2, \phi \to \gamma)$, $((\omega_1, \omega_2), 0 \to 1)$. For a metric cost function c, the cost of the upper sequence is given by $c(\alpha, \phi) + c(\beta, \phi) + c(\phi, \gamma) + 4c(0, 1)$, and the cost of the lower sequence is $c(\alpha, \phi) + 3c(\beta, \phi) + c(\phi, \gamma) + 4c(0, 1)$.

mappings describing isomorphisms of the edit grid G_Ω–as in Eq. (2.1).

$$\Gamma_n = \left\{ \eta \mid \exists \pi \in \Pi \text{ s.t. } \eta^i = \eta_n^{\pi^i} \right\} \tag{2.1}$$

Note that Π does not contain all possible permutations of the elements of the state vector η because elements of Π must describe an isomorphism of the edit grid. For example, an edit grid with two vertices $\Omega = \{\omega_1, \omega_2\}$ has only two isomorphisms: $\omega_1' = \omega_1$, $\omega_2' = \omega_2$ and $\omega_1' = \omega_2$, $\omega_2' = \omega_1$. Assuming the graph elements are indexed as $\rho = (\omega_1, \omega_2, (\omega_1, \omega_2))$, there are only two permutations of the state vector that comprise Π: $\pi_1 = (1, 2, 3)$ and $\pi_2 = (2, 1, 3)$. Indeed, it will always be the case that $|\Pi| = N!$. The permutations of the state vector corresponding to the isomorphisms of G_0 in Figure 2.3 are given in Table 2.1.

We define a cost function $c : (\Sigma \cup \phi)^2 \cup \{0,1\}^2 \rightarrow \Re_+$ that assigns a nonnegative cost to each graph edit. The cost of an edit grid state transition denoted as $C(\eta_{k-1}, \eta_k)$ is simply the sum of all edits separating the two states, i.e.

$$C(\eta_{k-1}, \eta_k) = \sum_{i=1}^{I} c(\eta_{k-1}^i, \eta_k^i) \tag{2.2}$$

where $I = N + \frac{1}{2}(N^2 - N)$ is the total number of graph elements (vertices and edges) in the edit grid. Similarly, the cost of a sequence of state transitions is simply the sum of the costs of individual transitions. We consider only cost functions that are metrics on the set of vertex and edge labels as characterized by Definition II.1.

Definition II.1. A cost function $c : (\Sigma \cup \phi)^2 \cup \{0,1\}^2 \rightarrow \Re_+$ is a metric if the following conditions hold for all $(x, y) \in (\Sigma \cup \phi)^2 \cup \{0,1\}^2$:

1. Positive definiteness: $c(x, y) = 0$ if and only if $x = y$.
2. Symmetry: $c(x, y) = c(y, x)$.
3. Triangle inequality: $c(x, y) \leq c(x, z) + c(z, y)$ for all (x, z) and (z, y) in $(\Sigma \cup \phi)^2 \cup \{0,1\}^2$.

Assuming a metric cost function, the cost of the upper sequence in Figure 2.4 is $c(\alpha, \phi) + c(\beta, \phi) + c(\phi, \gamma) + 4c(0,1)$, and the cost of the lower sequence is $c(\alpha, \phi) + 3c(\beta, \phi) + c(\phi, \gamma) + 4c(0,1)$. With such a cost function c, we have the following simple result.

Proposition II.2. *If $c : (\Sigma \cup \phi)^2 \cup \{0,1\}^2 \rightarrow \Re_+$ is a metric on the set of labels then C as defined in Eq. (2.2) is a metric on the edit grid state space.*

Proof. Expand C in terms of c using the definition in Eq. (2.2) and apply the metric properties of c to trivially obtain the corresponding properties for C. □

We thus have the following useful Lemma:

Lemma II.3. *Let c be a metric and $\{\eta_k\}_{k=0}^M$ be a sequence of edit grid state vectors. Then for all $M \geq 1$, $C(\eta_0, \eta_M) \leq \sum_{k=1}^M C(\eta_{k-1}, \eta_k)$.*

Proof. The $M = 1$ case is trivial and the $M = 2$ case follows from the triangle inequality. Assume the claim holds for some value M and proceed by induction:

$$\begin{aligned} C(\eta_0, \eta_{M+1}) &\leq C(\eta_0, \eta_M) + C(\eta_M, \eta_{M+1}) \\ &\leq \textstyle\sum_{k=1}^M C(\eta_{k-1}, \eta_k) + C(\eta_M, \eta_{M+1}) \\ &= \textstyle\sum_{k=1}^{M+1} C(\eta_{k-1}, \eta_k) \end{aligned} \tag{2.3}$$

where the first line in Eq. (2.3) follows from the triangle inequality and the second line uses the induction hypothesis. □

We now define the graph edit distance with respect to a cost function c as

$$d_c(G_0, G_1) = \min_{\{\eta_k\}_{k=1}^M | \eta_M \in \Gamma_1} \sum_{k=1}^M C(\eta_{k-1}, \eta_k) \tag{2.4}$$

where η_0 is the standard placement of G_0 on the edit grid, Γ_1 is the set of state vectors corresponding to isomorphisms of G_1 on the edit grid as in Eq. (2.1), and M is the maximum number of allowed state transitions (this can be as large as desired, but we are only concerned with finite graphs implying M will be finite). There is no loss of generality by fixing the number of terms in the summation of Eq. (2.4), since for $M' < M$ transitions we can simply repeat the final state $\eta_{M'}$ so that $\eta_k = \eta_{M'}$ for $k = M' + 1, M' + 2, \ldots M$. Note that since there is a finite number of state vector sequences $\{\eta_k\}_{k=1}^M$ such that $\eta_M \in \Gamma_1$, the graph edit distance as defined in Eq. (2.4) always exists. It essentially finds a state transition sequence of minimum cost that transforms G_0 into G_1 (the minimizing sequence need not be unique). Since our cost function is a metric, it seems logical that we should be able to achieve the minimum

cost with only one edit grid state transition. The following theorem shows this is indeed the case.

Theorem II.4. *For a given graph edit cost function,* $c : (\Sigma \cup \phi)^2 \cup \{0,1\}^2 \to \Re_+$ *that is a metric, there exists a single state transition* $(\eta_0, \bar{\eta}_1)$ *such that* $d_c(G_0, G_1) = C(\eta_0, \bar{\eta}_1)$ *where* η_0 *is the standard placement of* G_0 *and* $\bar{\eta}_1 \in \Gamma_1$.

Proof. Assume the initial state η_0 describes G_0 in its standard placement on the edit grid, and suppose $\{\tilde{\eta}_k\}_{k=1}^M$ solves the graph edit minimization in Eq. (2.4)–this optimal sequence always exists as argued earlier. Define $\bar{\eta}_1 = \tilde{\eta}_M$, then we have

$$\begin{aligned} d_c(G_0, G_1) &= C(\eta_0, \tilde{\eta}_1) + \textstyle\sum_{k=2}^M C(\tilde{\eta}_{k-1}, \tilde{\eta}_k) \\ &\geq C(\eta_0, \tilde{\eta}_M) \\ &= C(\eta_0, \bar{\eta}_1) \end{aligned} \tag{2.5}$$

where the second line follows from the first by applying Lemma II.3. Now define the state sequence $\{\hat{\eta}_k\}_{k=1}^M$ so that $\hat{\eta}_k = \bar{\eta}_1$ for all k. Then the positive definiteness of the metric C gives

$$\begin{aligned} C(\eta_0, \bar{\eta}_1) &= C(\eta_0, \hat{\eta}_1) + \textstyle\sum_{k=2}^M C(\hat{\eta}_{k-1}, \hat{\eta}_k) \\ &\geq \min_{\{\eta_k\}_{k=1}^M | \eta_M \in \Gamma_1} \textstyle\sum_{k=1}^M C(\eta_{k-1}, \eta_k) \\ &= d_c(G_0, G_1) \end{aligned} \tag{2.6}$$

The second line follows because $\hat{\eta}_M = \bar{\eta}_1 = \tilde{\eta}_M \in \Gamma_1$ and the proof is complete. □

Theorem II.4 allows the graph edit distance in Eq. (2.4) to be re-expressed equivalently as

$$d_c(G_0, G_1) = \min_{\bar{\eta}_1 \in \Gamma_1} C(\eta_0, \tilde{\eta}_1) = \min_{\pi \in \Pi} \sum_{i=1}^{I} c(\eta_0^i, \eta_1^{\pi^i}) \tag{2.7}$$

where the second equality follows from the definition of Γ_1 in Eq. (2.1) and η_0 and η_1 are the standard placements of G_0 and G_1 respectively. Note that this one-state-transition result is crucial for our binary linear programming formulation, and it

hinges on the metric properties of the cost function. Under a cost that is not a metric, there is no guarantee that the graph edit distance can be computed with just one edit grid state transition. One might consider multiple state transitions in a greedy algorithm for computing the graph edit distance in this case.

If π solves the minimization in Eq. (2.7), then we have

$$\begin{aligned} d_c(G_0, G_1) &= \sum_{i=1}^{I} c(\eta_0^i, \eta_1^{\pi^i}) \\ &= \sum_{i|\rho^i \in V_0 \cup V_1 \cup E_0 \cup E_1} c(\eta_0^i, \eta_1^{\pi^i}) \end{aligned} \tag{2.8}$$

Thus only elements of the edit grid comprising either G_0 or G_1 contribute to the sum; all other elements have the null label in both states and therefore have zero cost. The most terms contribute to the summation in Eq. (2.8) in the case where V_0 and V_1 are disjoint. This suggests we need an edit grid with no more than $N = |V_0| + |V_1|$ vertices in order to compute the graph edit distance in Eq. (2.7).

2.2.2 A Metric for Graphs

In addition to justifying the binary linear programming formulation to follow, Theorem II.4 also provides a simple means for showing that the graph edit distance (when derived from a metric cost) is a metric itself on the set of undirected, unweighted graphs with vertex attributes (denoted by Ξ). We need a preliminary lemma however. We have assumed for simplicity that the graph edit minimization always starts with the graph G_0 in its standard placement on the edit grid. It seems that this should not be necessary; i.e. that the graph edit distance should be the same regardless of where G_0 is on the edit grid. The following lemma establishes this.

Lemma II.5. *If $\bar{\eta}_0 \in \Gamma_0$ where Γ_0 is as defined in Eq. (2.1), then $d_c(G_0, G_1) = \min_{\tilde{\eta}_1 \in \Gamma_1} C(\bar{\eta}_0, \tilde{\eta}_1) = \min_{\pi \in \Pi} \sum_{i=1}^{I} c(\bar{\eta}_0^i, \eta_1^{\pi^i})$.*

Proof. Let $\bar{\eta}_0^i = \eta_0^{\bar{\pi}^i}$ for the standard placement η_0 then we have

$$\begin{aligned} d_c(G_0, G_1) &= \min_{\pi \in \Pi} \sum_{i=1}^{I} c(\eta_0^i, \eta_1^{\pi^i}) \\ &= \min_{\pi \in \Pi} \sum_{j=1}^{I} c(\eta_0^{\bar{\pi}^j}, \eta_1^{\pi^{\bar{\pi}^j}}) \\ &= \min_{\tilde{\pi} \in \Pi} \sum_{j=1}^{I} c(\bar{\eta}_0^j, \eta_1^{\tilde{\pi}^j}) \end{aligned} \tag{2.9}$$

where the second line follows by reordering the sum with index change $i = \bar{\pi}^j$, and the third by noting that applying $\bar{\pi}$ to any permutation π in Π results in another permutation $\tilde{\pi}$ in Π. □

We now prove that the graph edit distance is a metric.

Theorem II.6. *If the cost function $c : (\Sigma \cup \phi)^2 \cup \{0,1\}^2 \to \Re_+$ is a metric, then the associated graph edit distance $d_c : \Xi^2 \to \Re_+$ is a metric.*

Proof. Let G_0, G_1, and G_2 all be graphs in Ξ.

1. Positive definiteness: Apply Theorem II.4 to give $d_c(G_0, G_1) = C(\eta_0, \bar{\eta}_1)$. Clearly d_c is nonnegative because it is a sum of nonnegative edit costs. Now since C is a metric, $C(\eta_0, \bar{\eta}_1) = 0$ if and only if $\eta_0 = \bar{\eta}_1$. This occurs if and only if G_0 is isomorphic to G_1. In other words the standard placement of G_0, η_0, also describes an isomorphism of G_1 on the edit grid, i.e. $\eta_0 \in \Gamma_1$.

2. Symmetry: Theorem II.4 gives

$$\begin{aligned} d_c(G_0, G_1) &= C(\eta_0, \bar{\eta}_1) \\ &= C(\bar{\eta}_1, \eta_0) \\ &\geq \min_{\tilde{\eta}_0 \in \Gamma_0} C(\bar{\eta}_1, \tilde{\eta}_0) \\ &= d_c(G_1, G_0) \end{aligned} \tag{2.10}$$

   where symmetry of the metric C gives the second line, and Lemma II.5 gives the fourth line from the third. The reverse inequality, $d_c(G_1, G_0) \geq d_c(G_0, G_1)$, is established via an identical argument.

3. Triangle Inequality: The symmetry property gives $d_c(G_0, G_2) = d_c(G_2, G_0)$. Theorem II.4 gives $d_c(G_2, G_0) = C(\eta_2, \bar{\eta}_0)$ and $d_c(G_2, G_1) = C(\eta_2, \bar{\eta}_1)$ where η_2 is the standard placement of G_2. But by symmetry of C we have $C(\eta_2, \bar{\eta}_0) = C(\bar{\eta}_0, \eta_2)$ therefore

$$\begin{aligned} d_c(G_0, G_2) + d_c(G_2, G_1) &= C(\bar{\eta}_0, \eta_2) + C(\eta_2, \bar{\eta}_1) \\ &\geq C(\bar{\eta}_0, \bar{\eta}_1) \\ &\geq \min_{\tilde{\eta}_1 \in \Gamma_1} C(\bar{\eta}_0, \tilde{\eta}_1) \\ &= d_c(G_0, G_1) \end{aligned} \tag{2.11}$$

where the second line follows from the triangle inequality for C, and Lemma II.5 gives the fourth line from the third.

□

2.2.3 Binary Linear Program for the Graph Edit Distance

In order to develop a binary linear program for computing the graph edit distance, we organize the labels in the edit grid state vector using the adjacency matrix representation. The elements of the adjacency matrix consist of edge labels, and we associate a vertex label with each row(column) of the matrix (the matrix is symmetric since the graphs of interest are undirected). We adopt the ordering scheme in Table 2.1, so that if $A_k \in \{0,1\}^{N \times N}$ is the adjacency matrix corresponding to edit grid state vector η_k then the vertex label η_k^i is associated with the i^{th} row(column) of A_k for $i = 1, 2, \ldots, N$ (i.e. $l(A_k^i) = \eta_k^i$), and the upper half of the matrix is given by $A_k^{ij} = \eta_k^{iN+j-\frac{i^2+i}{2}}$ for $i < j \leq N$–the lower half follows similarly from symmetry. All zeros lie on the diagonal of A_k. For example, the adjacency matrix representations

of the isomorphisms in Figure 2.3 (with state vectors in Table (2.1) are given by

$$A_A = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} \alpha \\ \beta \\ \beta \\ \phi \\ \phi \end{matrix} \quad A_B = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{matrix} \beta \\ \phi \\ \beta \\ \alpha \\ \phi \end{matrix}$$

(2.12)

$$A_C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \begin{matrix} \phi \\ \phi \\ \beta \\ \beta \\ \alpha \end{matrix}$$

where the vertex labels associated with each row/column are printed to the right of the corresponding row.

The adjacency matrix representation also allows a convenient means for expressing an isomorphism permutation $\pi \in \Pi$. We can represent the first N elements of π as a permutation matrix $P \in \{0,1\}^{N \times N}$–recall that the remaining $\frac{1}{2}(N^2 - N)$ elements of π correspond to reordering of the edges which is determined by the vertex permutation in the first N elements. The elements of the permutation matrix P corresponding to π are given by $P^{ij} = \delta(\pi^i, j)$ for $i, j = 1, 2, \ldots, N$ where $\delta : \Re^2 \rightarrow \{0, 1\}$ is the Kronecker delta function.

We now write the graph edit distance in this framework. First, partition the re-expression of Eq. (2.7) into summations over state vector entries corresponding to

vertex and edge elements (assuming an ordering like the one in Table 2.1 is used).

(2.13)
$$\begin{aligned} d_c(G_0, G_1) &= \min_{\pi \in \Pi} \sum_{i=1}^{N} c(\eta_0^i, \eta_1^{\pi^i}) + \sum_{i=N+1}^{I} c(\eta_0^i, \eta_1^{\pi^i}) \\ &= \min_{\pi \in \Pi} \sum_{i=1}^{N} \sum_{j=1}^{N} c(\eta_0^i, \eta_1^j)\delta(\pi^i, j) + c(0,1) \sum_{i=N+1}^{I} (1 - \delta(\eta_0^i, \eta_1^{\pi^i})) \end{aligned}$$

where the second term in the second line follows because c is a metric.

Let A_n be the adjacency matrix corresponding to η_n (the standard placement vector of G_n), $l(A_n^i)$ be the label assigned to the i^{th} row/column of A_n as previously described, and B be the set of all permutation matrices on $\Re^{N \times N}$ given by

$$B = \left\{ X \in \{0,1\}^{N \times N} \mid \sum_j X^{kj} = \sum_i X^{ik} = 1 \ \forall k \right\} \tag{2.14}$$

Eq. (2.13) may then be rewritten as

$$d_c(G_0, G_1) = \min_{P \in B} \sum_{i=1}^{N} \sum_{j=1}^{N} c\left(l(A_0^i), l(A_1^j)\right) P^{ij} + \frac{1}{2} c(0,1) \left| A_0 - P A_1 P^T \right|^{ij} \tag{2.15}$$

Note that since A_n corresponds to the standard placement, only the first $|V_n|$ rows (columns) will have non-ϕ labels, and only the upper $|V_n| \times |V_n|$ block of A_n will have nonzero elements. Also, following the prior argument, we use $N = |V_0| + |V_1|$. In order to make the optimization in Eq. (2.15) linear, we follow the strategy in [2] by introducing the matrices $\tilde{S}$, $\tilde{T}$. The graph edit distance $d_c(G_0, G_1)$ is then the optimal value of the following problem.

$$\begin{aligned} \min_{P, \tilde{S}, \tilde{T} \in \{0,1\}^{N \times N}} \quad & \sum_{i=1}^{N} \sum_{j=1}^{N} c\left(l(A_0^i), l(A_1^j)\right) P^{ij} + \tfrac{1}{2} c(0,1) \left(\tilde{S} P + \tilde{T} P \right)^{ij} \\ \text{such that} \quad & (A_0 - P A_1 P^T + \tilde{S} - \tilde{T})^{ij} = 0 \ \forall i, j \\ & \textstyle\sum_i P^{ik} = \sum_j P^{kj} = 1 \ \forall k \end{aligned} \tag{2.16}$$

where we introduce an extra P in the second term of the objective function, which does not affect the result because it simply reorders the terms in the sum. Finally, we make the change of variables $S = \tilde{S} P$, $T = \tilde{T} P$ and right multiply the constraint

equation by P to obtain the following binary linear program (BLP) for $d_c(G_0, G_1)$.

$$\begin{aligned} \min_{P,S,T \in \{0,1\}^{N \times N}} \quad & \sum_{i=1}^{N} \sum_{j=1}^{N} c\left(l(A_0^i), l(A_1^j)\right) P^{ij} + \tfrac{1}{2} c(0,1) \left(S+T\right)^{ij} \\ \text{such that} \quad & \left(A_0 P - P A_1 + S - T\right)^{ij} = 0 \ \forall i,j \\ & \textstyle\sum_i P^{ik} = \sum_j P^{kj} = 1 \ \forall k \end{aligned} \tag{2.17}$$

Note that Eq. (2.17) an equivalent representation of the GED minimization, so that Theorem II.4 assures the existence of an optimal solution. More explicitly, feasibility of this BLP is seen by taking P as the identity matrix, S as the nonnegative part of $A_1 - A_0$, and T as the nonnegative part of $A_0 - A_1$. One might compare Eq. (2.17) to the linear programming approach for graph matching in [2]. Although [2] seeks to minimize the difference in adjacency matrix norms for graphs with the same number of vertices, this is a sort of generalization for the graph edit distance on attributed graphs. The optimal permutation matrix that solves Eq. (2.17), P_*, can be used to determine the optimal edit operations whose cost is the graph edit distance as follows: simply form the permuted adjacency matrix $\bar{A}_1 = P_* A_1 P_*^T$ remembering to also permute row/column labels, then compare the row/column labels of A_0 to those of $\bar{A}_1$ to determine the optimal vertex relabelings, similarly compare elements of A_0 and $\bar{A}_1$ to determine optimal edge relabelings.

2.2.4 Bounding the Graph Edit Distance in Polynomial Time

Unfortunately, binary linear programming in general is NP-Hard [72], so for large problems the graph edit distance as given by Eq. (2.17) may be too hard to compute. However, we can obtain upper $u_{d_c}(G_0, G_1)$ and lower $l_{d_c}(G_0, G_1)$ bounds for the graph edit distance $d_c(G_0, G_1)$ in polynomial time. The lower bound is obtained by relaxing the constraints $P, S, T \in \{0,1\}^{N \times N}$ on the variables in Eq. (2.17) to $P, S, T \in [0,1]^{N \times N}$. This results in the linear programming relaxation given in Eq.

(2.18).

$$
\begin{aligned}
\min_{P,S,T} \quad & \textstyle\sum_{i=1}^{N}\sum_{j=1}^{N} c\left(l(A_0^i), l(A_1^j)\right) P^{ij} + \frac{1}{2}c(0,1)\,(S+T)^{ij} \\
\text{s.t.} \quad & (A_0P - PA_1 + S - T)^{ij} = 0 \;\forall i,j \\
& \textstyle\sum_i P^{ik} = \sum_j P^{kj} = 1 \;\forall k \\
& 0 \le P^{ij} \le 1, \;\; 0 \le S^{ij} \le 1, \;\; 0 \le T^{ij} \le 1 \;\forall i,j
\end{aligned}
\tag{2.18}
$$

For n variables, a linear program can be solved in $O(n^{3.5})$ time using an interior point method [85]. Thus the lower bound can be computed in $O(N^7)$ time since the linear program in Eq. (2.18) has $O(N^2)$ variables. If $l_{d_c}(G_0, G_1)$ is the optimal value of Eq. (2.18) then $l_{d_c}(G_0, G_1) \le d_c(G_0, G_1)$ because the feasible region of the problem in Eq. (2.17) is a subset of the feasible region of the problem in Eq. (2.18). It follows that Eq. (2.18) is always feasible because Eq. (2.17) is always feasible as argued earlier. Thus the Weierstrass theorem assures existence of an optimal value since we are minimizing a linear functional over a nonempty compact set [60]. Notice that since the optimal matrix P_* that solves Eq. (2.18) is only guaranteed to be doubly stochastic, not necessarily a permutation matrix, there may not be a set of edit operations that achieves the lower bound $l_{d_c}(G_0, G_1)$. However in the event that P_* is a permutation matrix, such a set can be constructed as described in the previous section, and the optimal value of Eq. (2.18) is in fact the graph edit distance.

The upper bound is obtained in polynomial time by solving the assignment problem with only the vertex edit term. The assignment problem is given by

$$
\begin{aligned}
\min_{P \in \{0,1\}^{N\times N}} \quad & \textstyle\sum_{i=1}^{N}\sum_{j=1}^{N} c\left(l(A_0^i), l(A_1^j)\right) P^{ij} \\
\text{such that} \quad & \textstyle\sum_i P^{ik} = \sum_j P^{kj} = 1 \;\forall k
\end{aligned}
\tag{2.19}
$$

Note that the optimal value of Eq. (2.19) always exists since there is a finite number of permutations and the identity always serves as a feasible permutation. Indeed, the

Hungarian method may be used to solve it in $O(N^3)$ time [72]. If P_* is an optimal solution of Eq. (2.19), we compute S_* as the nonnegative part of $P_*A_1 - A_0P_*$ and T_* as the nonnegative part of $A_0P_* - P_*A_1$ so that (P_*, S_*, T_*) is in the feasible region of the problem in Eq. (2.17). $u_{d_c}(G_0, G_1)$ is then computed by evaluating the objective function in Eq. (2.17) at (P_*, S_*, T_*). It follows that $d_c(G_0, G_1) \leq u_{d_c}(G_0, G_1)$. Since P_* is a permutation matrix for the solution to Eq. (2.19), a set of edit operations whose cost is the upper bound to the graph edit distance can always be determined.

2.2.5 Selecting a Cost Metric for Uniform Distribution

We have assumed that the cost metric that characterizes the graph edit distance is available, however, in a given application it may not be clear what is the 'best' cost metric to use. We propose an empirical method for selecting a metric based on prior information suitable for a recognition problem. Suppose there is a set of prototype graphs $\{G_i\}_{i=1}^N$, and we classify a sample graph G_0 by selecting the prototype that is closest to it with respect to a graph distance metric. Prior information might suggest that the prototypes should be roughly uniformly distributed in the metric space of graphs defined by the graph edit distance. We can then choose an optimal metric with respect to this objective. Such a criterion will also have the effect of minimizing the worst case classification error, since it equalizes the probability of error under the minimum distance classifier.

Note that for a set of points uniformly distributed in some space, all nearest neighbor distances are the same. To uniformly distribute the prototypes, we first determine all pairwise distances using a cost metric that assigns unity to all edits, i.e. $c(0,1) = 1$, $c\left(l(A_0^i), l(A_1^j)\right) = 0$ if $l(A_0^i) = l(A_1^j)$ and $c\left(l(A_0^i), l(A_1^j)\right) = 1$ otherwise. The resulting edit operations under the unit cost matching are then fixed, and we optimize the normalized variance of pairwise distances over the set of cost metrics.

To carry out this optimization, we must tabulate the edits necessary to match each graph with its q nearest neighbors under a unit cost function. If the graphs are not too large, we may compute a permutation matrix that actually solves the graph edit distance minimization in Eq. (2.17). If this is not the case a permutation matrix that solves the assignment problem in Eq. (2.19) with unit weights will serve as a reasonable approximation. We consider each nearest neighbor pair only once. For example if G_i has G_j as one of its q nearest neighbors and G_j has G_i as one of its q nearest neighbors, then the edits necessary to match G_i to G_j are tabulated only once.

After determining the unit cost matching between all prototype pairs, we order all distinct edits that occur in any prototype matching and tabulate the vectors $\{H_j\}_{j=1}^K$. H_j^i indicates the number of times the i^{th} edit occurs to match the j^{th} pair of nearest neighbor prototypes (under a unit cost function) and K is the number of distinct nearest neighbor pairs. If c is a vector containing the corresponding edit costs, then the graph edit distance between the j^{th} pair is given by $d_c(G_{j0}, G_{j1}) = H_j^T c$. For example, consider as prototypes the standard placements of G_0 and G_1 as shown in the lower left and lower right respectively of Figure 2.4. If we order the edits as $(\{\alpha,\beta\},\{\alpha,\gamma\},\{\alpha,\phi\},\{\beta,\gamma\},\{\beta,\phi\},\{\gamma,\phi\},\{1,0\})$ then the vector of counts H corresponding to this matching is $(1,0,0,1,1,0,2)$. Indeed, the dimension of each H_j vector will always be $\frac{1}{2}(|\Sigma|^2+|\Sigma|)+1$ for a vertex label set Σ and edge label set $\{0,1\}$.

Using this notation, the scaled variance of pairwise distances is then given by

$$K\sigma_d^2 = c^T \left[\sum_{i=1}^{K}\left(H_i - \frac{1}{K}\sum_{j=1}^{K} H_j\right)\left(H_i - \frac{1}{K}\sum_{j=1}^{K} H_j\right)^T\right] c \equiv c^T Q c \tag{2.20}$$

We also require the cost function to be a metric. Positive definiteness may be

enforced by selecting some minimum positive cost for all edits $a > 0$. Symmetry is enforced implicitly by binning symmetric edits together in the count vector H and assigning the same cost to both edits. Finally, we must include linear inequalities of the form $c^i + c^j - c^k \geq 0$ to assure the triangle inequality holds for all sets of three vertex labels. There are $\frac{1}{2}|\Sigma|(|\Sigma|^2 - 1)$ of these; we define the matrix F such that $Fc \geq 0$ summarizes the triangle inequalities. The variance should be normalized before optimizing so that the result does not depend on the scale of the costs (determined by a). If we normalize by the sum of the costs, a convex program results where any local optimum is also a global optimum [10]. The optimal costs are then given by the convex program:

$$\begin{aligned} \min_{c} \quad & \frac{c^T Q c}{e^T c} \\ \text{s.t.} \quad & Fc \geq 0 \\ & c^i \geq a \ \ \forall i \end{aligned} \tag{2.21}$$

where e is a vector of ones. Note that the problem is always feasible because if we take $c^i = a$ for all i, then all necessary triangle inequalities are satisfied. Furthermore, the choice of a is irrelevant, provided $a > 0$; we only need that the costs c^i (all of which correspond to nontrivial edits) be uniformly bounded below away from zero so that a metric results. For any given a we may choose $a' = \kappa a$ for some $\kappa > 0$, and the change of variables $c' = \kappa c$ results in the original optimization problem in Eq. (2.21). The following proposition establishes the convexity of the problem. A barrier method will solve the convex program in polynomial time [10].

Proposition II.7. *The optimization problem in Eq. (2.21) is convex.*

Proof. All inequalities are linear, so we need only show that the objective function is convex. The objective function is defined over the positive orthant, which is a convex set, so the function is convex if and only if its Hessian is positive semidefinite [10].

Let $\Psi(c) = \frac{c^T Q c}{e^T c}$ be the objective function of the problem. The Hessian quadratic form with an arbitrary vector v may be factored as

(2.22)

$$\begin{aligned} v^T(\nabla^2\Psi(c))v &= v^T \left(\tfrac{2}{(e^T c)^3} \left[(c^T Q c) e e^T + (e^T c)^2 Q - (e^T c)(e c^T Q + Q c e^T) \right] \right) v \\ &= \tfrac{2}{(e^T c)^3} \left\| (e^T v) Q^{\frac{1}{2}} c - (e^T c) Q^{\frac{1}{2}} v \right\|^2 \geq 0 \end{aligned}$$

where $Q^{\frac{1}{2}}$ is the matrix square root of Q which exists because Q as defined in Eq. (2.20) is obviously symmetric positive semidefinite. The inequality follows because c is defined over the positive orthant so $(e^T c)^3 > 0$; thus $\nabla^2\Psi(c) \succeq 0$. □

2.3 Chemical Graph Recognition

As an application, we use the graph edit distance to recognize chemical graphs in the context of a chemical information system. We selected our database of 135 similar molecules from the Klotho Biochemical Compounds Declarative Database, which consists of small molecules useful in describing mechanisms of biochemical reactions [25]. Only molecules with 18 or fewer atoms were used so that we could compute exact distance measures in reasonable time. Attributed undirected graphs were generated from the 135 molecules (referred to as *chemical graphs*), then the optimal edit costs were computed to uniformly distribute them in the graph metric space. Finally, we compared the recognition ability of the graph edit distance with optimal costs and unit costs to that of two maximum common subgraph based distance metrics by using randomly perturbed prototype graphs from the database.

We first generated chemical graphs from the molecular structure diagrams by associating atoms with vertices and bonds with edges. Each vertex was labeled by the chemical symbol of the element to which that vertex corresponded. Our vertex label alphabet was thus given by $\Sigma = \{H, C, O, N, Cl, P, S, Br, Si\}$. Vertices

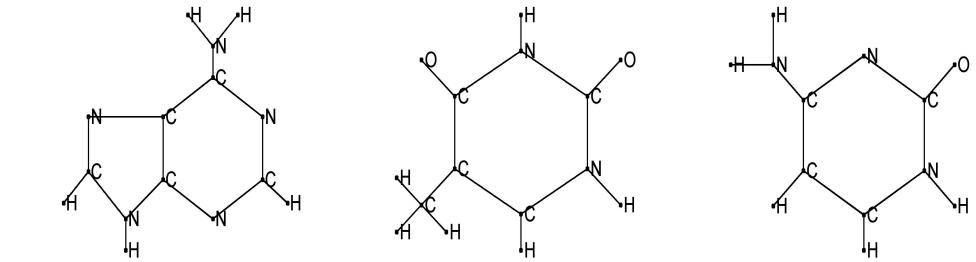


Figure 2.5: Chemical graphs derived from the familiar molecules from DNA: adenine (A), thymine (B), and cytosine (C).

in the graph were connected by an edge if and only if their corresponding atoms were bonded (single bonds, double bonds, etc. were treated equally). For example, the chemical graphs derived from the molecules adenine, thymine, and cytosine are shown in Figure 2.5.

In order to compute the optimal edit costs, we treated all 135 molecules as nearest neighbors (so that the number of nearest neighbor pairs K is given by $\frac{1}{2}(135^2 - 135) = 9045$). Indeed, all molecules in the database are of similar structure and function. In the context of a larger chemical database consisting of thousands or millions of molecules, one might suppose our 135 molecules are the result of some clustering [103] or pre-screening procedure [80] performed using a quickly computed similarity measure in order to isolate only the most likely matches to a given input. For example, one might use the lower bound obtained by the LP relaxation in Eq. (2.18) as a pre-screening criterion. We then wish to homogenize the most likely matches with respect to the graph edit distance using the optimal edit costs.

We used the permutation matrices that solve the binary linear program in Eq. (2.17) with unit costs to tabulate the edit operation counts in the vectors $\{H_j\}$ necessary for optimizing the cost metric. The publicly available `lp_solve` program was used to solve the integer program; it implements the simplex method in a branch-and-bound algorithm [7]. The optimal edit costs were then computed by solving the

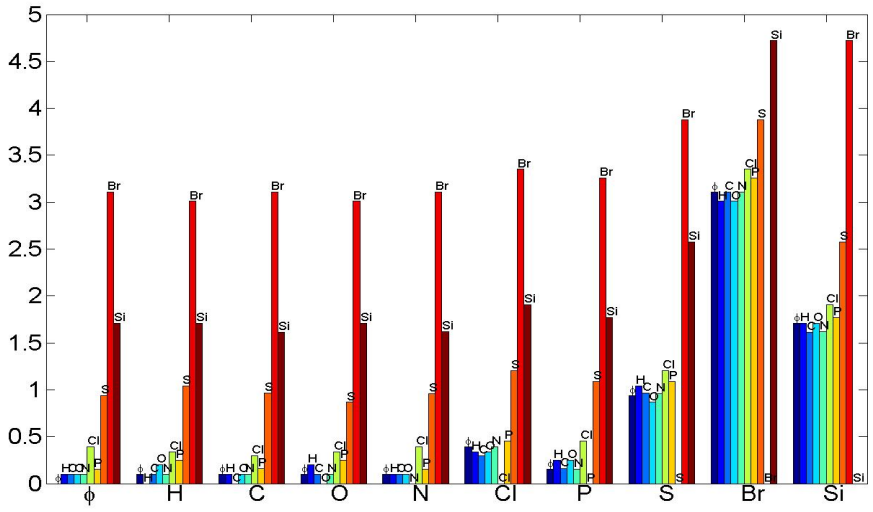


Figure 2.6: Optimal edit costs resulting from the convex program in Eq. (2.21). There is a label associated with each group of bars. Within the group, the edit cost of changing the group label to an individual label corresponds to the height of the bar below that individual label. The optimal edge edit cost $c(0,1)$ was 0.1.

convex program in Eq. (2.21) with $a = 0.1$ using a barrier method. The optimal edit costs for vertex relabelings are shown in Figure 2.6–the optimal edge edit cost was computed to be $c(0,1) = 0.1$. The associated edit counts tabulated over all pairs in the database matched with unity cost function are shown in Figure 2.7. The most frequently inserted/deleted atom types in matching the prototypes were H, C, and O, while the most frequent relabelings were $O \leftrightarrow H$ and $O \leftrightarrow N$. Note that there is roughly an inverse relationship between the number of times a particular edit occurs and its optimal cost, as one might expect. This does not hold exactly, however, because the edit costs must also satisfy the necessary triangle inequalities.

The maximum common subgraph (MCS) is frequently used as a similarity measure for chemical graphs [104]. Also, some graph metrics have been devised based on the MCS that are appropriate for comparison to our graph edit based metric [43, 13, 101]. There are some variations in the literature on what is meant by 'maximum common subgraph.' The differences amount to whether the vertices or the edges are the

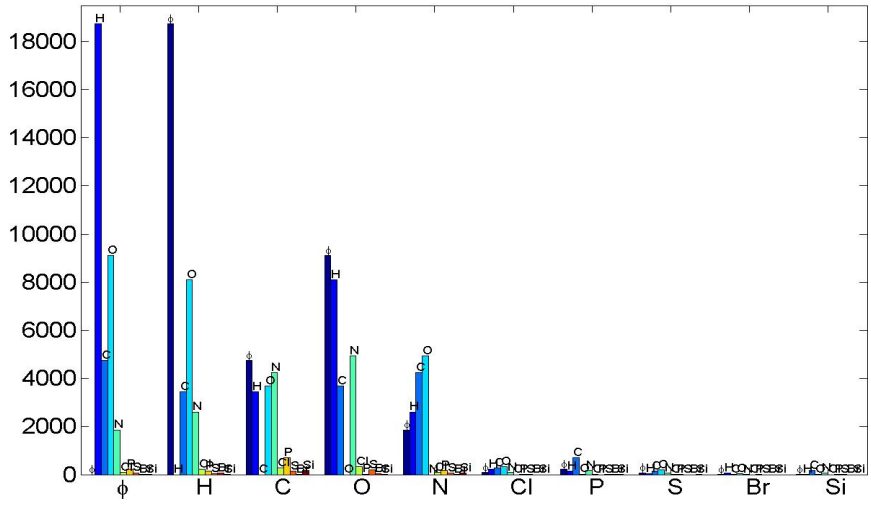


Figure 2.7: Total number of occurrences of each type of vertex edit tabulated over all pairs of database graphs matched with unit cost function. There is a label associated with each group of bars. Within the group, the edit cost of changing the group label to an individual label corresponds to the height of the bar below that individual label. We see that H, C, and O are the most frequently inserted/deleted atom types, and the most frequent relabelings are $O \leftrightarrow H$ and $O \leftrightarrow N$. Note that edits occurring more frequently are typically assigned a lower cost (Figure 2.6). There were 60974 total edge edits (not shown).

defining feature of the subgraph, resulting in a 'maximum common induced subgraph (MCIS)' or a 'maximum common edge subgraph (MCES)' respectively [81]. The MCIS is used in [19], while the MCES is used in [79, 35]. We will use the MCIS, which satisfies the MCS definition given in [11]. A slightly modified version of the distance metric proposed in [43] appropriate for the MCIS is given by

$$d_{mcs1}(G_0, G_1) = |V_0| + |V_1| - 2|V_{01}| \tag{2.23}$$

where $G_{01}(V_{01}, E_{01}, l_{01})$ is the MCS (MCIS) of graphs $G_0(V_0, E_0, l_0)$ and $G_1(V_1, E_1, l_1)$. Note that we are being somewhat careless with language–although we say 'the' MCS, it need not be unique. In addition to the MCS metric of Eq. (2.23), we also compared recognition performance to the following metric that is proposed in [13].

$$d_{mcs2}(G_0, G_1) = 1 - \frac{|V_{01}|}{\max(|V_0|, |V_1|)} \tag{2.24}$$

It has been shown that computing the MCS of graphs G_0 and G_1 is equivalent to computing the maximum clique in a modular product graph $G_p(V_p, E_p)$ [55]. In general finding the maximum clique is NP-Hard, so the worst case complexity is equivalent to binary linear programming [29]. The modular product graph is defined by the sets

$$
\begin{aligned}
V_p &= \{(v_0, v_1) \mid v_0 \in V_0, v_1 \in V_1, l_0(v_0) = l_1(v_1)\} \\
E_+ &= \{[(v_0, v_1), (u_0, u_1)] \mid v_0 \neq u_0, v_1 \neq u_1, (v_0, u_0) \in E_0, (v_1, u_1) \in E_1\} \\
E_- &= \{[(v_0, v_1), (u_0, u_1)] \mid v_0 \neq u_0, v_1 \neq u_1, (v_0, u_0) \notin E_0, (v_1, u_1) \notin E_1\} \\
E_p &= E_+ \cup E_-
\end{aligned}
\tag{2.25}
$$

We computed the MCS by using the algorithm in [71] to find the maximum clique in the modular product graph.

We calculated all 9045 pairwise distances between prototype graphs in the database using both the graph edit distance with optimal costs (GEDo) in Figure 2.6 and unit costs (GEDu), along with the two MCS distance metrics (MCS1 and MCS2) given in Eqs. (2.23) and (2.24) respectively. Histograms of the resulting pairwise distances are shown in Figure 2.8. Note that the GEDo pairwise distances are more concentrated around a single value than either of the MCS distances or the GEDu; this indicates the GEDo more uniformly distributes the prototypes in the graph metric space.

The ability of the four metrics to recognize input graphs as one of the prototype graphs in the database was tested next. An error-correcting graph isomorphism is indeed appropriate here, since each input graph was generated by applying a predetermined number of edits M (where $M \in \{1, 2, 3, 4, 5, 6\}$) to a randomly chosen prototype graph. The edits applied fell into one of the following five categories:

1. *edge edit*: M edges edits are selected with insertion and deletion having equal

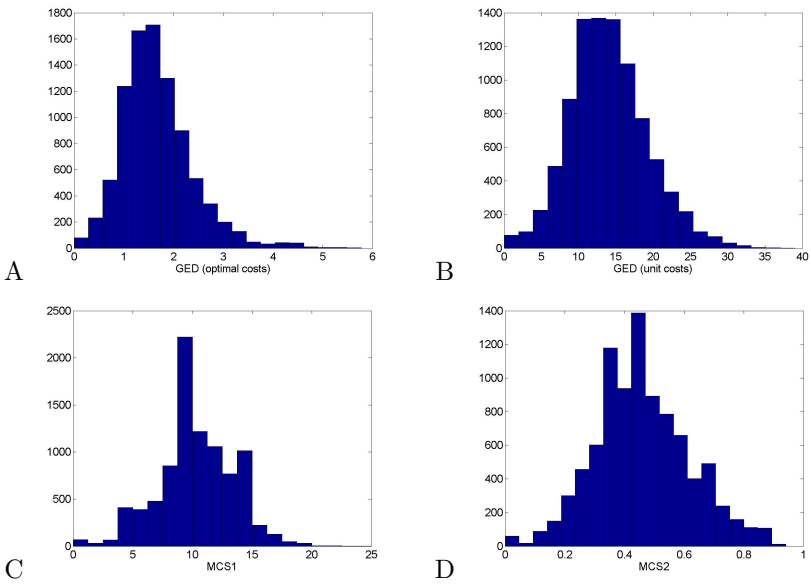


Figure 2.8: Pairwise distance histograms between all 9045 pairs of 135 prototype graphs in the database. Distances computed with the GEDo are shown in A, those computed with the GEDu are in B, those computed with the MCS1 metric are shown in C, and those computed with the MCS2 metric are in D. Ideally, all pairwise distances would be the same. Since the GEDo distances are more concentrated, the GEDo more uniformly distributes the prototype graphs. This should result in less ambiguity in the graph recognition phase, whereby the distance between a sample graph and each prototype graph is computed.

probability. Once the M edit operations are selected, pairs of vertices between which edges should be either inserted or deleted are selected at random.

2. *vertex deletion*: M vertices are selected to be deleted. First a label to be deleted is chosen with deletion probabilities given by normalizing the edit counts over the ϕ-group in Figure 2.7. Among the vertices having the chosen label, one is selected at random to be deleted along with all edges connected to it.

3. *vertex insertion*: M vertices are inserted. First a label to be inserted is chosen with insertion probabilities given by normalizing the edit counts over the ϕ-group in Figure 2.7. A vertex with the chosen label is then connected by a single edge to an existing vertex in the graph chosen at random.

4. *vertex relabeling*: M vertices are selected to be relabeled. First a pair of labels is chosen with probabilities given by normalizing the edit counts in Figure 2.7

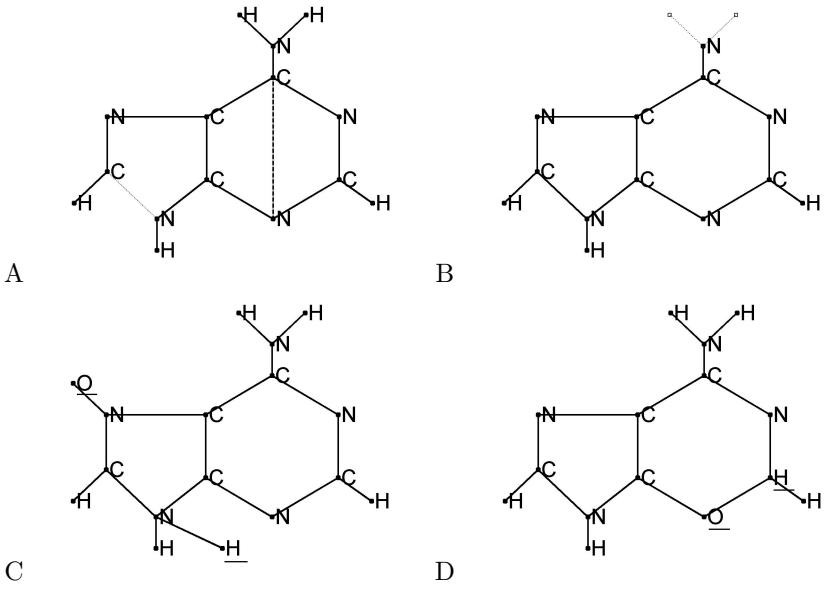


Figure 2.9: Example edits applied to the adenine chemical graph shown in Figure 2.5A. Two edge edits (one insertion, one deletion) are shown in A with the thick dashed line representing the inserted edge and the thin dashed line is the deleted edge. Two vertex deletions (represented by dotted lines and open boxes) are shown in B. C shows two vertex insertions (underlined), and D shows two vertex relabelings (underlined).

over all non-ϕ edits. Among the vertices having a label that matches one in the pair, one is selected at random and its label is changed to the complementary label in the pair.

5. *random*: The M edits to be performed are randomly chosen from the above four categories with each having equal probability.

Note that in performing vertex edits, we used the edit counts in Figure 2.7 as a guide so that the edits made would represent likely errors, say, in transcribing the chemical formula of one of the prototype graphs. Also, no regard was given to physical laws governing bonding, therefore some input graphs may not be physically realizable molecules. Examples of the different edit types applied to the adenine molecule are shown in Figure 2.9.

For each of the five edit categories, we generated ten input graphs from randomly chosen prototype graphs for each value of M (number of edits) ranging from 1 to

6; this resulted in $6 \times 10 \times 5 = 300$ sample input graphs. We then attempted to recognize the input graph by computing the distance (using GEDo, GEDu, MCS1, and MCS2) between the input graph and each of the 135 prototypes. There were $300 \times 135 = 40,500$ distinct input graph/prototype pairs matched using each of the four metrics to determine the corresponding graph distances. Due to the large number of matchings considered and the exponential complexity of the algorithms tested, we allowed a maximum of 45 seconds for any distance computation. If an optimal solution was not found within the allotted time, the best feasible suboptimal solution available was used. Running on Pentium 4, 2GHz processors, the average time required to solve the binary linear program necessary for GEDo or GEDu with `lp_solve` [7] was about 1.3 seconds, while the average time required to compute the maximum common subgraph using the maximum clique algorithm of [71] was about 0.1 seconds. Although the MCS routine is about ten times faster here, these times will vary depending on the particular algorithm/implementation one chooses for binary linear programming and maximum common subgraph detection.

We say an input graph is correctly recognized if it is closest (with respect to the appropriate distance metric) to the prototype graph from which it was generated. A 'classifier ratio' (CR) as given in Eq. (2.26) was computed for each input graph in order to gauge the level of ambiguity associated with the classification.

$$CR = \frac{d_*}{d_o} \tag{2.26}$$

Where d_* is the graph edit distance between the sample graph and the prototype from which it was generated, and d_o is the distance between the sample and the nearest incorrect prototype ('incorrect' in that the sample was not generated from this prototype). The lower CR is the less ambiguous the classification.

The proportion of graphs correctly recognized by each metric along with average

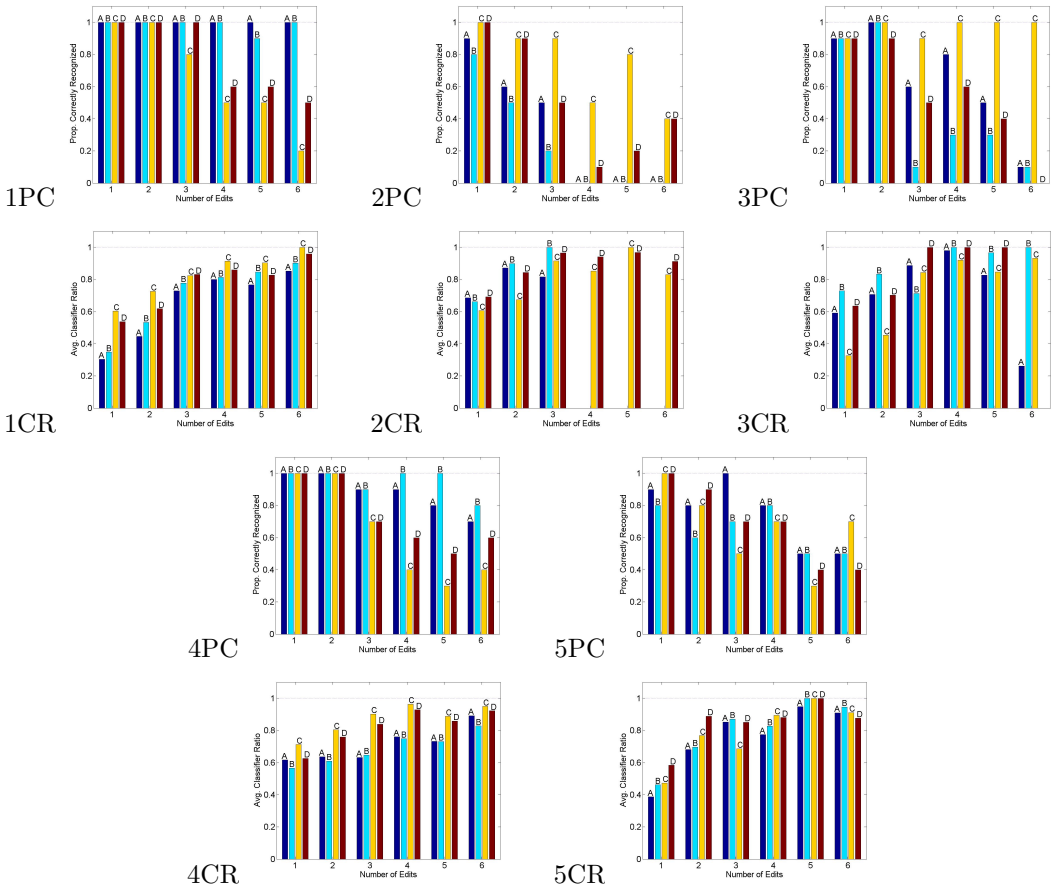


Figure 2.10: Proportion of graphs correctly recognized (PC) and average classifier ratios (CR) for the five different edit categories: 1) edge edit, 2) vertex deletion, 3) vertex insertion, 4) vertex relabeling, and 5) random. Within each plot, the letter above a bar denotes the metric used: A) GEDo (graph edit distance with optimal costs), B) GEDu (graph edit distance with unit costs), C) MCS1 (max common subgraph metric of Eq. (2.23)), and D) MCS2 (max common subgraph metric of Eq. (2.24)). Each set of four bars corresponds to a different number of edits M, indicated along the horizontal axis. Typically, as the number of edits increases, the proportion correctly recognized drops while the ambiguity of classification (as measured by the CR) rises. Note that the GED metrics perform better in the case of edge edits, vertex relabelings, and random edits (1, 4, and 5). The MCS metrics perform better in the case of vertex deletions and insertions (2 and 3). Marginal values of these distributions (averaged over M) are given in Table 2.2.

Metric	1) Edge Edit	2) Vertex Delete	3) Vertex Insert	4) Vertex Relabel	5) Random
GEDo	1.00 , 0.65	0.33 , 0.79	0.65 , 0.71	0.88 , 0.71	0.75 , 0.76
GEDu	0.98 , 0.70	0.25 , 0.85	0.45 , 0.87	0.95 , 0.69	0.65 , 0.80
MCS1	0.67 , 0.83	0.75 , 0.81	0.97 , 0.72	0.63 , 0.87	0.67 , 0.79
MCS2	0.78 , 0.77	0.52 , 0.89	0.55 , 0.87	0.73 , 0.82	0.68 , 0.85

Table 2.2: Proportion of graphs correctly recognized and average classifier ratio for each edit type category averaged over all graphs in that category. These are computed by marginalizing the plots in Figure 2.10 over the horizontal axis (number of edits, M). The first number in each pair is the proportion correctly recognized and the second number is the average classifier ratio (PC, CR). The GED metrics perform better in the case of edge edits, vertex relabelings, and random edits (1, 4, and 5); indeed, the GEDo metric correctly recognizes at least 75% of graphs in these categories. Only the MCS1 metric performs well in the case of vertex deletions and insertions (2 and 3) with at least 75% correct recognition in both cases. The GEDo metric (GED with optimal costs) has the lowest average CR in all categories but one, indicating reduced classification ambiguity.

classifier ratio associated with that metric for the five edit categories are shown in Figure 2.10. The classifier ratios were averaged only over those graphs that were correctly classified. The marginal values associated with these distributions averaged over the number of edits M are given in Table 2.2. Note that the GED metrics had superior performance in the edge edit, vertex relabeling, and random edit categories; the GEDo metric correctly recognizes at least 75% of graphs in these categories. The GED metrics were particularly successful in the edge edit category with all graphs correctly recognized by the GEDo metric, which also gave a consistently lower classifier ratio. The MCS1 metric was most robust in the case of vertex deletions and insertions (having at least 75% correct recognition); indeed both GED metrics had significant trouble when three or more vertices are deleted and trail off similarly in the case of vertex insertions. Undoubtedly, the changes on the prototype graph caused by inserting/deleting three or more vertices were so drastic that a different prototype was actually closer with respect to the GED to the sample graph produced. The GED metrics remained strong for up to five vertex relabelings, however, while the proportion correct for either MCS metric in this case decreased after three. In Table 2.2, we see that the optimal costs were indeed effective in reducing classification

ambiguity as measured by the CR since the GEDo metric has the lowest average CR in all categories but one.

2.4 Conclusion

This chapter develops a linear formulation of the graph edit distance for attributed graphs. We prove that the derived GED is a metric and show how to compute it using a binary linear program. Upper and lower bounds for the GED that can be computed in polynomial time are also given. A chemical graph recognition problem is presented as an application of the graph matching formalism. The edit costs are chosen using a normalized minimum variance criterion based on the prior information that the database graphs should be uniformly distributed in the graph metric space defined by the GED. This method is shown to give a metric that more uniformly distributes a database of 135 chemical graphs with similar structure than comparable maximum common subgraph based metrics. In recognizing chemical graphs generated by perturbing graphs in the database, the GED metrics with optimal costs and unit costs are shown to correctly recognize which prototype was perturbed more often than the MCS metrics in the case of edge edits and vertex relabelings. The MCS metrics perform better in the case of vertex insertions and deletions. When random edits are applied, the GED metrics are generally the best. Also, the GED with optimized edit costs is shown to have its intended effect of reducing the level of ambiguity associated with the chemical graph recognitions.

Unfortunately, the complexity of binary linear programming makes computing the GED between large graphs difficult using this method. However, the polynomial-time upper and lower bounds may be readily employed in this case. Also, these could be used in pre-screening on large chemical databases. For example, pre-screening may

be done by rejecting all molecules whose LP lower bound to the query exceeds a given value. Although we have developed a metric for unweighted graphs, it can be directly extended to graphs with edge weights provided the cost of editing these edges is proportional to the absolute difference in the weights with positive proportionality constant k. Indeed, one could proceed from Eq. (2.17) with weighted adjacency matrices A_0, A_1 used instead and $c(0,1)$ replaced by k. However, Eq. (2.17) would become a mixed integer program since, depending on the weights, S and T may not be binary matrices. Incorporating edge weights would yield a method applicable to 3-D structure searching of chemical graphs where weights are assigned to the graph edges based on the length of the bond they represent [23], along with other applications of weighted graphs. We anticipate the results of this chapter are applicable in any setting where it is necessary to compare graphical models.

CHAPTER III

Estimation of Message Source and Destination from Network Intercepts

3.1 Introduction

We present a method to estimate the endpoints (source and destination) of a data transmission in a network whose logical topology is unknown. We assume there are a number of asynchronous sensors placed on some subset of elements (links or nodes) in a network. A sensor is activated, and its activation recorded, whenever the path of a data transmission is intercepted on the element where the sensor is situated. The measurement apparatus is illustrated on a sample network in Fig. 3.1. Measurements are taken at discrete time instances, and the subscript k is used throughout the chapter to index time. If multiple sensors are activated by a single transmission, they may not be capable of providing the precise order in which they were activated. In general, a probability distribution on the possible orders of activation $P_k(\rho)$ is observed for each measurement; here the argument $\rho \in \{1, 2, \dots, \}$ is simply a natural number used to indicate a specific ordering of the sensors activated at time k. For example, a transmission with endpoints $u_1 = (\sigma_1, \delta_1)$ in Fig. 3.1 might activate $y_1 = \{\gamma_2, \gamma_3\}$–suppose this is the first measurement so $k = 1$. The ordering (γ_2, γ_3), corresponding to $\rho = 1$, might have probability $P_1(1) = \frac{3}{4}$, while the ordering (γ_3, γ_2), where $\rho = 2$, has probability $P_1(2) = \frac{1}{4}$. Since the orderings are defined

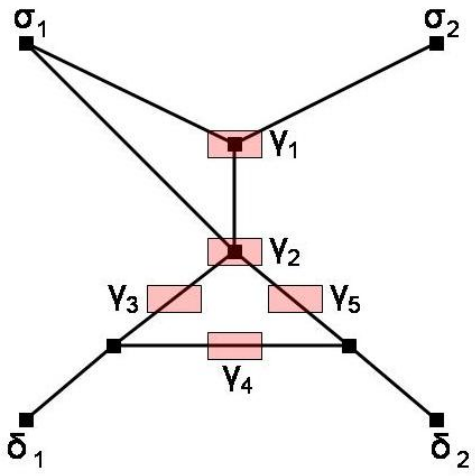


Figure 3.1: Diagram of the measurement apparatus on a sample network. Probing sites are sources $\Sigma = \{\sigma_1, \sigma_2\}$ and destinations $\Delta = \{\delta_1, \delta_2\}$. A box on a link or node represents a sensor that indicates when a transmission path intercepts that link/node. We see γ_1 and γ_2 monitor nodes while γ_3, γ_4, and γ_5 monitor links.

over distinct sensor sets, we implicitly assume the transmission does not cycle in its path–that is, a particular sensor is activated at most once by a single transmission. During a preliminary training phase, the network is probed by transmitting data packets between various pairs of probing sites $\{u_k = (\sigma_k, \delta_k)\}_{k=1}^{K_o-1}$, and the sensors $\{y_k\}_{k=1}^{K_o-1}$ activated by each transmission are recorded along with the distributions on orderings $\{P_k(\rho)\}_{k=1}^{K_o-1}$. A monitoring phase begins at time instant K_o and continues until some final time K, whereby we observe sensor activation sets $\{y_k\}_{k=K_o}^{K}$ and associated ordering distributions $\{P_k(\rho)\}_{k=K_o}^{K}$ for which the endpoints are unknown.

The probing data $\{x_k\}_{k=1}^{K_o-1} \equiv \{u_k, y_k, P_k(\rho)\}_{k=1}^{K_o-1}$, the monitored data $\{x_k\}_{k=K_o}^{K} \equiv \{y_k, P_k(\rho)\}_{k=K_o}^{K}$ and some prior information about the network topology are processed by the system shown in Fig. 3.2 to produce Monte Carlo estimates of the posterior distributions of possible endpoints of those transmissions observed in the monitoring phase. We allow prior information of the form $Q(\bar{A}) = b$ on the logical ($\{0,1\}$) adjacency matrix A describing connections among sensors and probing sites. $\bar{A}$ is some subset of the elements of A, Q is a fixed linear operator, and b is a vector. Thus

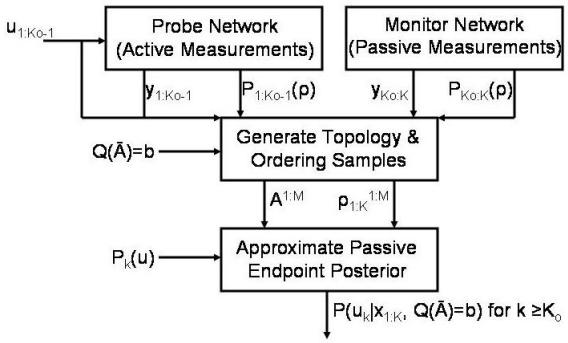


Figure 3.2: Diagram of the transmission endpoint estimation system, assuming sensors have already been deployed.

the prior information is essentially a set of linear equalities that the adjacency matrix A ought to satisfy. The linear operator Q can be expressed as an equivalent matrix if the elements of $\bar{A}$ are organized in a vector a. The linear prior information is then of the form $Qa = b$. In general, we make no assumptions about the structure of Q, so that given arbitrary Q and b the computation of feasible solutions to the linear equation is known to be an NP-Complete problem [72]. We consider the associated minimum norm problem $min \, \|Qa - b\|_\Lambda^2$ where $a \in \{0,1\}^n$ and $\|\cdot\|_\Lambda$ is a quadratic norm with respect to the positive definite matrix Λ. It is known that combinatorial optimization problems of this type may be successfully approximated by 'lifting' them into a higher dimensional matrix space where $X_{ij} = a_i a_j$ and $X \in \{0,1\}^{n \times n}$ [59].

With the advent of polynomial time interior point methods for linear programming that can be extended to semidefinite programming [85, 39], it is convenient to consider a semidefinite programming (SDP) relaxation of the higher dimensional problem. Indeed, SDP relaxations have proven to be powerful tools for approximating hard combinatorial problems [1, 31, 30, 38]. The SDP, however, is solved over a continuous domain so it is necessary to retrieve a 0-1 solution from the possibly fractional SDP solution. One possibility is a branch and bound scheme whereby certain variables

are fixed and the SDP is repeated until a discrete solution is found [72, 38]. The branch and bound algorithm can take an exponential amount of time, depending on how tight the desired bound is. A randomized rounding scheme was developed in [31] for SDP relaxations of the maximum cut (MAXCUT) and maximum 2-satisfiability (MAX2SAT) problems. This scheme is shown to produce solutions of expected value at least 0.878 times the optimal value in [31]. We develop an SDP relaxation of the 0-1 minimum norm problem and apply the randomized rounding method in conjunction with samples from the ordering distributions $\{\rho^m_{1:K}\}_{m=1}^M$ to produce a number of network topology adjacency matrices $\{A^m\}_{m=1}^M$ that approximately satisfy the linear prior information $Q(\bar{A}) = b$. We derive an expression for the expected value of the squared error $\mathbf{E}\left[\|Qa - b\|_\Lambda^2\right]$ of samples produced in this way. This expression depends on the solution of the SDP relaxation, but an upper bound on the error independent of the SDP solution is also given.

We wish to produce posterior distributions given the data and prior information of the endpoints of transmissions observed in the monitoring phase $P(u_k|x_{1:K}, Q(\bar{A}) = b)$ for $k \geq K_o$. The network topology and sensor ordering samples are used in conjunction with prior distributions on the endpoints of measurements made during the monitoring phase $P_k(u)$ for $k = K_o, K_o + 1, \ldots, K$ to compute Monte Carlo approximations of the desired posterior distributions via Bayes rule. Bayes formula for this problem essentially reduces to the expected value of a functional of the topology A and sensor ordering ρ; our approximation of the endpoint posterior thus becomes an average of the values of this functional at each sample topology A^m and ordering set $\rho^m_{1:K}$. It is readily apparent that this functional requires the conditionals $P(y|u, \rho, A)$–these path likelihood functions are the conditional probabilities of a sensor activation set y given the endpoints u and activation order ρ in a topology

A. We propose a path likelihood model inspired by shortest path routing, whereby the length of a path determines its probability. Since the model is probabilistic, it is also well suited to dynamic algorithms, such as distance vector routing [90], which may not always choose the same path for a single endpoint pair. With the endpoint posterior distribution in hand, we can immediately give the MAP estimate of u_k (with $k \geq K_o$) or an a posteriori confidence region of probable source/destination pairs.

The related area of network tomography has recently been a subject of substantial research. It refers to the use of traffic measurements over parts of a network to infer characteristics of the complete network. Some characteristics of interest include the following: source/destination traffic rates [98, 56], link-level packet delay distributions [88, 95, 52], link loss [14], and link topology [17, 24]. For an overview of relevant tomography problems see [53, 18]. In many applications, the tomography problem is ill posed since data is insufficient to determine a unique topology or delay distribution.

Our work is related to the internally sensed network tomography application described in [93, 78]. These works propose a methodology for estimating the topology of a telephone network using the measurement apparatus illustrated in Fig. 3.1. The data transmissions are of course telephone calls and the asynchronous sensors are located on trunk lines. A simple argument in [78] demonstrates that the number of topologies consistent with the data measured during the probing phase $\{x_k\}_{k=1}^{K_o-1}$ is exponential in the number of sensors. Indeed the problem is ill-posed as the data required to provide a reasonable estimate of the topology will never be available in practice. We sidestep the difficulties of developing a single topology estimate by averaging over many probable topologies in computing the endpoint posterior

distribution.

The solution approach we develop is very general, and we suspect it might have application in all sorts of networks: including telephone networks as described in [93], the Internet, social networks (such as command and control structures), or biological networks (such as protein-protein interaction networks) [70, 66]. Since we allow for sensor placement on arbitrary network elements, the method is equally applicable to networks where it may be more convenient to monitor nodes (as in the Internet) or monitor links (as in the telephone network of [93]). Also, the ordering distributions allow for situations involving sensors ranging from asynchronous to perfectly synchronized. At one extreme, the sensors are exactly synchronized–in which case the distribution $P_k(\rho)$ reduces to a delta function with all mass concentrated on the known ordering of sensors. A natural source of such information would be the noisy time stamp assigned by each sensor to when it saw the message. Indeed, this is an issue faced in many active probing scenarios. Methods involving GPS and calibration of PC clocks have been described in [73] and [74] respectively for addressing asynchronous sensors in active probing of technological networks. One might derive the ordering distributions from some noise model for the time stamps. In the present work, we assume the ordering distributions themselves are provided since the chronological order in which the sensors intercept a message is the crucial information.

Although the monitored network topology is unknown, the linear prior information permits inclusion of reasonably available information relevant to the topology. This is a generalization of the frequently used vertex degree prior. Vertex degree priors are used quite often due to the fact that many real world networks are characterized by specific degree distributions [4]. For example, studies have suggested a

power-law distribution describes vertex degrees in the Internet [26]. Such priors have recently been applied to research involving models of social and biological networks [70, 66, 33]. Since the degree of a vertex is equal to the sum over the row of the adjacency matrix describing connections to that vertex, one can easily construct a linear operator Q so that $Q(\bar{A}) = b$ expresses the degree prior for a given vector of vertex degrees b. Necessary conditions for connectivity among certain segments of the logical topology can also be expressed in this formalism by defining Q in terms of appropriate row sums.

The approach described here might also find utility in systems conveniently modelled by graphs, such as finite state automata. The problem of machine identification is a classic problem in the theory of automata testing [65, 54]. Here, we are given a black box with an automaton inside whose transition function is unknown. Based on the response of the system to certain input sequences, we wish to reconstruct the transition function. The link to the network topology recovery aspect of our problem is clear, since a graph provides a convenient representation for the transition function of interest. The probing sites chosen in the probing phase of our problem is analogous to the input sequences to the black box automaton. Similarly, link sensors correspond to events in the automaton's observable event set. An exhaustive algorithm for solving this problem is given in [65] and shown to have exponential run time. Our methods might be adapted to provide a polynomial time approximation algorithm. This would involve partitioning measurements with cycles (whereby an observable event occurs more than once in the same string) to satisfy the direct path assumption and selecting a different conditional path likelihood $P(y|u, \rho, A)$ since the shortest path routing model we suggest might not be appropriate.

The outline of this chapter is as follows. We review the problem, describe in

detail each component of the endpoint estimation system (Fig. 3.2), and analyze its complexity in Section II. In Section III, we provide some simulations of random graphs. In Section IV we conclude with some extensions of the method presented here and give directions for future work utilizing feedback for adaptive probing.

3.2 Model and Theory for Source-Destination Estimation

Let $G(V, E, f)$ be a simple graph defined by the vertex set V, edge set E, and incidence relation $f : E \to V \times V$ giving the vertices connected by each edge. We allow G to be either directed or undirected; however, it should be known a priori which is the case. In our application, E defines the set of links in the network topology, V defines the routers or switches connected by these links, and f determines the pair of routers/switches connected by each link. The graph G is unknown to us.

Let Γ denote a set of sensors we place in the network. Sensors are placed on some subset of graph elements; that is sensors may be placed on vertices, edges, or both. A sensor will indicate whenever a transmission through the network passes the element it is monitoring. Probing sites are selected from the vertex set V. The source vertex set $\Sigma \subseteq V$ is the set of vertices from which transmissions may originate, and the destination vertex set $\Delta \subseteq V$ are those vertices at which transmissions may terminate. A path observed at time $k < K_o$ between probing sites $s_k \in \Sigma$ and $d_k \in \Delta$ is given by $y_k \subseteq \Gamma$, where y_k contains the sensors activated by the transmission from s_k to d_k. We assume the first $K_o - 1$ measurements correspond to probes of the network (i.e. active measurements) so that the sources and destinations of these measurements are known (since we choose them). Because the sensors are in general asynchronous, the paths are unordered sets. However, along with each y_k, a discrete probability distribution $P_k(\rho)$ is observed on possible orderings indexed by ρ of the

set y_k; the observation data is then $x_k \equiv (u_k, y_k, P_k(\rho))$ for $k < K_o$. We assume a transmission does not cycle in its path from source to destination, so that only orderings of distinct elements of y_k are considered. It follows that if y_k has $|y_k|$ distinct elements, then $P_k(\rho)$ is defined over $|y_k|!$ different orderings. Note that the case of perfectly synchronized sensors is easily handled in this framework: simply take $P_k(\rho) = \delta(\rho - \rho_k)$ where ρ_k is the known order in which the sensors y_k were activated.

At time K_o we proceed with monitoring of the network, that is observing activated sensor sets with unknown source and destination. The observation data in the monitoring phase is $x_k \equiv (y_k, P_k(\rho))$ for $k \geq K_o$. The purpose of our system is to estimate the source and destination $u_k = (s_k, d_k)$ of an activated sensor set y_k. In order to estimate the endpoints of such a measurement, it is necessary to have some idea of the logical topology of the network. Instead of considering the logical adjacencies implied by the actual network $G(V, E, f)$, we are concerned with adjacency relationships among only those elements (vertices and edges) that are either monitored with a sensor or used as a probing site. For example, we cannot hope to pinpoint the position of a link e in the original network that is not monitored by a sensor. We assume unmonitored elements are essentially 'short-circuited' in the original network G. The idea here is to assure two elements are logically adjacent even if they are physically separated by an element (or subgraph of elements) that is not monitored. The particular topology we consider is then $G_A(V_A, E_A)$ where $V_A = \Gamma \cup \Sigma \cup \Delta$ is the set of sensors and probing sites, and $E_A \subseteq V_A \times V_A$ describes the logical adjacencies among these elements. G_A may be undirected or directed depending upon the nature of the network G. For computational purposes, we represent G_A by its adjacency matrix A where $A_{ij} = 1$ if and only if $(i, j) \in E_A$ for $i, j \in V_A$ and $A_{ij} = 0$ otherwise.

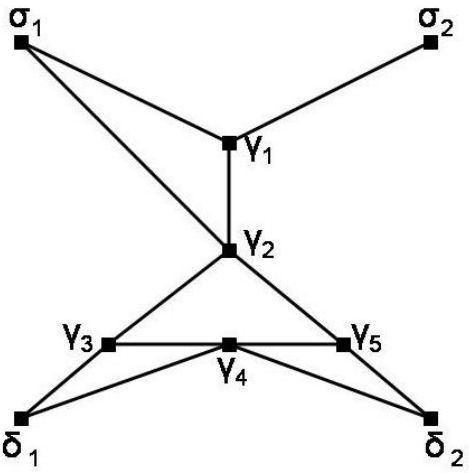


Figure 3.3: Example logical topology $G_A(V_A, E_A)$ for the monitored network G in Fig. 3.1. The vertex set of G_A consists of sensors $\Gamma = \{\gamma_i\}_{i=1}^5$ and probing sites $\Sigma = \{\sigma_1, \sigma_2\}$, $\Delta = \{\delta_1, \delta_2\}$, so that $V_A = \Gamma \cup \Sigma \cup \Delta$. The edges of G_A summarize logical adjacencies among sensors and probing sites with any intervening unmonitored elements short-circuited.

An example logical topology G_A is given in Fig. 3.3 for the monitored network G in Fig. 3.1.

We assume independence of measurements at different times and utilize a Bayesian framework to produce suitable approximations of the endpoint posterior distribution:

$$P(u_k|x_{1:K}, Q(\bar{A}) = b) = \mathbf{E}_{A,\rho_{1:K}}\left[\frac{P(y_k|u_k,\rho_k,A)P_k(u_k)}{\sum_u P(y_k|u,\rho_k,A)P_k(u)} \mid x_{1:K}, Q(\bar{A}) = b\right] \tag{3.1}$$

Where the expression is obtained quite simply by using the law of total probability to expand the distribution $P(u_k|x_{1:K}, Q(\bar{A}) = b)$ over the random variables A, $\{\rho_k\}_{k=1}^K \equiv \rho_{1:K}$ and applying Bayes rule with appropriate independence assumptions to $P(u_k|x_{1:K}, \rho_{1:K}, A)$. Of course $k \geq K_o$ in Eq. (3.1) so that we are considering the endpoint posterior of a passive measurement. We have available linear prior information on some of the logical adjacency elements $\bar{A}$ (a submatrix of A) of the form $Q(\bar{A}) = b$ where Q is a fixed linear operator and a prior distribution on endpoints $P_k(u)$. It is assumed the endpoint pair of a passive measurement is independent of the particular topology A, in other words, the parties communicating do not know the

network topology either. However, if there is no connection between a given endpoint pair u in a topology A, one would expect such a pair to have probability zero; we shall use a model for the term multiplying $P_k(u)$ to ensure the product is zero in this case. Here $x_{1:K} \equiv \{x_k\}_{k=1}^K$ represents all measured data ($x_k \equiv (u_k, y_k, P_k(\rho))$ for $k < K_o$ and $x_k \equiv (y_k, P_k(\rho))$ for $k \geq K_o$), and ρ_k is the ordering of the sensors activated in measurement y_k. The conditional expectation is therefore taken over all logical adjacency matrices A and sensor orderings for all measurements $\rho_{1:K}$. We introduce a shortest path routing model for the conditional path probabilities $P(y|u, \rho, A)$. The conditional expectation in Eq. (3.1) is approximated in a Monte Carlo fashion by summing over the argument evaluated at a number of adjacency matrix and sensor ordering samples. The sensor orderings $\rho_{1:K}$ are drawn independently from observed distributions $P_k(\rho)$ for $k = 1, 2, \ldots, K$. These are used in conjunction with the solution to a semidefinite programming relaxation that incorporates the prior information $Q(\bar{A}) = b$ to produce adjacency matrix samples A that are likely given both the data and the prior information. With the approximate endpoint posterior distribution in hand, we can provide MAP estimates of the endpoints of the passive measurement and compute appropriate error measures.

In the following, we first elaborate on probing of the network and the characterization of measurements obtained. Then we describe the distribution $P(A, \rho_{1:K}|x_{1:K}, Q(\bar{A}) = b)$ and how it may be efficiently sampled using the given ordering distributions and a semidefinite programming relaxation. Next we discuss how the samples are used to approximate the endpoint posterior and produce MAP estimates. Finally, we analyze the complexity of our algorithm.

3.2.1 Probing the Network and Taking Measurements

The set of all available measurements $\{x_k\}_{k=1}^{K}$ is partitioned into two disjoint sets. The measurements for $k = 1, 2, \ldots K_o - 1$ correspond to a training phase for the probing sites Σ, Δ. For each $k < K_o$, we select a probing pair $u_k \in \Sigma \times \Delta$ and pass a transmission between this pair to observe the sensors y_k activated and a distribution $P_k(\rho)$ on the $|y_k|!$ possible orderings of the activated sensors. The measurement data therefore consists of both the endpoints and the activated sensor set/ordering distribution $x_k = (u_k, y_k, P_k(\rho))$ for $k < K_o$. Such a measurement is referred to as an *active* measurement. The remaining measurements are due to monitored transmissions so that the endpoints are not available: $x_k = (y_k, P_k(\rho))$ for $K_o \leq k \leq K$. These are referred to as *passive* measurements since they were not due to active probing of the network on our behalf. It is assumed that the endpoints of these measurements are realizations of a random probing site pair described by the known distribution $P_k(u)$ defined on $\Sigma \times \Delta$. We desire to estimate the particular probing site pair between which a transmission was passed resulting in a given passive measurement.

3.2.2 Problem Statement

Our goal is to produce a MAP estimate of the endpoints u_k of some suspect observation $(y_k, P_k(\rho))$ taken at time k given all observations $x_{1:K}$ and the linear prior equalities $Q(\bar{A}) = b$ on the logical adjacency matrix. The MAP estimate is given by

$$\hat{u}_k = \arg\max_{u \in \Sigma \times \Delta} P(u | x_{1:K}, Q(\bar{A}) = b) \tag{3.2}$$

If the posterior distribution $P(u|x_{1:K}, Q(\bar{A}) = b)$ was known exactly, it would be simple to compute $\hat{u}_k$ because the optimization in Eq. (3.2) is over a discrete set.

Unfortunately, the computation of this posterior is of combinatorial complexity. We must therefore approximate this distribution and then use the approximation to produce the MAP estimate $\hat{u}_k$. We will shortly describe a semidefinite programming driven Monte Carlo method to calculate an approximate endpoint posterior $\hat{P}(u|x_{1:K}, Q(\bar{A}) = b)$. Our actual endpoint estimate is given as follows by plugging in the approximate posterior.

$$\hat{u}_k = \arg\max_{u \in \Sigma \times \Delta} \hat{P}(u|x_{1:K}, Q(\bar{A}) = b) \tag{3.3}$$

3.2.3 Generating Topology and Sensor Ordering Samples

In order to produce a Monte Carlo estimate of the conditional expectation in Eq. (3.1), we need to specify and sample from the distribution $P(A, \rho_{1:K}|x_{1:K}, Q(\bar{A}) = b)$. We first expand this distribution as

$$\begin{aligned} &P(A, \rho_{1:K}|x_{1:K}, Q(\bar{A}) = b) = \\ &P(A|x_{1:K}, \rho_{1:K}, Q(\bar{A}) = b) \textstyle\prod_{k=1}^{k} P(\rho_k|x_{1:K}, Q(\bar{A}) = b) \end{aligned} \tag{3.4}$$

where independence over the measurement time index k is used to write the second term in product form. We now note that each measurement x_k contains a distribution over orderings $P_k(\rho)$ for the activated sensor set. Since these distributions are observations, it is reasonable to suspect that all topological considerations are folded into them. We therefore assume that given the ordering distributions, the particular orderings ρ_k are independent of the linear prior on topology. Eq. (3.4) therefore becomes

$$\begin{aligned} &P(A, \rho_{1:K}|x_{1:K}, Q(\bar{A}) = b) = \\ &P(A|x_{1:K}, \rho_{1:K}, Q(\bar{A}) = b) \textstyle\prod_{k=1}^{K} P_k(\rho_k) \end{aligned} \tag{3.5}$$

The factored form of the distribution in Eq. (3.5) suggests the first thing we should do in generating our samples is to select orderings ρ_k independently from the dis-

tributions P_k for each $k = 1, 2, \ldots, K$. This is a simple matter since each P_k is a discrete distribution defined over a finite number of orderings.

Consider now what a measurement x_k equipped with an ordering ρ_k implies about the adjacency matrix A. Let $x_{k\rho_k}$ denote the ordered sensor activation set where, if x_k is an active measurement, the source probing site is taken as the first element followed by the ordering ρ_k of the activated sensors and the destination probing site is taken as the last element. If x_k is a passive measurement, $x_{k\rho_k}$ is simply the ordering ρ_k of the activated sensors. The fact that the transmission passes from the l^{th} element of $x_{k\rho_k}$, given by $x^l_{k\rho_k}$, to $x^{l+1}_{k\rho_k}$ implies there must be a logical connection between $x^l_{k\rho_k}$ and $x^{l+1}_{k\rho_k}$. Thus if we select an ordering ρ_k for each measurement (i.e. for $k = 1, 2, \ldots K$), then every adjacency element in the set $A^{x\rho}$ must be 1, where $A^{x\rho}$ is defined by

$$A^{x\rho} = \left\{ A_{ij} \mid \exists k, l : (x^l_{k\rho_k}, x^{l+1}_{k\rho_k}) = (i, j) \right\} \tag{3.6}$$

Once we draw orderings $\rho_{1:K}$ as previously described, the adjacency matrix elements in $A^{x\rho}$ are immediately fixed at unity by these. It remains, however, to select the remaining adjacency elements. In drawing these, we must account for the prior information $Q(\bar{A}) = b$. Since Q is a linear operator, we may re-express this information as $Qa = b$ where Q is now understood to be a matrix and $a \in \{0, 1\}^n$ is a vectorized version of the adjacency elements $\bar{A}$. For arbitrary Q, finding a 0-1 vector a that satisfies the equation $Qa = b$ is an NP-Complete problem [29]. We will shortly discuss how randomized rounding of a semidefinite programming relaxation may be used to find approximate solutions. The randomized rounding will induce a distribution on $P(A|x_{1:K}, \rho_{1:K}, Q(\bar{A}) = b)$, the remaining factor in Eq. (3.5). The induced distribution will have the desirable property that it assigns high probability to samples that approximately satisfy the linear constraint $Q(\bar{A}) = b$.

Consider the matrix equation $Qa = b$ equivalent to the linear prior information $Q(\bar{A}) = b$. Producing vectors a that satisfy this equation amounts to finding several solutions to the problem

$$\begin{aligned} &\text{find } a \in \{0,1\}^n \\ &\text{such that } Qa = b \end{aligned} \tag{3.7}$$

Unfortunately, the problem in Eq. (3.7) is NP-complete for an arbitrary, unstructured matrix Q [29]. We consider an equivalent restatement of Eq. (3.7)

$$\begin{aligned} &\text{minimize } (Qa-b)^T \Lambda (Qa-b) \\ &\text{such that } a \in \{0,1\}^n \end{aligned} \tag{3.8}$$

where Λ is a (symmetric) positive definite matrix that may be chosen to emphasize the relative importance of the different constraints. Obviously any optimal solution of the problem in Eq. (3.8) with zero value solves the feasibility problem in Eq. (3.7). The problem in Eq. (3.8) is no easier than the original statement, however, it has been shown that problems of this type (0-1 quadratic programs) can be approximated quite well using a semidefinite relaxation [30].

We now proceed to derive the SDP relaxation of Eq. (3.8). Our relaxation is similar to the one derived in [31] for MAX2SAT. First note that the optimization in Eq. (3.8) is equivalent to

$$\begin{aligned} &\text{minimize } a^T D a - 2d^T a \\ &\text{such that } a \in \{0,1\}^n \end{aligned} \tag{3.9}$$

where $D = Q^T \Lambda Q$ and $d = Q^T \Lambda b$. This is easily seen by expanding the objective in Eq. (3.8) and dropping the constant term. Now note that $a_i^2 = a_i$ since $a_i \in \{0,1\}$; this fact this allows Eq. (3.9) to be re-expressed as

$$\begin{aligned} &\text{minimize } \textstyle\sum_{i,j} D_{ij} a_i a_j - 2 \sum_j d_j a_j^2 \\ &\text{such that } a \in \{0,1\}^n \end{aligned} \tag{3.10}$$

We now introduce variables $w_i \in \{-1, 1\}$ for each $a_i \in \{0, 1\}$ for $i = 1 \dots n$ along with an additional $w_{n+1} \in \{-1, 1\}$ so that the change of variables is given by

$$a_i = \frac{1}{2}\left(1 + w_{n+1} w_i\right) \tag{3.11}$$

The identities in Eq. (3.12) follow from this change of variables.

$$\begin{aligned} & a_i a_j = \\ & \tfrac{1}{4}\left[(1 + w_i w_j) + (1 + w_{n+1} w_i) + (1 + w_{n+1} w_j) - 2\right] \\ & -a_i a_j = \\ & \tfrac{1}{4}\left[(1 - w_i w_j) + (1 - w_{n+1} w_i) + (1 - w_{n+1} w_j) - 4\right] \end{aligned} \tag{3.12}$$

If we introduce a negative sign in the objective, then the optimization in Eq. (3.10) becomes

$$\begin{aligned} & \max \tfrac{1}{4} \textstyle\sum_{i,j} \left[B_{ij}(1 + w_i w_j) + C_{ij}(1 - w_i w_j) - 4D_{ij}\right] \\ & \text{such that } w \in \{-1, 1\}^{n+1} \end{aligned} \tag{3.13}$$

where e is a vector of ones and matrices B, C are given by

$$\begin{aligned} B &= \begin{pmatrix} 0 & 2d \\ 2d^T & 0 \end{pmatrix} \\ C &= \begin{pmatrix} D & De \\ (De)^T & 0 \end{pmatrix} \end{aligned} \tag{3.14}$$

In order to obtain a semidefinite program, define the matrix $W = ww^T$. It is simple to show that $W = ww^T$ for some vector w if and only if $W \succeq 0$ (i.e. W is positive semidefinite) and $rank(W) = 1$. We drop the nonconvex rank-1 constraint to obtain the SDP relaxation

$$\begin{aligned} & \text{maximize } Tr\left[(B - C)W\right] \\ & \text{such that } \begin{aligned} & diag(W) = e \\ & W \succeq 0 \end{aligned} \end{aligned} \tag{3.15}$$

where $Tr[\cdot]$ indicates the trace operation and the constraint $diag(W) = e$ is added to enforce $w_i^2 = 1$. The equivalence of the objective functions in Eq. (3.15) and Eq. (3.13) can be seen easily by replacing $w_i w_j$ with W_{ij} and dropping constant terms. The SDP in Eq. (3.15) may be solved in polynomial time using a primal-dual path following algorithm [39]. The result of this optimization W^* will in general be a non-integer symmetric positive semidefinite matrix. In [31], a randomized rounding methodology is proposed to recover a -1,1 vector w from the SDP solution W^*. The strategy is to first perform the Cholesky factorization $W^* = V^T V$. A random hyperplane through the origin with normal vector r is then chosen by selecting r from the uniform distribution on the surface of the unit hypersphere $S_n = \{r \in \mathbf{R}^{n+1} | r^T r = 1\}$. The value of w_i is then determined by whether the corresponding column v_i of V lies above or below the hyperplane, i.e. $w_i = 1$ if $v_i^T r \geq 0$ and $w_i = -1$ if $v_i^T r < 0$. The i^{th} element of the vectorized adjacency sample $\hat{a}$ is then given by

$$\hat{a}_i = \begin{cases} 1 & \text{if } sign(v_i^T r) = sign(v_{n+1}^T r) \\ 0 & \text{if } sign(v_i^T r) \neq sign(v_{n+1}^T r) \end{cases} \tag{3.16}$$

This result can be seen by applying the rounding method and then using the change of variable formula given in Eq. (3.11).

We now proceed to derive the mean squared error $\mathbf{E}\left[\|Q\hat{a} - b\|_\Lambda^2\right]$ of the sample adjacency in Eq. (3.16) and thereby quantify how close the samples produced in this way come to satisfying the linear prior information on average. First note that the rounding scheme used implies the following identities.

$$\begin{aligned} \mathbf{E}[1 + w_i w_j] &= 2P\left(sign(v_i^T r) = sign(v_j^T r)\right) \\ \mathbf{E}[1 - w_i w_j] &= 2P\left(sign(v_i^T r) \neq sign(v_j^T r)\right) \end{aligned} \tag{3.17}$$

where r is a random vector from the uniform distribution on S_n as previously defined. We may evaluate the probabilities in Eq. (3.17) quite easily via the observation in [31]. Note that symmetry of the distribution implies $P\left(sign(v_i^T r) \neq sign(v_j^T r)\right) = 2P\left(v_i^T r \geq 0, v_j^T r < 0\right)$. And if $\theta = \arccos(v_i^T v_j)$ is the angle between the vectors v_i and v_j then it follows $P\left(v_i^T r \geq 0, v_j^T r < 0\right) = \frac{\theta}{2\pi}$ since the distribution of r is uniform on S_n. A similar argument applies to the case of matching sign. The results are summarized below.

$$
\begin{aligned}
P\left(sign(v_i^T r) = sign(v_j^T r)\right) &= 1 - \tfrac{1}{\pi}\arccos(v_i^T v_j) \\
P\left(sign(v_i^T r) \neq sign(v_j^T r)\right) &= \tfrac{1}{\pi}\arccos(v_i^T v_j)
\end{aligned}
\tag{3.18}
$$

If we define the matrix Z such that $Z_{ij} = \arccos(W^*_{ij})$ where W^* is the solution of the SDP relaxation in Eq. (3.15) and note that the objective function in Eq. (3.13) is exactly equal to $b^T \Lambda b - \|Q\hat{a} - b\|_\Lambda^2$, then we may take the expectation of the objective in Eq. (3.13) and apply the identities in Eqs. (3.17) and (3.18) to obtain the mean squared error as

$$
\mathbf{E}\left[\|Q\hat{a} - b\|_\Lambda^2\right] = \|Qe - b\|_\Lambda^2 - \frac{1}{2\pi} Tr\left[(C - B)Z\right] \tag{3.19}
$$

where e is a vector of ones.

We may obtain a bound on the expected value of the squared error in Eq. (3.19) independent of the solution to the SDP. As in [31], define the constant α

$$
\alpha = \min_{z \in [0,\pi]} \frac{2}{\pi} \frac{z}{1 - \cos z} \tag{3.20}
$$

From this definition of α, the following identities follow immediately

$$
\begin{aligned}
\tfrac{1}{2}\alpha(1 + \cos z) &\leq 1 - \tfrac{1}{\pi} z \\
\tfrac{1}{2}\alpha(1 - \cos z) &\leq \tfrac{1}{\pi} z
\end{aligned}
\tag{3.21}
$$

We take the expected value of the objective function in Eq. (3.13) and apply the identities in Eq. (3.21) with $Z_{ij} = \arccos(W^*_{ij})$ to give

$$
\begin{aligned}
& b^T \Lambda b - \mathbf{E}\left[\|Q\hat{a} - b\|^2_\Lambda\right] \geq \\
& \alpha \tfrac{1}{4}\left(\textstyle\sum_{i,j}\left[B_{ij} + C_{ij}\right] + Tr\left[(B-C)W^*\right]\right) - e^T D e
\end{aligned}
\tag{3.22}
$$

Now suppose the equation $Qa = b$ has at least one feasible solution a^0. Let w^0 be the corresponding -1,1 vector and $W^0 = w^0(w^0)^T$. We then have

$$
\begin{aligned}
& 0 = \|Qa^0 - b\|^2_\Lambda = e^T D e + b^T \Lambda b - \\
& \tfrac{1}{4}\left(\textstyle\sum_{i,j}\left[B_{ij} + C_{ij}\right] + Tr\left[(B-C)W^0\right]\right)
\end{aligned}
\tag{3.23}
$$

But since W^* solves the SDP in Eq. (3.15), it follows

$$
\begin{aligned}
& Tr\left[(B-C)W^*\right] \geq Tr\left[(B-C)W^0\right] = \\
& 4e^T D e + 4b^T \Lambda b - \textstyle\sum_{i,j}\left[B_{ij} + C_{ij}\right]
\end{aligned}
\tag{3.24}
$$

We may now combine the inequalities in Eqs. (3.22) and (3.24) and rearrange to obtain a bound on the expected value of the squared error that is independent of the SDP solution

$$
\mathbf{E}\left[\|Q\hat{a} - b\|^2_\Lambda\right] \leq (1-\alpha)\left(||Qe||^2_\Lambda + ||b||^2_\Lambda\right) \tag{3.25}
$$

In practice, the bound in Eq. (3.25) tends to exceed the true expected value in Eq. (3.19) by a large amount. However, it is of theoretical interest since it gives a general idea of how close samples produced in this way will come to satisfying the linear prior information, given the matrix Q and vector b specifying this information. One must be careful to apply this bound only when all elements of Q and b are nonnegative (such as when a vertex degree prior is used). A similar bound can be derived when some elements of Q or b are negative, but we will omit it here.

A naive procedure for generating the necessary samples using these procedures would be to first draw the ordering variables $\rho_{1:K}$ then fix the adjacency elements

in $A^{x\rho}$ corresponding to the draw. One could then reduce the system $Q(\bar{A}) = b$ by eliminating elements in $\bar{A} \cap A^{x\rho}$ and proceed to formulate and solve the SDP for use in randomized rounding. This approach is computationally prohibitive, however, because it requires solving a new SDP for every single sample. Instead, we prefer to solve a single SDP and use its solution to generate all samples. The single SDP is derived from the system $Q(\bar{A}) = b$ where the eliminated variables A_{ij} are those whose probability of being in the set $A^{x\rho}$ exceeds a threshold. The probability $P(A_{ij} \in A^{x\rho})$ is computed from the ordering distributions $P_k(\rho)$ as

$$P(A_{ij} \in A^{x\rho}) = \max_k \sum_{\rho \mid \exists l : x^l_{k\rho}=i, x^{l+1}_{k\rho}=j} P_k(\rho) \tag{3.26}$$

Note that by fixing the variables that are likely to be in $A^{x\rho}$ and eliminating them from the prior constraints $Q(\bar{A}) = b$, we are throwing away some prior information. Provided the threshold is fairly high, the eliminated variables will most often be set to unity anyway due to the ordering samples. In the interest of keeping down computational costs, this is a reasonable approach.

There may be adjacency matrix elements that are not in $\bar{A}$ and have zero probability of being in $A^{x\rho}$. Define $A^o \equiv \{A_{ij} \mid A_{ij} \notin \bar{A}, P(A_{ij} \in A^{x\rho}) = 0\}$; A^o then denotes the adjacency matrix elements that we have no information about. We adopt the principle of parsimony and assume all elements in A^o are zero. A summary of our procedure for generating M sample adjacency matrices and orderings follows.

- Compute $P(A_{ij} \in A^{x\rho})$ for all $A_{ij} \in \bar{A}$ as in Eq. (3.26).
- Eliminate $\{A_{ij} \mid P(A_{ij} \in A^{x\rho}) \geq \delta\}$ from $\bar{A}$ and adjust the system $Q(\bar{A}) = b$ with these variables fixed at 1.
- Solve the SDP corresponding to $Q(\bar{A}) = b$ for the optimum W^*.

- Compute and store the Cholesky factor V of the SDP solution W^*.
- For $m = 1, 2, \ldots M$
  - Draw ρ_k from $P_k(\rho)$ for $k = 1, 2, \ldots, K$.
  - Determine $A^{x\rho}$ as in Eq. (3.6) and set $A_{ij} = 1$ for all $A_{ij} \in A^{x\rho}$.
  - Draw r from the uniform distribution on S_n.
  - Take inner products of the Cholesky factors with r to determine $A_{ij} \notin A^{x\rho}$ that are organized in the vector a as shown in Eq. (3.16).
  - Set all remaining adjacency elements to 0.

We may now write down the conditional distribution $P(A|x_{1:K}, \rho_{1:K}, Q(\bar{A}) = b)$ from which the SDP rounding method is sampling. First define the set $H(A_{ij})$ as

$$H(A_{ij}) = \begin{cases} \{r \in S_n \mid sign(v_{ij}^T r) = sign(v_{n+1}^T r)\} & \text{if } A_{ij} = 1 \\ \{r \in S_n \mid sign(v_{ij}^T r) \neq sign(v_{n+1}^T r)\} & \text{if } A_{ij} = 0 \end{cases} \tag{3.27}$$

where S_n is the surface of the unit hypersphere and v_{ij} is the appropriate column of the Cholesky factor V corresponding to the variable A_{ij} as defined earlier. Since the only random elements of A given $x_{1:K}$, $\rho_{1:K}$ and $Q(\bar{A}) = b$ are those in $\bar{A} - A^{x\rho}$, the desired conditional distribution is given by

$$P(A|x_{1:K}, \rho_{1:K}, Q(\bar{A}) = b) = \frac{Vol\left(\bigcap_{A_{ij} \in \bar{A} - A^{x\rho}} H(A_{ij})\right)}{Vol(S_n)} \tag{3.28}$$

The expression in Eq. (3.28) is a rather complicated distribution shaped by the prior data Q and b through the solution of the SDP in Eq. (3.15) formulated from this data. Luckily, we do not need to evaluate it. The crucial point is that samples from this distribution will approximately satisfy the prior information $Q(\bar{A}) = b$. In

order to investigate the quality of individual samples, define the sublevel set S_ϵ of adjacency matrices as follows:

$$S_\epsilon = \left\{ A \in \{0,1\}^{n \times n} \;\middle|\; \frac{||Q(\bar{A}) - b||_\Lambda^2}{||Q(ee^T)||_\Lambda^2 + ||b||_\Lambda^2} \leq \epsilon \right\} \tag{3.29}$$

where we have resumed the operator notation for Q and ee^T is a matrix of ones used to coincide with the vectorized notation in Eq. (3.25). It follows from the definition that if $\epsilon_1 \leq \epsilon_2$ then $S_{\epsilon_1} \subseteq S_{\epsilon_2}$. Ideally we would prefer samples from S_0 so that the prior linear equalities are exactly satisfied, however we settle for adjacency samples from the larger set S_ϵ for some tolerance $\epsilon > 0$. For $\epsilon \geq 1 - \alpha \approx 0.88$, we can apply the Markov inequality along with the bound in Eq. (3.25) to give the following general result for sample adjacencies $\hat{A}$ produced using this method.

$$P(\hat{A} \in S_\epsilon) \geq 1 - \frac{1-\alpha}{\epsilon} \tag{3.30}$$

Although the method is capable (in principle) of producing any adjacency matrix in S_∞, a lower bound for the proportion of samples falling in the tolerance set S_ϵ is given by Eq. (3.30). One might investigate further the shape of the sampling distribution in Eq. (3.28) in order to determine the relative likelihood of different adjacencies, however we will conclude our analysis here.

3.2.4 Approximating the Endpoint Posterior

We use the topology and sensor ordering samples obtained in the previous section to derive an approximate endpoint posterior distribution of a passive measurement indexed by k as given in Eq. (3.1). If $\{A^m\}_{m=1}^M$ are the topology samples and $\{\rho_{1:K}^m\}_{m=1}^M$ are the sensor ordering samples (for each measurement), then the strong law of large numbers suggests a Monte Carlo estimate of the conditional expectation

given by

$$\hat{P}(u_k|x_{1:K}, Q(\bar{A}) = b) = \frac{1}{\kappa}\sum_{m=1}^{M} \frac{P(y_k|u_k,\rho_k^m,A^m)P_k(u_k)}{\sum_u P(y_k|u,\rho_k^m,A^m)P_k(u)} \tag{3.31}$$

where κ is a normalization constant inserted to ensure the total mass of the approximate posterior is unity. Since we are given a distribution on the endpoints of the passive measurement $P_k(u)$, we need only specify a model for the conditional path probability $P(y|u, \rho, A)$ in order to approximate the posterior as in Eq. (3.31). Routing mechanisms and traffic data might figure prominently into such a model. We propose a simple model whereby the length of a path determines its probability (as in shortest path routing). If $|y_\rho|$ denotes the length of the ordered path y_ρ, and $y_\rho^{u,A}$ denotes the shortest ordered path between endpoints u in topology A, then the conditional distribution is given by

$$P(y|u,\rho,A) = \begin{cases} \theta & \text{if } |y_\rho| = |y_\rho^{u,A}| < \infty \\ 1-\theta & \text{if } |y_\rho^{u,A}| < |y_\rho| < \infty \\ 0 & \text{if } |y_\rho| = \infty \end{cases} \tag{3.32}$$

The model basically says that the shortest path between endpoints u in topology A is chosen with probability θ, and all other valid paths (that is, paths of finite length) have probability $1 - \theta$. If a path does not connect the endpoints u in the given topology A, then naturally it has zero probability. Note that for arbitrary θ, we need to run Dijkstra's algorithm (or some other shortest path routing algorithm) for each topology sample A^m in order to compute the conditional path probability in Eq. (3.32) [72]. This is not necessary, however, in the case that $\theta = \frac{1}{2}$.

We may give maximum a posteriori (MAP) estimates of the endpoints u_k of a passive measurement y_k after computing the posterior distribution estimate in Eq.

(3.31). Indeed, the MAP estimate is simply given by

$$\hat{u}_k = \arg\max_{u \in \Sigma \times \Delta} \hat{P}(u | x_{1:K}, Q(\bar{A}) = b) \tag{3.33}$$

Recall that $u \equiv (s, d)$, thus MAP estimates of s_k or d_k individually may be obtained by maximizing the appropriate marginal $\hat{P}(s|x_{1:K}, Q(\bar{A}) = b)$ or $\hat{P}(d|x_{1:K}, Q(\bar{A}) = b)$ respectively.

We use as an error measure the ratio $\Lambda_u(k)$ below for the estimated endpoints $\hat{u}_k$.

$$\Lambda_u(k) = \frac{\hat{P}(\hat{u}_k | x_{1:K}, Q(\bar{A})=b)}{\hat{P}(\hat{u}_k | x_{1:K}, Q(\bar{A})=b) + \max\limits_{u \in \Sigma \times \Delta - \hat{u}_k} \hat{P}(u | x_{1:K}, Q(\bar{A})=b)} \tag{3.34}$$

It is also useful to compute the corresponding ratios associated with the marginalized distributions $\Lambda_s(k)$ and $\Lambda_d(k)$, as it may be the case that either the source or destination of a passive measurement is more accurately determined individually than are both collectively. These are defined exactly as in Eq. (3.34), except u is replaced with s or d throughout (so that the appropriate marginal distribution is considered). It is clear that the ratio in Eq. (3.34) must lie in the interval $[\frac{1}{2}, 1]$. Larger values of this ratio in a sense indicates more 'confidence' in the MAP estimate since a value of 1 is achieved only when all of the mass of the estimated posterior distribution is concentrated at the MAP estimate.

3.2.5 Algorithm Complexity

We now analyze the complexity of the source/destination estimation scheme developed here. The two fundamental quantities that determine the size of the problem are denoted by N and h; N is the total number of sensors plus probing sites, so that $N = |\Gamma| + |\Sigma \cup \Delta|$, while h is the maximum number of activated sensors in any measurement, so that $|y_k| \leq h$ for all $k = 1, 2, \ldots, K$. The maximum number of hops h may be a function of N, depending upon the type of network considered.

For networks that obey the *small world* effect, as many real world networks do, h will remain approximately constant with increasing N [64, 70]. The number of measurements K and the number of Monte Carlo samples M also affect the complexity; however we shall see the complexity dependence on these is always linear.

First note that we must store the ordering distributions $P_k(\rho)$ for all measurements. Since each distribution is defined over $O(h!)$ orderings, this requires $O(Kh!)$ space. The adjacency matrix A considers all logical connections among sensors and probing sites, so that A has $O(N^2)$ elements. In the worst case, the linear prior information $Q(\bar{A}) = b$ will constrain all elements of this matrix so that $\bar{A} = A$. It will therefore take $O(KN^2h!)$ time to compute $P(A_{ij} \in A^{x\rho})$ for all $A_{ij} \in \bar{A}$. Now in the worst case, thresholding these probabilities will produce a negligible reduction in the size of the system $Q(\bar{A}) = b$, so that we still have to contend with $O(N^2)$ variables in solving the SDP relaxation. Typically interior point methods are used to solve SDP's to within ϵ of the optimal solution. These are based on Newton's method; therefore at each iteration it is necessary to solve a linear system of equations for the Newton directions ($O(n^3)$ for a system of size n). An algorithm given in [57] is shown to take $O(|\log \epsilon|\sqrt{n})$ iterations for a problem of size n–this performance is typical for all interior point algorithms. Our problem has dimension $O(N^2)$, thus solving the SDP takes $O((N^2)^{3.5})$ or $O(N^7)$ time. A Cholesky factorization is then performed on the SDP solution, which takes $O((N^2)^3)$ or $O(N^6)$ time.

After solving the SDP, the M topology and ordering samples may be produced relatively quickly. For each sample, we need to draw an ordering for each of the K measurements, thus requiring $O(MK)$ time to produce the ordering samples. Given the K orderings for a single sample, $A^{x\rho}$ may be generated in $O(Kh)$ time. Finally, we may draw the vector r and take inner products to determine the remaining elements

of the topology sample. Since the time required for each inner product is linear, it takes a total of $O(MN^2 + MKh)$ time to produce the M topology samples.

The final step is to compute the Monte Carlo approximation of the endpoint posterior distribution of a passive measurement. A quick inspection of Eq. (3.31) reveals that we need to determine the conditional path probabilities $P(y|u, \rho, A)$ for every endpoint pair u–there are $O(N^2)$ such pairs. Also, computing each path probability for a given ordered path y_ρ requires tracing this path through the topology A, which takes $O(h)$ time. Now, if $\theta \neq \frac{1}{2}$ we must take $O(MN^3)$ time to run a shortest path algorithm on each sample [72]. Therefore, it takes $O(MN^2h+MN^3)$ to produce the approximate endpoint posterior for $\theta \neq \frac{1}{2}$; this reduces to $O(MN^2h)$ for $\theta = \frac{1}{2}$.

The factors that give some cause for concern in this algorithm are the $h!$ in considering all possible orderings and the N^7 in the SDP solution complexity. If we are dealing with small world networks, then h might be around four or five so that $h!$ is still manageable. And if this is not the case, one would hope that the ordering distributions $P_k(\rho)$ are nonzero only over a reasonable number of orderings since we need only consider ρ with $P_k(\rho) > 0$. In practice, the actual SDP complexity is likely to be significantly less than the worst case bound of $O(N^7)$ after reducing the system $Q(\bar{A}) = b$, especially if the original prior only constrains some small subset of the adjacency elements. The complexity might be significantly reduced, however, if there is some natural decoupling of the equalities. In this way, one might solve several smaller SDP's rather than a single large one. Our algorithm would still benefit from speedy SDP algorithms as solving the relaxation takes the most time in the worst case. A parallel implementation of an interior point algorithm for SDP's might reduce the time requirements if multiple processors are available [68].

Figure 3.4: Example sparsity patterns for the adjacency matrices of three undirected random graphs with 25 nodes and 40 randomly selected links. The method is illustrated by simulating on topologies of this type. 12 of the 25 nodes are selected as probing sites: 6 of these are taken as sources Σ and 6 are taken as destinations Δ. We assume in one case that sensors are placed on all 40 links (100% sensor coverage) and in another that sensors are placed on 30 randomly selected links (75% sensor coverage). Active measurements consist of 18 of the 36 distinct pairs in $\Sigma \times \Delta$ randomly selected for use in the probing phase, denoted L. The remaining 18 pairs are denoted L^c. Passive measurements consist of sensor activations monitored between all pairs in $\Sigma \times \Delta$. Shortest path routing is used to determine the transmission path.

3.3 Simulations

We performed some numerical simulations to demonstrate the utility of the method described in this chapter. We generated undirected random graphs with 25 nodes to serve as test networks. The number of edges in each graph was fixed at 40 by randomly selecting 40 of the possible 300 vertex pairs and connecting the selected pairs by an edge. The adjacency matrix sparsity patterns for three example graphs are shown in Fig. 3.4. We randomly chose 12 of the 25 nodes to serve as probing sites–this set was then partitioned in half so that both the source set Σ and destination set Δ each had 6 distinct elements. Sensors were placed on links in the network for two cases: 100% sensor coverage (in which all 40 links were monitored by a sensor) and 75% sensor coverage (in which 30 of the 40 links were selected at random for hosting a sensor). In the 75% coverage case, networks were generated in a rejection sampling manner so that every measurement (whether passive or active) activated at least one sensor.

In order to probe a network, we randomly selected 18 of the 36 distinct pairs in $\Sigma \times \Delta$ to serve as endpoints for active measurements. This set of 18 endpoint

pairs is denoted $L \subset \Sigma \times \Delta$; the remaining pairs are denoted by $L^c \equiv \Sigma \times \Delta - L$. Sensor activations in response to transmissions between all pairs in $\Sigma \times \Delta$ were observed in the monitoring phase. All transmissions were routed through the network using shortest path routing, and activated sensor sets y_k were observed. Thus for each network we had $K = 54$ data points: $K_o - 1 = 18$ active measurements $x_{1:18} \equiv (u_{1:18}, y_{1:18}, P_{1:18}(\rho))$ and 36 passive measurements $x_{19:54} \equiv (y_{19:54}, P_{19:54}(\rho))$. For each data point ($k = 1, 2, \ldots K$), a distribution on the order in which sensors were activated $P_k(\rho)$ was generated as follows: first the true ordering of sensors ρ_k was noted, then noise $n(\rho)$ was drawn independently from the $Uniform[0, 0.2]$ distribution for $\rho = 1, 2, \ldots |y_k|!$, finally the distribution $P_k(\rho)$ was generated by normalizing the corrupted delta function distribution as in Eq. (3.35).

$$P_k(\rho) = \frac{\delta(\rho - \rho_k) + n(\rho)}{\sum_{\rho=1}^{|y_k|!} \delta(\rho - \rho_k) + n(\rho)} \tag{3.35}$$

The linear prior information was generated from degree information on the logical topology A. Indeed vertex degree information is a commonly used special case of the more general linear prior specified by $Q(\bar{A}) = b$ [66, 33]. The sensor degree, that is the number of sensors b_i to which the i^{th} vertex in the logical topology is adjacent, was known for all $v_i \in V_A$. In addition to knowing the sensor degrees of vertices in the logical topology G_A, a random subset consisting of no more than 60% of the sensors not adjacent to a given vertex were also known. For the i^{th} vertex, the i^{th} row of the operator $Q_i(\bar{A})$ therefore sums over the elements of A for which adjacency to vertex i is uncertain, and the i^{th} element of b, b_i, is simply the known sensor degree of vertex i. As an example, consider vertex γ_5 of the logical topology in Fig. 3.3. Vertex γ_5 is adjacent to sensors $\{\gamma_2, \gamma_4\}$, therefore its sensor degree is two. Since there are two sensors not adjacent to γ_5, $\lfloor 2 * 60\% \rfloor = 1$ sensor, say γ_x, is selected at random from the set $\{\gamma_1, \gamma_3\}$. Let Γ_i denote the set of sensors known to

be nonadjacent to vertex i, so that $\Gamma_{\gamma_5} = \{\gamma_x\}$ in this example. The row of the prior $Q(\bar{A}) = b$ corresponding to γ_5 is then given by $\sum_{j \in \Gamma - \gamma_5 - \Gamma_{\gamma_5}} A_{\gamma_5 j} = 2$. Similarly, we construct the entire operator Q for the topology in Fig. 3.3 as follows

$$Q(\bar{A}) = \begin{pmatrix} \sum_{j \in \Gamma - \gamma_1 - \Gamma_{\gamma_1}} A_{\gamma_1 j} \\ \sum_{j \in \Gamma - \gamma_2 - \Gamma_{\gamma_2}} A_{\gamma_2 j} \\ \sum_{j \in \Gamma - \gamma_3 - \Gamma_{\gamma_3}} A_{\gamma_3 j} \\ \sum_{j \in \Gamma - \gamma_4 - \Gamma_{\gamma_4}} A_{\gamma_4 j} \\ \sum_{j \in \Gamma - \gamma_5 - \Gamma_{\gamma_5}} A_{\gamma_5 j} \\ \sum_{j \in \Gamma - \Gamma_{\sigma_1}} A_{\sigma_1 j} \\ \sum_{j \in \Gamma - \Gamma_{\sigma_2}} A_{\sigma_2 j} \\ \sum_{j \in \Gamma - \Gamma_{\delta_1}} A_{\delta_1 j} \\ \sum_{j \in \Gamma - \Gamma_{\delta_2}} A_{\delta_2 j} \end{pmatrix} \tag{3.36}$$

The vector b for this example topology is $\begin{pmatrix} 1 & 3 & 2 & 2 & 2 & 2 & 1 & 2 & 2 \end{pmatrix}^T$ obtained simply by reading the number of sensors (γ vertices) adjacent to each vertex in Fig. 3.3.

Given the sensor degree prior information and the ordering distributions, we eliminated those adjacency elements whose probability of being in the set $A^{x\rho}$ exceeded $\frac{1}{2}$ from $\bar{A}$, where $P(A_{ij} \in A^{x\rho})$ was computed as in Eq. (3.26). The reduced system $Q(\bar{A}) = b$ was then used to formulate the SDP relaxation in Eq. (3.15) for the minimum norm solution with the weight matrix Λ taken as the identity. The relaxation was solved with a predictor-corrector path following algorithm given in [39]. A publicly available C implementation of this algorithm was used [9]. The SDP solution was used along with the ordering distributions $P_k(\rho)$ to produce $M = 500$ samples of measurement orderings $\rho_{1:K}$ and adjacency matrices A for computing the Monte Carlo estimates of the endpoint posteriors.

We assumed the endpoint priors $P_k(u)$ were uniform over $\Sigma \times \Delta$ for all 36 passive measurements $k = 19, 20, \ldots, 54$. Also, the parameter θ in the conditional path probabilities of Eq. (3.32) was taken as $\frac{1}{2}$ so that it was not necessary to run a shortest path routing algorithm on every sample topology. The 500 ordering and topology samples were then used to compute the approximate endpoint posteriors for all passive measurements as given in Eq. (3.31). These were used to produce joint MAP estimates of the transmission endpoints and to compute the resolution measures $\Lambda_u(k)$. An example endpoint posterior is given in Fig. 3.5, for which the correct endpoint pair is source no. 6 and destination no. 3. It is clear that the MAP estimate will result in the correct pair in this case. Also indicated in the Figure is the second most likely pair $u = (6, 5)$; this is used in computing the resolution measure Λ_u as in Eq. (3.34)–$\Lambda_u(k) = 0.60$ for this case. Marginal distributions of the approximate posterior are given in Fig. 3.6. These were used in individual MAP estimation of source and destination. It is clear that the individual estimates will match the joint estimate for this case; the resolution measures were a bit lower though with $\Lambda_s(k) = 0.58$ and $\Lambda_d(k) = 0.59$. This completes the simulation process for a single graph.

We repeated the simulation procedure for 30 networks with 100% sensor coverage and 30 networks with 75% sensor coverage. Table 3.1 demonstrates the effectiveness of the SDP randomized rounding algorithm for producing topology samples that approximately agree with the sensor degree prior information. It lists the normalized squared topology sample error $\frac{1}{||Qe||^2+||b||^2} \frac{1}{M} \sum_m \left\|Q\hat{a}^m - b\right\|^2$ averaged over the $M = 500$ samples along with the normalized expected squared error $\frac{1}{||Qe||^2+||b||^2} \mathbf{E}\left[\left\|Q\hat{a} - b\right\|^2\right]$ as in Eq. (3.19) for each graph. The bound derived in Eq. (3.25) assures the expected squared error can never exceed $1 - \alpha \approx 0.12$. We see

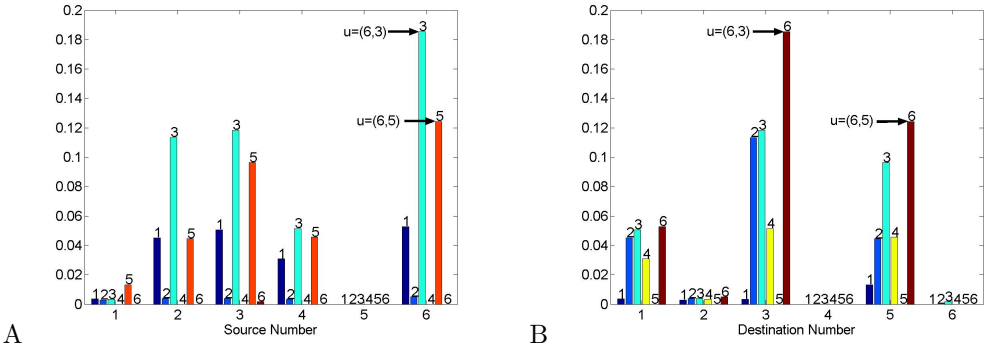


Figure 3.5: Example endpoint posterior distribution $\hat{P}(u_k|x_{1:K}, Q(\bar{A}) = b)$ for a passive measurement at time $k \geq K_o$ with endpoints $u = (s, d) = (6, 3)$. In plot A, the probabilities are grouped by source, with each of 6 bars in a group corresponding to a different destination (noted above the individual bar). Plot B displays the same information except probabilities are grouped by destination with source number noted above each individual bar. The largest and second largest values of the posterior are indicated–it is these values that are used in computing the resolution ratio Λ_u of Eq. (3.34), calculated as $\Lambda_u(k) = 0.60$. It is clear in this example that the endpoints of this transmission will be correctly estimated by the joint MAP estimate.

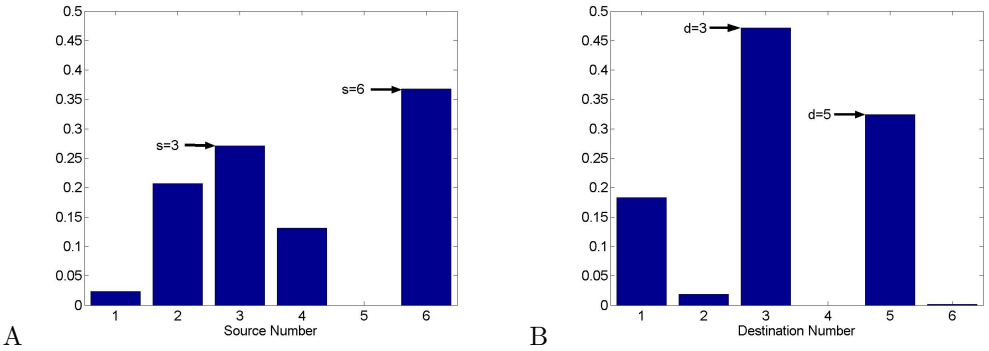


Figure 3.6: Marginal distributions ($\hat{P}(s_k|x_{1:K}, Q(\bar{A}) = b)$ in A and $\hat{P}(d_k|x_{1:K}, Q(\bar{A}) = b)$ in B) associated with the example endpoint posterior distribution shown in Fig. 3.5. The largest and second largest values of the marginal posteriors are indicated–it is these values that are used in computing the resolution ratios Λ_s and Λ_d, calculated as $\Lambda_s(k) = 0.58$ and $\Lambda_d(k) = 0.59$. It is clear in this example that the endpoints of this transmission (source number 6 and destination number 3) will be correctly estimated by the individual MAP estimates as well.

that graphs with 75% sensor coverage tend to have lower error values.

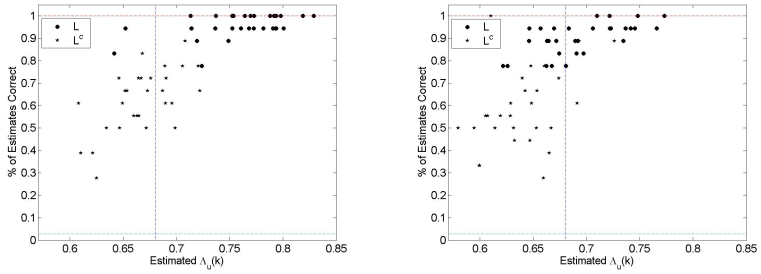
Plots of proportion of passive measurement endpoint estimates correct for a given set (L or L^c) versus the resolution ratio from Eq. (3.34) averaged over the corresponding set are given in Fig. 3.7. Plots are shown for joint estimates of u_k via the joint distribution as well as for individual estimates of s_k and d_k from the marginals. We observe an approximately linear relation between the proportion of correct estimates and the appropriate Λ ratio when the Λ ratio exceeds 0.68. In this regime, the Λ ratio might be used as a measure of confidence in the endpoint estimates. Also note that transmissions in set L tend to have higher Λ ratios (and are correct more often) than those in set L^c because it is the transmissions in set L that are used in training the probing sites. We see that marginalized MAP estimates are often better than joint MAP estimates. Marginalization certainly blurs the linear relation in the higher confidence regime. We also observe some degradation in the quality of the estimates when only 75% of the links are equipped with sensors; this is to be expected though. Recall that these results are obtained with completely random placement of sensors and random choices for the (s, d) pairs to use in the probing phase. These two factors will clearly affect the estimates of passive measurement endpoints, and therefore provide an interesting direction for future work.

3.4 Summary and Extensions

In this chapter, we have developed a methodology for estimating the endpoints of a transmission in a network using link-level transmission interceptions. The estimation is done using Monte Carlo simulation in a Bayesian framework. A semidefinite programming relaxation is used to generate logical network topology samples that approximately agree with linear prior information. It is possible to envision applica-

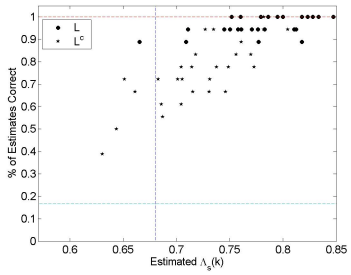
100% Avg	100% Exp	75% Avg	75% Exp
0.0211	0.0208	0.0142	0.0145
0.0229	0.0214	0.0110	0.0113
0.0195	0.0200	0.0108	0.0111
0.0247	0.0241	0.0119	0.0117
0.0178	0.0170	0.0146	0.0146
0.0247	0.0257	0.0152	0.0156
0.0189	0.0200	0.0139	0.0133
0.0247	0.0236	0.0154	0.0155
0.0230	0.0221	0.0143	0.0138
0.0222	0.0221	0.0121	0.0123
0.0243	0.0244	0.0135	0.0141
0.0241	0.0229	0.0118	0.0117
0.0217	0.0209	0.0139	0.0136
0.0190	0.0182	0.0125	0.0125
0.0248	0.0235	0.0127	0.0125
0.0195	0.0198	0.0131	0.0140
0.0257	0.0261	0.0133	0.0138
0.0180	0.0182	0.0147	0.0143
0.0236	0.0237	0.0137	0.0131
0.0214	0.0213	0.0122	0.0113
0.0250	0.0237	0.0112	0.0117
0.0253	0.0255	0.0128	0.0119
0.0191	0.0207	0.0135	0.0139
0.0186	0.0196	0.0150	0.0142
0.0200	0.0219	0.0142	0.0140
0.0272	0.0245	0.0119	0.0122
0.0212	0.0221	0.0110	0.0109
0.0188	0.0188	0.0112	0.0114
0.0244	0.0249	0.0128	0.0130
0.0269	0.0269	0.0116	0.0110

Table 3.1: Squared error values for compliance of samples with linear prior information $Q(\bar{A}) = b$. Sample topology errors (Avg) averaged over the 500 samples produced for each of the thirty graphs in the two simulation cases (100% coverage and 75% coverage) are given along with the theoretical expected value of the error (Exp). Once the elements of $\bar{A}$ are organized in the vector a, the normalized average sample error (Avg) is simply $\frac{1}{||Qe||^2+||b||^2}\frac{1}{M}\sum_m \|Q\hat{a}^m - b\|^2$ for the m^{th} sample $\hat{a}^m$ produced by the SDP rounding method. The normalized expected error (Exp) $\frac{1}{||Qe||^2+||b||^2}\mathbf{E}\left[\|Q\hat{a} - b\|^2\right]$ as derived in Eq. (3.19) is also given. Note that the bound in Eq. (3.25) assures (Exp) never exceeds $1 - \alpha \approx 0.12$. We see also that the graphs with 75% sensor coverage typically have lower squared error values.

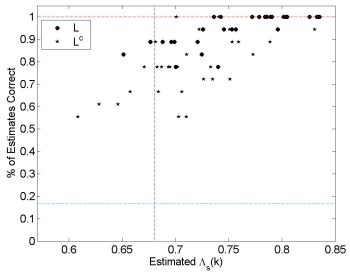


A: Λ_u, 100% coverage

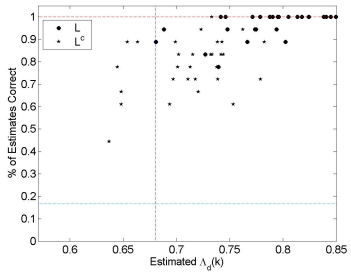
B: Λ_u, 75% coverage



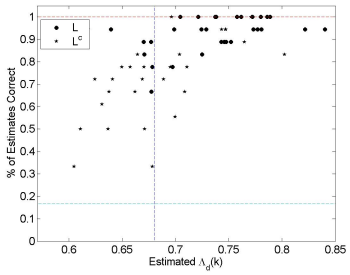
C: Λ_s, 100% coverage



D: Λ_s, 75% coverage



E: Λ_d, 100% coverage



F: Λ_d, 75% coverage

Figure 3.7: Plots of proportion of endpoint estimates correct for a given set (L or L^c) versus the resolution ratios of Eq. (3.34) averaged over the corresponding set for the two simulation cases: 100% sensor coverage in the first column and 75% sensor coverage in the second. Circles indicate averages over paths from set L and pentagrams indicate averages over paths from set L^c. The first row (Λ_u) is for joint MAP estimation of $u_k = (s_k, d_k)$ from joint distribution $\hat{P}(u_k|x_{1:K}, Q(\bar{A}) = b)$. The second row ($\Lambda_s$) is for individual estimation of s_k from marginal distribution $\hat{P}(s_k|x_{1:K}, Q(\bar{A}) = b)$. The third row ($\Lambda_d$) is for individual estimation of d_k from marginal distribution $\hat{P}(d_k|x_{1:K}, Q(\bar{A}) = b)$. Some reference lines are also plotted: a horizontal line indicating the chance line for randomly selecting endpoints (1/36 for joint estimation and 1/6 for individual estimation), and a vertical line at 0.68. Note that above $\Lambda(k) = 0.68$, an approximately linear behavior is observed. This behavior is somewhat washed out for the marginalized estimates, however marginalizing tends to increase the percent of correct estimates. It is not surprising that there appears to be some degradation in the quality of the estimates when only 75% of the links are equipped with sensors.

tions of the method in all sorts of networks, or systems with key features modeled by networks. We have displayed simulations of its utility on some random networks. We now discuss some extensions of the theory presented here and possibilities for future work on this problem.

It is possible to extend our algorithm for source/destination estimation to the cases of noisy sensors and sensor excitation due to multiple transmissions without much trouble. Consider first when the sensors are noisy: then the observed set of activated sensors y may not match the true set of sensors $\tilde{y}$ passed by a particular transmission. Suppose that each sensor $\gamma \in \Gamma$ has an associated miss probability $\alpha_m(\gamma) = P(\gamma \notin y | \gamma \in \tilde{y})$ and false alarm probability $\alpha_f(\gamma) = P(\gamma \in y | \gamma \notin \tilde{y})$. The probing mechanism then repeats the data transmission from σ_k to δ_k N times for each k. These N measurements are used to construct a maximum likelihood estimate $\hat{y}_k$ of each path $\tilde{y}_k$ according to the following model. Along the lines of a generalized likelihood approach, the measurement mechanism passes along the maximum likelihood path estimates for each $\tilde{y}_k$ for use in approximating the endpoint posterior. Note that we will likely have to settle for $N = 1$ for passive measurements.

Define the path indicator vector ν whose elements are given by $\nu(j) = I_y(\gamma_j)$ for all $j = 1, 2, \ldots |\Gamma|$ where $I_A : A \to \{0, 1\}$ is the usual indicator function. If we assume sensor errors are independent across paths and measurements, then the joint probability mass function of the N observed path vectors for a given source/destination pair ν_i is

$$\begin{aligned} &P\left(\nu_1, \nu_2, \ldots \nu_N | \tilde{\nu}\right) = \\ &\textstyle\prod_{i=1}^{N} \prod_{j=1}^{|\Gamma|} \alpha_m(\gamma_j)^{(1-\nu_i(j))\tilde{\nu}(j)} \beta_m(\gamma_j)^{\nu_i(j)\tilde{\nu}(j)} \\ &\alpha_f(\gamma_j)^{\nu_i(j)(1-\tilde{\nu}(j))} \beta_f(\gamma_j)^{(1-\nu_i(j))(1-\tilde{\nu}(j))} \end{aligned} \tag{3.37}$$

where $\beta_m(\gamma) \equiv 1 - \alpha_m(\gamma)$ and $\beta_f(\gamma) \equiv 1 - \alpha_f(\gamma)$. If we define the likelihood function

$L(\tilde{\nu})$ as the logarithm of the expression in Eq. (3.37), then it may be written explicitly as

$$
\begin{aligned}
& L(\tilde{\nu}) = \\
& \textstyle\sum_{j=1}^{|\Gamma|} \left(N \log \beta_f(\gamma_j) + \sum_{i=1}^{N} \nu_i(\gamma_j) \log \frac{\alpha_f(\gamma_j)}{\beta_f(\gamma_j)} \right) + \\
& \textstyle\sum_{j=1}^{|\Gamma|} \left(N \log \frac{\alpha_m(\gamma_j)}{\beta_f(\gamma_j)} + \sum_{i=1}^{N} \nu_i(\gamma_j) \log \frac{\beta_e(\gamma_j)}{\alpha_e(\gamma_j)} \right) \tilde{\nu}(\gamma_j)
\end{aligned} \tag{3.38}
$$

where $\alpha_e(\gamma) \equiv \alpha_m(\gamma)\alpha_f(\gamma)$ and $\beta_e(\gamma) \equiv \beta_m(\gamma)\beta_f(\gamma)$. Since only the second term in Eq. (3.38) depends on $\tilde{\nu}$ and $\tilde{\nu} \in \{0,1\}^{|\Gamma|}$, the maximum likelihood path estimate may be written quite compactly as

$$
\begin{aligned}
& \hat{y} = \\
& \left\{ \gamma_j \in \Gamma \,|\, N \log \tfrac{\alpha_m(\gamma_j)}{\beta_f(\gamma_j)} + \textstyle\sum_{i=1}^{N} \nu_i(j) \log \frac{\beta_e(\gamma_j)}{\alpha_e(\gamma_j)} \geq 0 \right\}
\end{aligned} \tag{3.39}
$$

As another extension, suppose that for passive measurements the activated sensor set y_k is due to transmissions passed between n source/destination pairs u_{ki} for $i = 1, 2, \ldots, n$ where n is known. The strategy here is to introduce a random variable η_k for each passive measurement that represents a partition of the activated sensor set y_k into sets y_{ki} for $i = 1, 2, \ldots, n$, where the sensors in each y_{ki} are activated in response to a single transmission. We may then split the single measurement y_k into n different passive measurements y_{ki} according to the value of the partition variable η_k and proceed with the previous theoretical development. In this case, the endpoint posterior of Eq. (3.1) becomes

$$
\begin{aligned}
& P(u_k | x_{1:K}, Q(\bar{A}) = b) = \\
& \mathbf{E}_{A,\rho_{1:K},\eta_{K_o:K}} \left[\tfrac{P(y_k|u_k,\rho_k,A)P_k(u_k)}{\sum_u P(y_k|u,\rho_k,A)P_k(u)} \,|\, x_{1:K}, Q(\bar{A}) = b \right]
\end{aligned} \tag{3.40}
$$

where we must now also take the expectation over partition variables $\eta_{K_o:K}$ of all passive measurements. The first step of the Monte Carlo sampling would then be to draw a partition variable for each passive measurement from some (presumably

available) distribution $P_k(\eta)$. Given the partition variable, appropriate orderings may be drawn and so on as before.

One can similarly account for the case of random linear prior information $Q(\bar{A}) = b$. Suppose that instead of being given a fixed operator Q and vector b, we are given a distribution on these $P(Q, b)$. This might occur, for example, when we know that the vertex degrees follow a power-law distribution [26]– in which case a distribution on b is induced. We must now also take the expectation over Q and b, so that the endpoint posterior becomes

$$P(u_k|x_{1:K}) = \mathbf{E}_{A,\rho_{1:K},Q,b}\left[\frac{P(y_k|u_k,\rho_k,A)P_k(u_k)}{\sum_u P(y_k|u,\rho_k,A)P_k(u)} \mid x_{1:K}\right] \tag{3.41}$$

A Monte Carlo approximation of Eq. (3.41) would therefore require drawing Q and b then proceeding as before. Unfortunately, a new SDP must be solved for every Q and b in order to produce topology samples A. If the SDP relaxation is not too large, this might be reasonable. If the size is prohibitive, one might approximate the expectation by selecting only a few of the most likely realizations of (Q, b) and solving the SDP for these. The distribution $P(Q, b)$ is then restricted to be nonzero only at elements of this preselected dictionary so that the Monte Carlo simulation selects those only those values for which we have already solved the SDP.

An interesting direction for future work would be to develop an adaptive probing scheme. It is obvious that the quality of endpoint estimates for suspect transmissions will depend on which endpoints were used in the probing phase. The idea here is to use the approximate endpoint posterior distributions to suggest additional active measurements that should be made in order to improve the estimates. One can hypothesize criteria for determining the new probing pairs. For example, nodes that tend to have similar posterior probabilities over several suspect paths might

be selected for probing so as to distinguish them more explicitly in the constraints. The question of efficient online implementation naturally arises in this context. A forgetting factor could be used in conjunction with existing topology and ordering samples so that an entirely new batch would not be required at each probing cycle.

CHAPTER IV

Online Methods for Network Endpoint Localization

4.1 Introduction

We present efficient, recursive techniques for estimating the source and destination (endpoints) of a suspect transmission through a network based on the activation pattern of sensors placed on network components. Our results lead to significant reduction in computational complexity and permit online tracking of possibly changing endpoint locations of a suspect message over time. Estimation is based on a hierarchical Bayesian model relating routing, tracking, and topological parameters. Estimates are derived and analyzed using a recursive expectation-maximization (EM) algorithm and semidefinite programming (SDP) methods. Under a complete lack of ordering information, the recursive and exact EM algorithms require a number of operations at each iteration that grow exponentially with the number of sensors activated by a given path. To cope with this problem, we present approximate methods based on permutation clustering that reduce the complexity to only quadratic growth in the number of activated sensors. Some ideas are also given for the design problem; we apply a recursive algorithm to control a multiarmed bandit model for online probe scheduling. Finally, we illustrate the effectiveness of the new methods through experiments involving Internet data.

The measurement apparatus for the system is identical to that described in [46]. It consists of a number of asynchronous sensors, denoted Γ, placed on some subset of elements (links or nodes) in a network. A sensor is activated, and its activation recorded, whenever the path of a data transmission is intercepted on the element where the sensor is situated. We suppose transmissions originate at some source nodes in a set Σ and terminate at some destination nodes in a set Δ, activating sensors in Γ along the way. The apparatus is illustrated on a sample network in Fig. 4.1. Individual transmissions produce measurements recorded at discrete time instances, and the subscript k is used throughout the chapter to index time. If multiple sensors are activated by a single transmission, they may not be capable of providing the precise order in which they were activated. In general, a probability distribution $P_k(\rho)$ on the possible orders of activation is observed for each measurement; here the argument $\rho \in \{1, 2, \ldots, \}$ is simply a natural number used to indicate a specific ordering of the sensors activated at time k. For example, a transmission with endpoints (σ_1, δ_1) in Fig. 4.1 might activate sensors $y_1 = \{\gamma_2, \gamma_3\}$–suppose this is the first measurement so $k = 1$. The ordering (γ_2, γ_3), corresponding to $\rho = 1$, might have probability $P_1(1) = \frac{3}{4}$, while the ordering (γ_3, γ_2), where $\rho = 2$, has probability $P_1(2) = \frac{1}{4}$. We are able to probe the network by scheduling a message to be passed from some source $\sigma \in \Sigma$ to some destination $\delta \in \Delta$ and observing the activated sensor set y and ordering distribution $P(\rho)$. Based on the results of our probing observations, we wish to determine the unknown endpoints (source and destination in $\Sigma \times \Delta$) of suspect observations, each consisting of an activated sensor set and ordering distribution.

A Monte Carlo method for endpoint estimation was developed in [46]. This approach averages over feasible sample topologies given a batch of measurements in

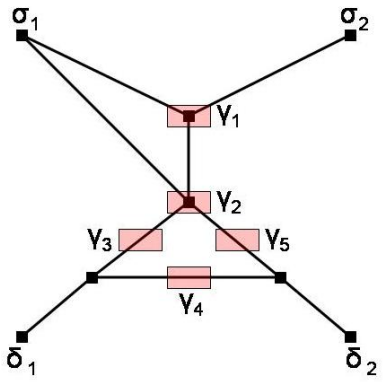


Figure 4.1: Diagram of the measurement apparatus on a sample network. Probing sites are sources $\Sigma = \{\sigma_1, \sigma_2\}$ and destinations $\Delta = \{\delta_1, \delta_2\}$. Sensors are $\Gamma = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5\}$. A box on a link or node represents a sensor that indicates when a transmission path intercepts that link/node. We see γ_1 and γ_2 monitor nodes while γ_3, γ_4, and γ_5 monitor links.

order to produce endpoint posteriors. It is not clear how one might recursively update posteriors produced in this fashion. We address the updating problem here. Fundamentally, the online model utilizes a generalization of the homogeneous Markovian routing assumption in [77]. We suppose that the next hop in a message's path depends only on its current position and its final destination. This model induces a set of routing parameters θ_{ij}^d that represent the probability of going from element i to element j given that the final destination is d. Since the ordering of activated sensors is uncertain, up to a probability distribution $P(\rho)$, the probabilty of any measured path under the Markovian assumption takes the form of a multinomial mixture distribution parameterized by θ. Endpoint posterior distributions then immediately follow from this model given plug-in estimates of the routing parameters $\hat{\theta}_{ij}^d$ and suspect endpoint priors $P(s, d)$. In order to avoid a growing memory problem, we use a recursive form of the EM algorithm [91] to update approximate MAP estimates of the routing parameters when new measurements are made. This is the first of three key approximations necessary to make our method practical for online implementation. The recursive EM method requires that we retain only an information state that

summarizes all past measurements, rather than the measurements themselves. We are able to prove, however, that the asymptotic estimates produced by the recursive EM are fixed points of an exact EM algorithm that uses all measurements directly.

Because of the multinomial form of the path likelihoods, it is analytically convenient to make use of Dirichlet priors on the routing parameters [89]. The Dirichlet priors are defined by the hyperparameters β_{ij} for all sources/destinations/sensors i, j. We can then track the endpoints of suspect transmissions by using the suspect observations to compute estimates of the hyperparameters in an empirical Bayes framework [82]. This scheme not only allows the use of suspect measurements to augment the probes in forming a more complete picture of routing in the network, but also localizes which elements of the network are being utilized by the suspects, and thereby tracks them. EM recursions, similar to those used for estimation of the routing parameters θ_{ij}^d, update approximate MAP estimates. Any prior information taking the form of linear equalities constraining the unknown network topology's adjacency matrix can be included through a Dirichlet hyperprior on the tracking parameters β_{ij}. Such a characterization is useful because common priors, including vertex degree information and necessary conditions for connectivity of certain network components, can be written as linear constraints on the logical topology's adjacency matrix. The hyperprior is parameterized by γ_{ij}, where these are estimated by averaging over approximately feasible topologies produced using a semidefinite programming (SDP) relaxation generated from the linear prior equalities. The computation of γ_{ij} is done only during an initialization step, so the burden of solving an SDP online is not an issue. This second key approximation allows us to include any topological information with a polynomial time algorithm. Analysis and performance guarantees for the SDP algorithm are presented in [46].

Our models lead to estimator update equations that involve sums over all activated sensor set orderings ρ. The number of such orderings grows exponentially with the number of activated sensors. We may have sufficient synchronization to rule out many of the possible permutations (i.e. $P_k(\rho) = 0$ for most values of ρ). Without such information, however, computing the necessary sums quickly becomes intractable when the number of activated sensors exceeds six or seven. This problem motivates our final crucial approximation, which is to form permutation clusters [16] and use these to compute the sums. The clusters are defined using a sort of augmented generating tree whose construction is driven by the particular estimate values appearing in the sum to be approximated. The tree is built to some depth and then truncated when either the approximation error is tolerable or the number of clusters is too large. In the case that the tree is not truncated, it represents the exact sum by enumerating every possible permutation. The idea of using generating trees for enumerating permutations was first proposed in [16]. It has more recently been applied to the enumeration of restricted permutations [99], and extended to allow for the enumeration of a wide variety of combinatorial objects [5]. Our technique of clustering the permutations is also similar to the separable operator approximations used in [8]. By grouping permutations into clusters, we are effectively decoupling many of the terms that appear in the full sum. This allows us to approximate the full sum with fewer terms that separate according to the clusters.

In addition to estimation and tracking, we can implement probe scheduling online using an algorithm developed for the nonstochastic multiarmed bandit [3]. The idea is to treat each source/destination pair as a different arm on a multiarmed bandit. The multiarmed bandit is a classic problem model used to capture the tradeoff between exploration and exploitation [83]. Imagine a slot machine with several arms, each

giving some unknown reward. The objective is to decide a strategy for pulling the arms so as to maximize your reward over time. The slot machine has a cost per play, so exploration in the form of trying different arms is costly; however if a single arm is played always, one might miss out on exploiting an arm with higher payoff. In our scenario, the reward associated with scheduling a probe between a specific pair is determined by the reduction in the entropy of suspect endpoint posterior distributions resulting from the probe. This sort of information gain criterion has found successful application to sensor management [40, 50]. Under this framework, we can directly apply the Exp3 algorithm of [3]. Exp3 draws the probing pair to be scheduled from a mixture distribution containing two components: a uniform component and a shaped component determined by normalizing some weights. The weights, in turn, are recursively updated in response to observed rewards. The two components of the mixture distribution reflect the explore/exploit tradeoff; the uniform component promotes even exploration, while the shaped component exploits the high payoff of certain arms. Several performance guarantees are proven for the algorithm in [3]. Along with the performance guarantees, the computational simplicity and recursive nature of the algorithm makes it very suitable for online scheduling.

As in [46], we utilize an abstract network model that does not appeal to any physical specifications. These techniques are therefore applicable to endpoint localization in a wide variety of networks, such as those describing partially observed technological, social, or biological structures [70]. Also, the methods may be adapted to produce an online topology inference scheme to address the problem considered in [77]. The method for permutation clustering tackles the general problem of reduced complexity approximations, and may well find applications beyond this present work.

The chapter is organized as follows. Section II presents the hierarchical Bayesian model used to explain observed measurements. Section III derives recursive, adaptive estimators for parameters appearing in the hierarchical model, while Section IV analyzes the convergence properties of these estimators. In Section V, we describe the permutation clustering method for approximating certain combinatorial sums that appear in the estimate update equations. Section VI discusses the application of an algorithm for control of the multiarmed bandit to the problem of online probe scheduling. A variety of tracking simulations that apply the new methods to real Internet data collected by Rabbat et. al. [77] are presented in Section VII. We conclude our discussion in Section VIII with a summary and propositions for future work.

4.2 Hierarchical Bayesian Model

The basis for our online estimation scheme is a hierarchical Bayesian model. A diagram illustrating the relationships among variables in the model is shown in Figure 4.3A. At the highest level, we have parameters γ_{ij} associated with characteristics (such as vertex degree and connectivity) of the logical topology of the network. The logical topology considers adjacency relationships among only those elements (vertices and edges) that are either monitored with a sensor or used as a probing site. For example, we cannot hope to pinpoint the position of a link in the original network that is not monitored by a sensor. We assume unmonitored elements are essentially 'short-circuited' in the logical network. The idea here is to assure two elements are logically adjacent even if they are physically separated by an element (or subgraph of elements) that is not monitored. An example logical topology is given in Figure 4.2 for the monitored network in Figure 4.1.

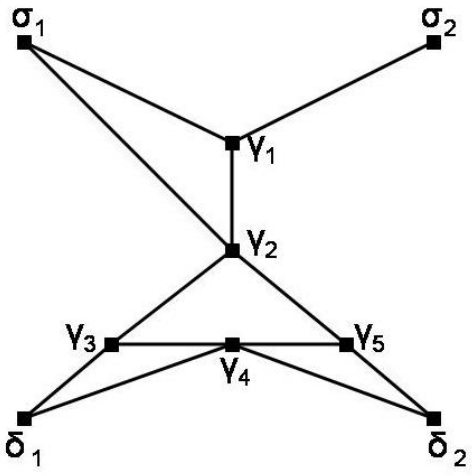


Figure 4.2: Example logical topology for the monitored network in Figure 4.1. The vertex set of the logical network consists of sensors $\Gamma = \{\gamma_i\}_{i=1}^{5}$ and probing sites $\Sigma = \{\sigma_1, \sigma_2\}$, $\Delta = \{\delta_1, \delta_2\}$. The edges summarize logical adjacencies among sensors and probing sites with any intervening unmonitored elements short-circuited.

The topology parameters serve as priors for the tracking parameters β_{ij}. The tracking parameters indicate the extent to which the suspects are utilizing specific parts of the network; they are therefore updated in response to new observed suspects. The routing parameters θ_{ij}^{d} appear next in the hierarchy. These are updated by the probing measurements, and serve as parameters in a controlled Markovian routing model for observed message paths. One might compare this model to the measurement model of [46], which is depicted in Figure 4.3B. This model explains observed message paths using only the logical topology A_{ij}, which is constrained by linear equalities in the same way that our γ_{ij} are constrained. The method in [46] processes a batch of data offline, so that there is no need for adaptation. Our model introduces routing and tracking parameters into the hierarchy in order to adaptively account for changes in network routing or suspect location. We will proceed to describe in detail each of the components in the model of Figure 4.3A from the bottom up.

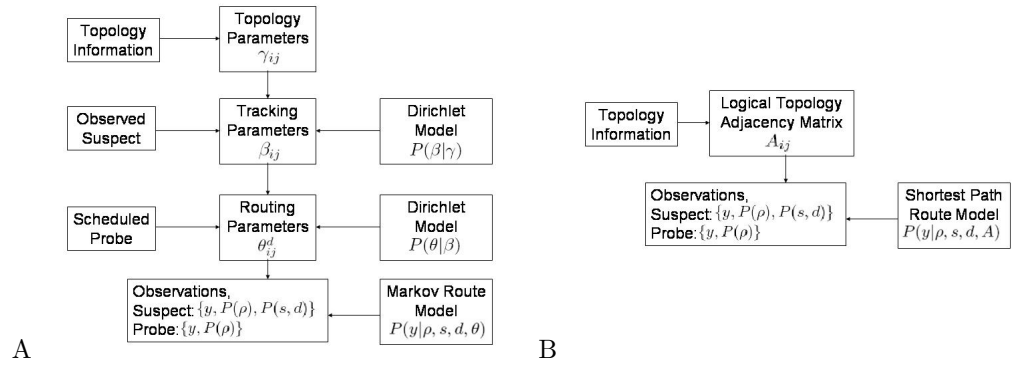


Figure 4.3: Diagram of the hierarchical Bayesian models. The model for our present online system is given in A, while the model of [46] for offline estimation is in B. Vertical arrows represent prior dependencies, while right arrows indicate data used in updating parameter estimates, and left arrows indicate the associated probability models. The model introduces routing and tracking parameters into the hierarchy in order to adaptively account for changes in network routing or suspect location. The method in [46] processes a batch of data offline, so there is no need for adaptation.

4.2.1 Controlled Markov Routing Model

We use a generalization of the Markovian routing model supposed in [77]. The basic assumption is that the next hop in a message's path through the network depends only on its current position and its final destination. Note that this is a fair assumption for many modern routing algorithms [90]. In contrast, [77] assumes the next hop in a path depends only on the current position of a message, irrespective of the final destination. We let θ^d_{ij} denote the probability that a message currently at element (source/sensor) i will go next to element (sensor/destination) j given that its final destination is d. Under this assumption, we can interpret each θ^d (for all i, j) as the transition matrix of some Markov chain. Indeed, one may view this as a controlled Markov model where d serves as the control [51]. The model in [77] utilizes a single transition matrix since it does not treat d as a control; note that although our model might be more realistic, it does require more parameters.

The Markov chain assumption implies the following path likelihood model for an

activated sensor set y given the ordering ρ and endpoints s, d:

$$\begin{aligned} P(y|\rho, s, d, \theta) &= \theta^d_{sy^1_\rho} \prod_{n=1}^{|y|-1} \theta^d_{y^n_\rho y^{n+1}_\rho} \theta^d_{y^{|y|}_\rho d} \\ &= \prod_{(i,j)\in\chi_\rho} \theta^d_{ij} \end{aligned} \tag{4.1}$$

where $y_\rho \equiv (y^1_\rho, y^2_\rho, \ldots, y^{|y|}_\rho)$ indicates the particular ordering ρ of the activated sensor set y and $\chi_\rho \equiv \{(s, y^1_\rho), (y^1_\rho, y^2_\rho), (y^2_\rho, y^3_\rho), \ldots (y^{|y|-1}_\rho, y^{|y|}_\rho), (y^{|y|}_\rho, d)\}$. From this model, we easily get the endpoint posterior distribution of a suspect measurement y as

$$P(s, d|y, \theta) = \frac{1}{\kappa} \sum_\rho P(y|\rho, s, d, \theta) P(\rho) P(s, d) \tag{4.2}$$

where $P(\rho)$ is the ordering distribution associated with the measurement, $P(s, d)$ is the endpoint prior, and κ is a normalization constant independent of ρ.

4.2.2 Dirichlet Priors

Because Eq. (4.1) is in the form of a multinomial distribution, and its parameters θ^d_i lie on the probability simplex for all d, i, the Dirichlet prior, which is conjugate to the multinomial distribution, provides an analytically tractable scheme for incorporating additional information about θ [89]. The prior is given by

$$P(\theta|\beta) = \frac{1}{\kappa'} \prod_{d=1}^{|\Delta|} \prod_{i=1}^{|\Gamma|+|\Sigma|} \prod_{j=1}^{|\Gamma|+1} (\theta^d_{ij})^{\beta_0 \beta_{ij}} \tag{4.3}$$

where conditional independence is assumed across transition matrices indexed by d and rows indexed by i. Although it appears that we have also assumed independence over columns j, there is in fact coupling over columns since all rows θ^d_i must satisfy $\sum_j \theta^d_{ij} = 1$. The tracking parameters β_{ij} in the prior are nonnegative and satisfy $\sum_j \beta_{ij} = 1$ for all i. Also in Eq. (4.3), we have a normalization constant κ' and a positive precision parameter β_0 that allows one to scale the strength of the prior.

We utilize the law of total probability as follows to derive a likelihood model for an ordered path given the tracking parameters β.

$$P(y|\rho, s, d, \beta) = \mathbf{E}\left[P(y|\rho, s, d, \theta) \mid \beta\right] \tag{4.4}$$

where the expectation is taken over θ with respect to the prior $P(\theta|\beta)$. In order to evaluate this expectation in closed form, we require that all paths be loopless i.e. $y_\rho^m \neq y_\rho^n$ for all $m \neq n$. This ensures that distinct terms $\{\theta_{ij}^d\}$ appearing in Eq. (4.1) are conditionally independent (given β) since they all come from different rows of the transition matrix θ^d (recall in defining the prior, we assumed conditional independence over rows). Applying conditonal independence and plugging in from Eq. (4.1) allows the expectation to be written as

$$\begin{aligned} P(y|\rho, s, d, \beta) &= \prod_{(i,j)\in\chi_\rho} \mathbf{E}\left[\theta_{ij}^d|\beta\right] \\ &= \prod_{(i,j)\in\chi_\rho} (1+\beta_0\beta_{ij})/(|\Gamma|+|\Delta|+\beta_0) \end{aligned} \tag{4.5}$$

where the second line follows from inserting the mean of the Dirichlet distribution in Eq. (4.3). This is the only result that requires a loopless path. One might still apply these techniques to paths with cycles, however it would then be necessary to compute higher order moments of the Dirichlet distribution and revise subsequent estimators.

As mentioned previously, each row of the tracking parameter matrix also lies in the probability simplex. Again given the multinomial-like product factorization of the likelihood in Eq. (4.5), it is convenient to assume a Dirichlet prior on these given by

$$P(\beta|\gamma) = \frac{1}{\kappa''} \prod_{i=1}^{|\Gamma|+|\Sigma|} \prod_{j=1}^{|\Gamma|+|\Delta|} (\beta_{ij})^{\gamma_0\gamma_{ij}} \tag{4.6}$$

Conditional independence (given γ) over rows is assumed as before. The topology parameters γ_{ij} define this prior, along with a positive scale factor γ_0. We may set the scale factor based on our confidence in the topological information.

Although the exact logical topology of the monitored network is unknown to us, we have available some prior information of the form $Q(A) = v$. Here, $A \in$

$\{0,1\}^{(|\Gamma|+|\Sigma|+|\Delta|)\times(|\Gamma|+|\Sigma|+|\Delta|)}$ is the adjacency matrix of the logical topology, Q is a linear operator, and v is a vector. Through appropriate choices of Q and v, it is possible to define various network priors including cliques, vertex degrees, or even some known portions of the topology. See [46] for some concrete examples of these. We define the topology parameter γ_{ij} as the probability that a logical connection exists between element i and element j given the prior information; i.e.

$$\gamma_{ij} = \mathbf{E}[A_{ij} \mid Q(A) = v] \tag{4.7}$$

In this way, one might also use the topology parameters, along with associated precision parameters β_0 and γ_0, to account for knowledge of stable network routing components.

In contrast to independence assumptions for the tracking and routing parameters, it is clear that the topology parameters might share complicated dependencies due to coupling of adjacency elements A_{ij} by the prior equalities. This observation not only strengthens our conditional independence assumptions made earlier, but also illustrates an advantage of the hierarchical Bayesian model. It is usually topological aspects of a network that induce dependencies among routes. We might assume independence of parameters related to routing, provided we condition on topology. Since the topology parameters are placed at the highest level of the Bayesian hierarchy, we are able to exploit independence to simplify computations at lower levels.

4.3 Parameter Estimation

We now proceed to derive estimators for the parameters introduced in the model of the previous section. It is useful, however, to first give a high level view of the flow of computations necessary for the online system. The system is first initialized by formulating and solving a semidefinite program using prior information about

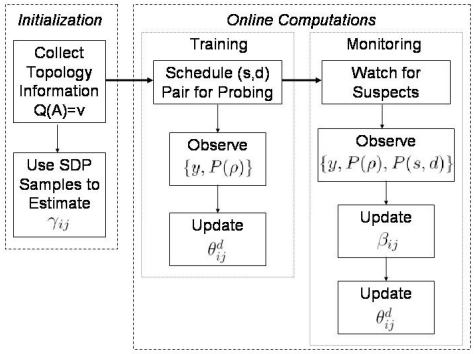


Figure 4.4: Operational diagram of the online system. Heavy horizontal arrows indicate transitions between stages of operation (initialization, training, and monitoring), while light vertical arrows indicate the flow of computation within each stage. The system is initialized by formulating and solving a semidefinite program (SDP) associated with the prior equality constraints $Q(A) = v$ on the logical adjacency matrix A. Once online operation commences, we have a training phase in which probes are scheduled and routing parameter estimates are recursively updated in response to probe observations. Next we monitor the network for suspect transmissions, and update tracking and routing parameter estimates whenever a suspect is observed. Refer to Figure 4.3 for parameter definitions and relations.

the network that is linear in the logical topology's adjacency matrix (such as vertex degrees). We average over samples produced using the rounding scheme of [46] in order to estimate the topology parameters. With these, we move into online operation of the system. For some initial training period, a probe of the network is made at each tick of the clock. A probe consists of sending a message between some known source and destination and observing the activated sensor set y and ordering distribution $P(\rho)$. We update routing parameters in response to the results of each new probe. After the training phase ends, we begin monitoring for suspect transmissions–i.e. transmissions whose source and destination are unknown. Each suspect measurement includes an activated sensor set y, ordering distribution $P(\rho)$, and prior over possible endpoints $P(s, d)$. When a suspect is observed, the tracking parameters are updated, which forces an update in the routing parameters. The best available estimates of the routing parameters may be used at any given time to build endpoint posteriors of observed suspects. An operational diagram of the system is given in Figure 4.4.

In the following, we will first define the precise estimation problems that we wish to solve. We then proceed to derive the estimators associated with each stage of the system: initialization (topology parameters), training (routing parameters), and monitoring (tracking parameters).

4.3.1 Estimation Objectives and Problem Statement

Assuming the probability of each edge A_{ij} is unknown, our goal in the initialization phase is to produce a suitable Monte Carlo approximation of the expectation in Eq. (4.7) in order to estimate the topology parameters. Once this is done, we move to online operation.

We desire estimators of the routing and tracking parameters to be recursive (so as to avoid growing memory problems) and scalable (so that online computation does not become intractable as the size of the problem increases). We also want estimators that adapt to changes in routing protocols and suspect locations. With that in mind, we choose the following penalized likelihood objective for estimation of the routing parameters.

$$\phi_k(\theta) = \sum_t a^{k-t} l_t(\theta) + \log P(\theta|\hat{\beta}(k)) \tag{4.8}$$

where $a \in [0, 1)$ is a forgetting factor, and $\hat{\beta}(k)$ is a plug-in estimate of the prior parameters β at time k. Note that here and throughout the chapter, k indexes the current clock tick. The log-likelihood is explicitly given by

$$l_t(\theta) = \log \left(\sum_\rho P(y_t|\rho, s_t, d_t, \theta) P_t(\rho) \right) \tag{4.9}$$

where $P(y_t|\rho, s_t, d_t, \theta)$ follows from the model in Eq. (4.1).

Note that the objective in Eq. (4.8) would be precisely the maximum a posteriori (MAP) objective under an i.i.d. measurement model if the forgetting factor was

unity. The introduction of the factor $a < 1$ is a common heuristic used in the design of adaptive algorithms to reduce the effect of old measurements on current parameter estimates [82]. One might set a using knowledge of dynamic routing in the network; e.g. the more quickly standard routes are expected to change in the network, the closer to zero a should be set. Alternatively, there exists a Dirichlet type generative model for measurements taken over time for which the objective in Eq. (4.8) yields precisely the MAP estimate. It is straightforward to write down such a model, so we omit it here.

The Dirichlet prior on θ serves to condition the routing parameters θ_{ij}^d by the tracking parameters β_{ij}. A reasonable way to estimate β is to make use of the observed suspect transmissions. If the suspect frequently utilizes the link from i to j, we would like β_{ij} to be closer to one. In this way, we fill in gaps in the probing measurements by making direct use of the suspects to form a more complete picture of routing in the network. Note also, that by taking account of the links being used by the suspects, we are essentially tracking their positions in the network as characterized by sensor activations. The estimates of β can be combined with the endpoint posterior distribution to provide additional information about the suspects' locations.

The estimation of the tracking parameters β is formalized through the use of empirical Bayes techniques [82]. We choose a similar sort of adaptive MAP objective for estimation of the tracking parameters.

$$\phi_k(\beta) = \sum_t b^{k-t} l_t(\beta) + \log P(\beta|\hat{\gamma}) \tag{4.10}$$

Again, a forgetting factor $b \in [0, 1)$ is used to discount old suspect measurements, and $\hat{\gamma}$ is the static plug-in estimate of the prior parameters γ. The log-likelihood is

given by

$$l_t(\beta) = \log\left(\sum_{s,d}\sum_{\rho} P(y_t|\rho, s, d, \beta)P_t(\rho)P_t(s, d)\right) \tag{4.11}$$

where $P(y_t|\rho, s, d, \beta)$ is the likelihood of some suspect measurement y_t given in Eq. (4.5). In order to allow for adaptation, one might tune b by knowledge of a suspect's motion through some network.

Ideally, our routing and tracking parameter estimates at time k, $\hat{\theta}(k)$ and $\hat{\beta}(k)$, would maximize the objective functions in Eqs. (4.8) and (4.10) respectively. The goal in online estimation is therefore to develop scalable, recursive algorithms to optimize these functions. We shall see that the key qualifiers 'scalable' and 'recursive' will necessitate certain approximations to be made. Our estimates will therefore only approximately optimize the selected objective functions. By plugging in routing parameter estimates, we can also compute the endpoint posterior distributions of suspect measurements as given in Eq. (4.2).

4.3.2 Initialization with Topology Information

If the edge probabilities $P(A_{ij} = 1)$ in the logical topology are unknown, except for some prior constraints on the logical adjacency matrix $Q(A) = v$, we must approximate the expectation of Eq. (4.7). Let $\{A^m\}_{m=1}^M$ be the sample adjacencies produced from a semidefinite program (SDP) formulated to approximately solve the prior equalities $Q(A) = v$ as described in [46]. We produce a Monte Carlo estimate of the expectation in Eq. (4.7) as follows

$$\hat{\gamma}_{ij} = \frac{1}{M}\sum_{m} A_{ij}^m \tag{4.12}$$

Note that solving an SDP is typically a very demanding computational task ($O(N^7)$ for a matrix of size $N \times N$ [57]). Fortunately, we need only solve the SDP once, during an offline initialization phase.

4.3.3 Online Routing Parameter Estimation

The form of the likelihood in Eq. (4.9) suggests the EM algorithm as a natural candidate for implementing the estimator [22]. Exact maximization of Eq. (4.8) via EM would require storing all past probing measurements. In order to avoid this growing memory problem, we utilize a recursive form of the EM algorithm described in [91] to update the maximum value of an approximation to Eq. (4.8). Recursive EM approximates the likelihood term $\sum_t a^{k-t} l_t(\theta)$ by $L_k(\theta)$, which is obtained recursively as follows:

$$L_k(\theta) = \mathbf{E}\left[\log P(y_k|\rho, s_k, d_k, \theta) \mid y_k, s_k, d_k, P_k(\rho), \hat{\theta}(k-1)\right] + aL_{k-1}(\theta) \tag{4.13}$$

where $L_0(\theta) = 0$ and $P(y_k|\rho, s_k, d_k, \theta)$ is taken as 1 if a probe is not scheduled at time k (in order to remain consistent with the likelihood term in Eq. (4.8)). Evaluating the expectation in Eq. (4.13) over orderings ρ and regrouping terms yields

$$L_k(\theta) = \sum_{d,i,j} c_{ij}^d(\hat{\theta}(k-1); k) \log \theta_{ij}^d + aL_{k-1}(\theta) \tag{4.14}$$

with $c_{ij}^d(\theta; k)$ given by the following for $d = d_k$:

$$c_{ij}^d(\theta; k) = \frac{\sum_{\rho \mid (i,j) \in \chi_{k,\rho}} P(y_k|\rho, s_k, d_k, \theta) P_k(\rho)}{\sum_\rho P(y_k|\rho, s_k, d_k, \theta) P_k(\rho)} \tag{4.15}$$

We have $c_{ij}^d(\theta; k) = 0$ if $d \neq d_k$; also $c_{ij}^d(\theta; k) = 0$ for all d if a probe is not scheduled at time k. If we define the recursion for $\bar{c}(k)$ as

$$\bar{c}_{ij}^d(k) = c_{ij}^d(\hat{\theta}(k-1); k) + a\bar{c}_{ij}^d(k-1) \tag{4.16}$$

with $\bar{c}_{ij}^d(0) = 0$ for all d, i, j, then we can express the function $L_k(\theta)$ simply as

$$L_k(\theta) = \sum_{d,i,j} \bar{c}_{ij}^d(k) \log \theta_{ij}^d \tag{4.17}$$

The routing parameter estimates at time k are then given by

$$\begin{aligned}\hat{\theta}(k) &= \underset{\theta|\sum_j \theta_{ij}^d=1\forall d,i}{\arg\max} \quad \tilde{\phi}_k(\theta) \\ &= \underset{\theta|\sum_j \theta_{ij}^d=1\forall d,i}{\arg\max} \quad \sum_{d,i,j}\left(\bar{c}_{ij}^d(k)+\beta_0\hat{\beta}_{ij}(k)\right)\log\theta_{ij}^d\end{aligned} \tag{4.18}$$

A simple application of the KKT conditions to this concave maximization gives the following routing parameter estimates:

$$\hat{\theta}_{ij}^d(k) = \frac{\bar{c}_{ij}^d(k)+\beta_0\hat{\beta}_{ij}(k)}{\sum_l \bar{c}_{il}^d(k)+\beta_0\hat{\beta}_{il}(k)} \tag{4.19}$$

Eqs. (4.16) and (4.19) define the recursive routing parameter estimator. Note that these parameter estimates, although derived from a stochastic routing model, will indeed converge to a deterministic route if for a given element i we always observe a transition to element j_* when the destination is d. The forgetting factors $a, b < 1$ ensure that the estimates $\hat{\theta}_{ij}^d$ will be driven to some minimal value (depending on γ_{ij}, γ_0, and β_0) for all $j \neq j_*$ with almost all of the mass of $\hat{\theta}_i^d$ concentrated on $\hat{\theta}_{ij_*}^d$

4.3.4 Online Tracking Parameter Estimation

Comparing Eqs. (4.10) and (4.8) indicate tracking parameter estimation problem is almost identical to routing parameter estimation. The only fundamental difference in computation, is that a sum over source/destination pairs also appears inside the logarithm of Eq. (4.11). This is a consequence of our lack of knowledge of suspect endpoints. We again apply the recursive EM approximation for the likelihood term in the objective:

$$L_k(\beta) = \mathbf{E}\left[\log P(y_k|\rho,s,d,\beta) \mid y_k, P_k(s,d), P_k(\rho), \hat{\beta}(k-1)\right] + bL_{k-1}(\beta) \tag{4.20}$$

where $L_0(\beta) = 0$ and $P(y_k|\rho,s,d,\beta)$ is taken as 1 if a suspect is not observed at time k. After evaluating the expectation over ordering ρ and endpoints s, d, we can write

$L_k(\beta)$ as

$$L_k(\beta) = \sum_{i,j} \bar{g}_{ij}(k) \log(1 + \beta_0 \beta_{ij}) \tag{4.21}$$

with $\bar{g}(k)$ defined recursively as

$$\bar{g}_{ij}(k) = g_{ij}(\hat{\beta}(k-1); k) + b\bar{g}_{ij}(k-1) \tag{4.22}$$

where $\bar{g}_{ij}(0) = 0$ for all i, j. The factor depending on the new measurement $g_{ij}(\beta; k)$ is given by

$$g_{ij}(\beta; k) = \frac{\sum_{\rho,s,d \mid (i,j) \in x_{k,\rho}} P(y_k|\rho, s, d, \beta) P_k(\rho) P_k(s, d)}{\sum_{\rho,s,d} P(y_k|\rho, s, d, \beta) P_k(\rho) P_k(s, d)} \tag{4.23}$$

with $g_{ij}(\beta; k)$ taken as zero if a suspect is not observed at time k.

We replace the likelihood term $\sum_t b^{k-t} l_t(\beta)$ in Eq. (4.10) with $L_k(\beta)$ from Eq. (4.21) to arrive at the following expression for $\hat{\beta}(k)$

$$\begin{aligned} \hat{\beta}(k) &= \underset{\beta \mid \sum_j \beta_{ij} = 1 \forall i}{\arg\max} \quad \tilde{\phi}_k(\beta) \\ &= \underset{\beta \mid \sum_j \beta_{ij} = 1 \forall i}{\arg\max} \quad \sum_{i,j} \bar{g}_{ij}(k) \log(1 + \beta_0 \beta_{ij}) + \gamma_0 \hat{\gamma}_{ij} \log \beta_{ij} \end{aligned} \tag{4.24}$$

If one attempts to apply the KKT conditions as before, a system of quadratic equations results. We encounter this problem because of the sum inside the first logarithm. The familiar structure suggests a generalized EM framework wherein another EM iteration is used to increase the likelihood as an alternative to solving the quadratic system. We add $\bar{g}_{ij} \log \frac{2}{\beta_0 + 2}$ to the objective and combine with the first term to give

$$\hat{\beta}(k) = \underset{\beta \mid \sum_j \beta_{ij} = 1 \forall i}{\arg\max} \quad \sum_{i,j} \bar{g}_{ij}(k) \log \left(\frac{2}{\beta_0 + 2} + \frac{\beta_0}{\beta_0 + 2} 2\beta_{ij} \right) + \gamma_0 \hat{\gamma}_{ij} \log \beta_{ij} \tag{4.25}$$

We now recognize the first term as the logartihm of a uniform and linear mixture distribution. Applying EM to Eq. (4.25) in the standard way gives the operator f^k, defined component-wise as

$$f_{ij}^k(\beta) = \frac{\bar{g}_{ij}(k) \frac{\beta_0 \beta_{ij}}{1 + \beta_0 \beta_{ij}} + \gamma_0 \hat{\gamma}_{ij}}{\sum_l \bar{g}_{il}(k) \frac{\beta_0 \beta_{il}}{1 + \beta_0 \beta_{il}} + \gamma_0 \hat{\gamma}_{il}} \tag{4.26}$$

Action	Online Computation
Probe Scheduled	1. $\bar{c}_{ij}^{d}(k) = \frac{\sum_{\rho \mid (i,j)\in x_{k,\rho}} P(y_k\mid\rho,s_k,d_k,\hat{\theta}(k-1))P_k(\rho)}{\sum_{\rho} P(y_k\mid\rho,s_k,d_k,\hat{\theta}(k-1))P_k(\rho)} + a\bar{c}_{ij}^{d}(k-1)$ 2. $\hat{\theta}_{ij}^{d}(k) = \frac{\bar{c}_{ij}^{d}(k)+\beta_0\hat{\beta}_{ij}(k)}{\sum_{j'}\bar{c}_{ij'}^{d}(k)+\beta_0\hat{\beta}_{ij'}(k)}$
Suspect Observed	1. $\bar{g}_{ij}(k) = \frac{\sum_{\rho,s,d \mid (i,j)\in x_{k,\rho}} P(y_k\mid\rho,s,d,\hat{\beta}(k-1))P_k(\rho)P_k(s,d)}{\sum_{\rho,s,d} P(y_k\mid\rho,s,d,\hat{\beta}(k-1))P_k(\rho)P_k(s,d)} + b\bar{g}_{ij}(k-1)$ 2. $f_{ij}^{k}(\beta) = \frac{\bar{g}_{ij}(k)\frac{\beta_0\beta_{ij}}{1+\beta_0\beta_{ij}}+\gamma_0\hat{\gamma}_{ij}}{\sum_{j'}\bar{g}_{ij'}(k)\frac{\beta_0\beta_{ij'}}{1+\beta_0\beta_{ij'}}+\gamma_0\hat{\gamma}_{ij'}}$ 3. $\hat{\beta}(k) = f^k \circ f^k \circ \ldots f^k(\hat{\beta}(k-1)) = (f^k)^N(\hat{\beta}(k-1))$ 4. $\hat{\theta}_{ij}^{d}(k) = \frac{\bar{c}_{ij}^{d}(k)+\beta_0\hat{\beta}_{ij}(k)}{\sum_{j'}\bar{c}_{ij'}^{d}(k)+\beta_0\hat{\beta}_{ij'}(k)}$

Table 4.1: Summary of online computations.

The new estimate $\hat{\beta}(k)$ is the fixed point of the operator f^k. Provided $\hat{\gamma}_{ij} > 0$ for all i, j, the optimization in Eq. (4.25) is strictly concave. Since the Q-function from which Eq. (4.26) is derived is also continuous in both arguements, it follows that EM will converge to the unique global maximum [28]. Thus the fixed point of f^k is unique. In practice, we can obtain $\hat{\beta}(k)$ by initializing (with perhaps $\hat{\beta}(k-1)$) and then successively applying the operator f^k until $||(f^k)^N(\beta) - (f^k)^{N-1}(\beta)|| < \epsilon$ for some tolerance ϵ. So that

$$\hat{\beta}(k) = f^k \circ f^k \circ \ldots f^k(\hat{\beta}(k-1)) = (f^k)^N(\hat{\beta}(k-1)) \tag{4.27}$$

where N is chosen large enough to satisfy a desired tolerance ϵ. In practice, we have observed that this internal EM iteration converges very quickly; our simulations indicated an N value of 2 or 3 was typically sufficient to obtain convergence with a tolerance of $\epsilon = 10^{-8}$. The recursions in Eqs. (4.22) and (4.27) define the tracking parameter estimator. Table 4.1 provides a summary of all online computations.

4.4 Convergence Analysis

An asymptotic analysis of the unconstrained recursive EM algorithm is presented in [91] by using a quadratic expansion of the likelihood to relate the algorithm to

a stochastic approximation method. When constraints are present, the mathematical form of the expansion's optimum no longer reveals an obvious mapping to the stochastic approximation. We will argue convergence of our algorithms directly, instead of attempting to derive such a mapping. We also show that the recursive approximation produces asymptotic estimates that are fixed points of the EM algorithm applied to the exact objective for MAP estimation.

Before beginning, note that the quantities we consider are in fact random variables, so that all equalities or inequalities hold with probability one. Consider first the sequences $\bar{g}(k)$ and $\bar{c}(k)$ as defined in Eqs. (4.22) and (4.16) respectively.

Lemma IV.1. *The sequence $\bar{g}(k)$ converges to some limit $\bar{g}(\infty)$ as $k \to \infty$. Similarly, $\bar{c}(k) \to \bar{c}(\infty)$ as $k \to \infty$.*

Proof. The recursive expression in Eq. (4.22) can be written in closed form as follows

$$\bar{g}_{ij}(k) = \sum_{t=1}^{k} b^{k-t} g_{ij}(\hat{\beta}(t-1); t) \tag{4.28}$$

Now from the definition in Eq. (4.23), we see that $0 \leq g_{ij}(\beta; k) \leq 1$ for all k because all involved probabilities are nonnegative and every term in the numerator sum also appears in the denominator sum. It follows that $\bar{g}_{ij}(k)$ is a monotone nondecreasing sequence. Furthermore, we have

$$\begin{aligned} \bar{g}_{ij}(k) &\leq \textstyle\sum_{t=1}^{k} b^{k-t} \\ &= \tfrac{1-b^k}{1-b} \\ &\leq \tfrac{1}{1-b} \end{aligned} \tag{4.29}$$

since $b < 1$. Thus $\bar{g}_{ij}(k)$ is also bounded from above for all k. It follows that the sequence must converge to some value $\bar{g}_{ij}(\infty)$ as $k \to \infty$. We have thus established convergence of each i, j component; we therefore have $\bar{g}(k) \to \bar{g}(\infty)$. An identical argument establishes $\bar{c}(k) \to \bar{c}(\infty)$ as $k \to \infty$. □

The parameter estimates as defined in Eqs. (4.18) and (4.24) are the optimal values of strictly concave functions. Given that $\bar{c}$ and $\bar{g}$ define these functions and converge, we argue convergence of the estimates through uniform convergence of the functions they optimize. The following lemma is key to this argument.

Lemma IV.2. *Let $\{f_k\}$ and f_∞ be strongly concave functions over a compact, convex set C such that f_k converges to f_∞ uniformly over C as $k \to \infty$. If x_k denotes the maximum of f_k over C for all $k \in \{1, 2, \ldots \infty\}$, then x_k exists and is unique for all k, and furthermore $x_k \to x_\infty$ as $k \to \infty$.*

Proof. Since each function f_k is strongly concave, we have immediately that its optimizer over a compact, convex set exists and is unique. Now suppose x_k does not converge to x_∞, so there exists $\epsilon_* > 0$ such that $||x_k - x_\infty|| \geq \epsilon_*$ for all $k < \infty$. A Taylor expansion of f_k about the maximum gives

$$f_k(x_\infty) = f_k(x_k) + \nabla f_k(x_k)^T(x_\infty - x_k) + \frac{1}{2}(x_\infty - x_k)^T \nabla^2 f_k(z)(x_\infty - x_k) \tag{4.30}$$

where $z = \alpha x_k + (1-\alpha)x_\infty \in C$ for some $\alpha \in [0, 1]$. Optimality of x_k ensures $-\nabla f_k(x_k)^T(x_\infty - x_k) \geq 0$, and strong concavity of f_k implies there is some $m > 0$ such that $-\nabla^2 f_k(z) \succeq mI$ for all k [10]. We therefore rearrange Eq. (4.30) and apply these inequalities to arrive at

$$f_k(x_k) - f_k(x_\infty) \geq \frac{m}{2}\epsilon_*^2 \tag{4.31}$$

for all $k < \infty$.

Now uniform convergence of f_k to f_∞ ensures that for all $\epsilon > 0$, there is some $n(\epsilon)$ such that $|f_k(x) - f_\infty(x)| < \epsilon$ for all $k \geq n(\epsilon)$ and any $x \in C$. Consider any index k_* satisfying $k_* \geq n\left(\frac{m}{8}\epsilon_*^2\right)$; uniform convergence gives

$$\begin{aligned} f_{k_*}(x_{k_*}) &< f_\infty(x_{k_*}) + \tfrac{m}{8}\epsilon_*^2 \\ f_{k_*}(x_\infty) &> f_\infty(x_\infty) - \tfrac{m}{8}\epsilon_*^2 \end{aligned} \tag{4.32}$$

But the inequality in (4.31) must hold for all k, so we may substitute the inequalities from (4.32) into (4.31) to obtain

$$f_\infty(x_{k_*}) - f_\infty(x_\infty) \geq \frac{m}{4}\epsilon_*^2 \tag{4.33}$$

This contradicts the assumption that x_∞ is the unique maximizer of f_∞ over C. $\square$

In order to apply Lemma IV.2, we must decide on a compact, convex set C over which uniform convergence holds. Because of the $\log(\theta_{ij}^d)$ term in Eq. (4.18), it is convenient to bound the routing parameter estimates away from zero. Note that this can be achieved by including a sample adjacency matrix consisting of all ones into the sum of Eq. (4.12); the extra sample would ensure positivity while having a negligible effect on the topology parameter estimates, provided M is sufficiently large. Indeed, we have $\hat{\gamma}_{ij} \in [1/M, 1]$ for all i, j. This filters down to $\hat{\beta}(k)$ and $\hat{\theta}(k)$ through Eqs. (4.19) and (4.26). One can easily verify that $\hat{\beta}(k) \in C_\beta$ and $\hat{\theta}(k) \in C_\theta$ for all k, where C_β and C_θ are compact, convex sets defined by

(4.34)

$$C_\beta \equiv \left\{ \beta \in \left[\tfrac{\gamma_0(1-b)}{M(|\Gamma|+|\Delta|)(1+\gamma_0(1-b))}, 1 \right]^{(|\Gamma|+|\Sigma|)\times(|\Gamma|+|\Delta|)} \mid \textstyle\sum_j \beta_{ij} = 1 \forall i \right\}$$

$$C_\theta \equiv \left\{ \theta \in \left[\tfrac{\gamma_0\beta_0(1-b)(1-a)}{M(|\Gamma|+|\Delta|)(|\Gamma|+1)(1+\gamma_0(1-b))(1+\beta_0(1-a))}, 1 \right]^{(|\Gamma|+|\Sigma|)\times(|\Gamma|+1)\times|\Delta|} \mid \textstyle\sum_j \theta_{ij}^d = 1 \forall d, i \right\}$$

Enforcing positivity in this fashion allows us to ignore the inequality constraints in deriving the expressions in Eqs. (4.19) and (4.26), since they automatically satisfy the bounds by design.

This leads to the primary convergence theorem below.

Theorem IV.3. *The tracking and routing parameter estimates converge as $k \to \infty$; that is $\hat{\beta}(k) \to \hat{\beta}(\infty)$ and $\hat{\theta}(k) \to \hat{\theta}(\infty)$.*

Proof. We first show uniform convergence of $\tilde{\phi}_k(\beta)$ as defined in Eq. (4.24) to

$$\tilde{\phi}_\infty(\beta) \equiv \sum_{ij} \bar{g}_{ij}(\infty)\log(1+\beta_0\beta_{ij}) + \gamma_0\hat{\gamma}_{ij}\log\beta_{ij} \tag{4.35}$$

Lemma IV.1 implies that for all $\epsilon > 0$, there is some $n_g(\epsilon)$ such that $||\bar{g}(k) - \bar{g}(\infty)|| < \epsilon$ for all $k \geq n_g(\epsilon)$. In order to show uniform convergence, we take $n_1^* \equiv n_g\left(\frac{\epsilon}{(|\Gamma|+|\Sigma|)(|\Gamma|+|\Delta|)\log(1+\beta_0)}\right)$ for any given $\epsilon > 0$. For any $k \geq n_1^*$ we have

$$\begin{aligned}
\left|\tilde{\phi}_k(\beta) - \tilde{\phi}_\infty(\beta)\right| &= \left|\textstyle\sum_{i,j}\log(1+\beta_0\beta_{ij})(\bar{g}_{ij}(k) - \bar{g}_{ij}(\infty))\right| \\
&\leq \textstyle\sum_{i,j}|\log(1+\beta_0\beta_{ij})||\bar{g}_{ij}(k) - \bar{g}_{ij}(\infty)| \\
&< \tfrac{\epsilon}{(|\Gamma|+|\Sigma|)(|\Gamma|+|\Delta|)\log(1+\beta_0)}\textstyle\sum_{i,j}|\log(1+\beta_0\beta_{ij})| \\
&\leq \epsilon
\end{aligned} \tag{4.36}$$

where $\beta \in C_\beta$. Thus we have uniform convergence of $\tilde{\phi}_k(\beta)$ to $\tilde{\phi}_\infty(\beta)$ over C_β. Since $\hat{\gamma}_{ij} \geq 1/M$ for all i, j, the functions $\tilde{\phi}_k(\beta)$ and $\tilde{\phi}_\infty(\beta)$ are strongly concave over C_β; so Lemma IV.2 immediately gives convergence of $\hat{\beta}(k)$ to $\hat{\beta}(\infty)$ (the maximum value of $\tilde{\phi}_\infty(\beta)$).

In a similar fashion, we show uniform convergence of $\tilde{\phi}_k(\theta)$ as defined in Eq. (4.18) to

$$\tilde{\phi}_\infty(\theta) \equiv \sum_{d,i,j}\left(\bar{c}_{ij}^d(\infty) + \beta_0\hat{\beta}_{ij}(\infty)\right)\log\theta_{ij}^d \tag{4.37}$$

Lemma IV.1 ensures that for all $\epsilon > 0$, there is some $n_c(\epsilon)$ such that $||\bar{c}(k) - \bar{c}(\infty)|| < \epsilon$ for all $k \geq n_c(\epsilon)$. And let $n_\beta(\epsilon)$ ensure $||\hat{\beta}(k) - \hat{\beta}(\infty)|| < \epsilon$ for all $k \geq n_\beta(\epsilon)$. To show uniform convergence, take $n_2^* \equiv \max\left\{n_c\left(\frac{\epsilon}{2\kappa|\log\theta_{min}|}\right), n_\beta\left(\frac{\epsilon}{2\beta_0\kappa|\log\theta_{min}|}\right)\right\}$ where $\kappa \equiv |\Delta|(|\Gamma|+|\Sigma|)(|\Gamma|+1)$ and θ_{min} is the lower bound in C_θ of Eq. (4.34). We then have for any $k \geq n_2^*$

$$\begin{aligned}
\left|\tilde{\phi}_k(\theta) - \tilde{\phi}_\infty(\theta)\right| &\leq \textstyle\sum_{d,i,j}|\log\theta_{ij}^d|\left(|\bar{c}_{ij}^d(k) - \bar{c}_{ij}^d(\infty)| + \beta_0|\hat{\beta}_{ij}(k) - \hat{\beta}_{ij}(\infty)|\right) \\
&< \epsilon
\end{aligned} \tag{4.38}$$

Again the functions are strongly concave over C_θ since $\hat{\beta}_{ij}(k) > 0$; we therefore apply Lemma IV.2 to give convergence of $\hat{\theta}(k)$ to the maximum value of $\tilde{\phi}_\infty(\theta)$, denoted $\hat{\theta}(\infty)$. □

We can use the convergence results just established to analyze the relationship between the recursive approximation and the exact EM algorithm for large k. Before proceeding, we establish a useful lemma about the limit points of the sequences $\bar{g}(k)$ and $\bar{c}(k)$.

Lemma IV.4. *The limit points $\bar{g}(\infty)$ and $\bar{c}(\infty)$ alluded to in Lemma IV.1 are given explicitly by*

$$\begin{aligned} \bar{g}_{ij}(\infty) &= \lim_{k\to\infty} \sum_{t=1}^{k} b^{k-t} g_{ij}(\hat{\beta}(\infty); t) \\ \bar{c}^d_{ij}(\infty) &= \lim_{k\to\infty} \sum_{t=1}^{k} a^{k-t} c^d_{ij}(\hat{\theta}(\infty); t) \end{aligned} \tag{4.39}$$

for all d, i, j.

Proof. It is useful to first establish Lipschitz continuity of the functions $g_{ij}(\beta; k)$ and $c^d_{ij}(\theta; k)$ over C_β and C_θ respectively. The derivative of $g_{ij}(\beta; k)$ as in Eq. (4.23) satisfies

(4.40)

$$\begin{aligned} \left\| \tfrac{\partial g_{ij}}{\partial \beta}(\beta; k) \right\| &\leq 2\left((|\Gamma| + |\Sigma|)(|\Gamma| + |\Delta|)\right)^{1/2} \left(\textstyle\sum_{\rho,s,d} P(y_k|\rho, s, d, \beta) P_k(\rho) P_k(s, d) \right)^{-2} \\ &\leq 2\left((|\Gamma| + |\Sigma|)(|\Gamma| + |\Delta|)\right)^{1/2} (|\Gamma| + |\Delta| + \beta_0)^{2(|\Gamma|+1)} \\ &= L_g \end{aligned}$$

where the second line follows from Eq. (4.5), with $\beta \in C_\beta$. We therefore have that L_g is a Lipschitz constant over C_β independent of k and i, j [48]. In a similar fashion, one can establish a Lipschitz constant for $c^d_{ij}(\theta; k)$ over C_θ as $L_c = 2\kappa^{1/2}\theta_{min}^{-2(|\Gamma|+1)}$ where θ_{min} and κ are as defined in the proof of Theorem IV.3.

We proceed now with the main result. By Theorem IV.3 $\hat{\beta}(k)$ converges, and Lipschitz continuity of $g_{ij}(\beta; t)$ implies that for all $\epsilon > 0$ there is some $n(\epsilon)$ such that

$|g_{ij}(\hat{\beta}(k-1);t) - g_{ij}(\hat{\beta}(\infty);t)| < L_g\epsilon$ for all $k \geq n(\epsilon)$. For any given $\epsilon > 0$, take $n^* \equiv n\left(\frac{(1-b)\epsilon}{2L_g}\right) + \max\left\{1, \frac{\log((1-b)\epsilon/2)}{\log b} - 1\right\}$. We then have for all $k \geq n^*$

(4.41)

$$\begin{aligned} |\bar{g}_{ij}(k) - \textstyle\sum_{t=1}^{k} b^{k-t} g_{ij}(\hat{\beta}(\infty);t)| &\leq \textstyle\sum_{t=1}^{k} b^{k-t} |g_{ij}(\hat{\beta}(t-1);t) - g_{ij}(\hat{\beta}(\infty);t)| \\ &< \tfrac{(1-b)\epsilon}{2} \textstyle\sum_{t=n(\frac{(1-b)\epsilon}{2L_g})}^{k} b^{k-t} + \sum_{t=1}^{n(\frac{(1-b)\epsilon}{2L_g})-1} b^{k-t} \\ &< \tfrac{\epsilon}{2} + \tfrac{b^{k-n((1-b)\epsilon/(2L_g))+1}}{1-b} \\ &< \epsilon \end{aligned}$$

where the second term in the second line follows because $g_{ij}(\beta;t) \in [0,1]$ for all β, t. The argument for $\bar{c}_{ij}^{d}(\infty)$ is identical. □

Suppose we apply standard EM to optimize the tracking parameter objective $\phi_k(\beta)$ as in Eq. (4.10). Performing the E step averages over orderings and endpoints of each individual measurement and results in the following Q-function:

$$Q_k(\beta|\tilde{\beta}) = \sum_{i,j} \left(\gamma_0 \hat{\gamma}_{ij} \log \beta_{ij} + \log(1 + \beta_0 \beta_{ij}) \sum_{t=1}^{k} b^{k-t} g_{ij}(\tilde{\beta};t) \right) \tag{4.42}$$

If we wish to optimize the routing parameter objective $\phi_k(\theta)$ in Eq. (4.8) using exact EM, the E step averages over orderings only and gives a similar Q-function.

$$Q_k(\theta|\tilde{\theta}) = \sum_{d,i,j} \log \theta_{ij}^{d} \left(\beta_0 \hat{\beta}_{ij}(k) + \sum_{t=1}^{k} a^{k-t} c_{ij}^{d}(\tilde{\theta};t) \right) \tag{4.43}$$

Note that each measurement defines $g(\beta;t)$ or $c(\theta;t)$ for a single clock tick t. Thus we require all past measurements in order to compute the Q functions. The EM algorithm then proceeds to iteratively maximize the Q functions until a fixed point is reached. The recursive approximation maintains only a summary of the past measurements in $\bar{g}$ and $\bar{c}$. The following theorem shows that the asymptotic estimates produced by the recursive approximation will in fact be fixed points of the exact EM algorithm as $k \to \infty$.

Theorem IV.5. *If $\beta_Q(k)$ and $\theta_Q(k)$ denote the maximizers of $Q_k(\beta|\hat{\beta}(\infty))$ over C_β and $Q_k(\theta|\hat{\theta}(\infty))$ over C_θ respectively, then $\beta_Q(k) \to \hat{\beta}(\infty)$ and $\theta_Q(k) \to \hat{\theta}(\infty)$ as $k \to \infty$.*

Proof. Notice the structure of $Q_k(\beta|\hat{\beta}(\infty))$ is the same as that of $\tilde{\phi}_k(\beta)$ as defined in Eq. (4.24), with the term $\bar{g}_{ij}(k)$ replaced by $\sum_{t=1}^{k} b^{k-t} g_{ij}(\hat{\beta}(\infty); t)$. Similarly, if we replace $\bar{c}_{ij}^{d}(k)$ in $\tilde{\phi}_k(\theta)$ of Eq. (4.18) with $\sum_{t=1}^{k} a^{k-t} c_{ij}^{d}(\hat{\theta}(\infty); t)$, we arrive at $Q_k(\theta|\hat{\theta}(\infty))$. Thus we can use Lemma 3 to construct an argument that exactly parallels the proof of Theorem IV.3. □

Although the parameter estimates arrive at fixed points of the exact EM algorithm asymptotically, we are not guaranteed that these are in fact maxima of the appropriate objective functions. This is because EM might not converge to a maximum of the likelihood. The work in [106] gives an extensive analysis of this issue.

4.5 Permutation Clustering

The posterior computation in Eq. (4.2) and the update formulas in Eqs. (4.15), (4.23) require evaluating sums of the form

$$\sum_{\rho} P_k(\rho) \prod_{(i,j)\in\chi_{k,\rho}} v_{ij} \tag{4.44}$$

where v is some parameter (e.g. θ^d or $1+\beta_0\beta$) and $\rho = 1, 2, \ldots |y_k|!$ indexes different permutations. The number of terms in this sum therefore grows exponentially with the number $|y_k|$ of activated sensors. It is not feasible to compute these sums online when more than 5 or 6 sensors are activated. However, we might have some ordering information that could rule out many of these permutations, i.e. $P_k(\rho) = 0$ for most of the orderings ρ. If no ordering information is available, computing the sums directly is hopeless for long paths. In the remainder of this section and the next, we

formulate a combinatorial scheme for approximating the sums under such conditions where the path is long (say, $|y_k| > 6$) and $P_k(\rho) = 1/|y_k|!$ for all ρ. We assume a uniform ordering distribution throughout this discussion for simplicity, however, these techniques easily extend to the situation where some permutations have larger probabilities and the rest are equally likely by simply changing the weighting scheme.

4.5.1 Permutation Approximation Algorithm

A key point to notice in developing an approximation algorithm is that the index set of Eq. (4.44) satisfies $\chi_{k,\rho} \subset S_k \equiv y_k^2 \cup \Sigma \times y_k \cup y_k \times \Delta - \bigcup_{n=1}^{|y_k|}(y_k^n, y_k^n)$ for all ρ. The number of distinct terms in the product therefore grows only as $|y_k|^2$, even though the total number of terms in the sum grows exponentially in $|y_k|$. Suppose further that all of the parameters v_{ij} for $(i,j) \in S_k$ are similar. In this case, we could obtain a reasonable approximation to the sum in Eq. (4.44) by the following.

$$\sum_{\rho} \frac{1}{|y_k|!} \prod_{(i,j)\in\chi_{k,\rho}} v_{ij} \approx \bar{v}_0^{|y_k|+1} \tag{4.45}$$

where $\bar{v}_0$ is the geometric mean of $\{v_{ij} \,|\, (i,j) \in S_k\}$. This approximation essentially clusters all $|y_k|!$ permutations into a single term. Note that we need only that the geometric mean of $\{v_{ij}|(i,j) \in \chi_{k,\rho}\}$ be similar for all ρ for the approximation in Eq. (4.45) to hold. Although this is a weaker condition than requiring v_{ij} be similar for all $(i,j) \in S_k$, it is much harder to verify precisely because there is an exponential number of orderings ρ. We can refine the approximation iteratively by removing elements from S_k and including all valid permutations over such elements in the sum. For example, we arrive at a first refinement of the approximation in Eq. (4.45) by setting $C_1 \equiv S_k - (i_1, j_1)$ and including $v_{i_1 j_1}$ explicitly in the sum.

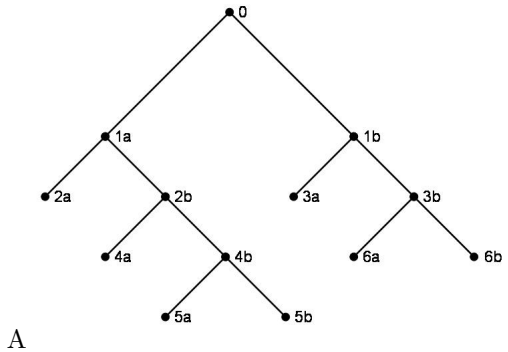
$$\sum_{\rho} \frac{1}{|y_k|!} \prod_{(i,j)\in\chi_{k,\rho}} v_{ij} \approx \frac{|y_k|! - (|y_k|-1)!}{|y_k|!} \bar{v}_1^{|y_k|+1} + \frac{(|y_k|-1)!}{|y_k|!} v_{i_1 j_1} \bar{v}_2^{|y_k|} \tag{4.46}$$

where $\bar{v}_l$ is the geometric mean of $\{v_{ij} \,|\, (i,j) \in C_l\}$, and C_2 is defined by all elements $(i,j) \in S_k$ such that the sequence (i,j) could exist in a valid permutation with the sequence (i_1, j_1) (we denote this by $(i,j) \sim (i_1,j_1)$). We could continue to produce refinements of the sum approximation in this way until S_k is empty and all $|y_k|!$ permutations appear in the sum. This describes the essential idea of the sum approximation algorithm.

We utilize an ordered version of the set S_k, denoted $\tilde{S}_k$, along with a binary tree to organize the terms in the sum. At each refinement step, the first element of $\tilde{S}_k$ is removed and used to update the binary tree. Each node of the tree is characterized by three quantities: Z, α, and C. The characteristic Z is the set of all $(i,j) \in S_k$ such that v_{ij} appears explicitly in the sum term represented by that node, α is the number of permutations that are clustered into the term, and C is the set of all $(i,j) \in \tilde{S}_k$ such that $(i,j) \sim Z$ (that is, the set of all (i,j) such that $(i,j) \cup Z$ might form a valid permutation). For example, the characteristics associated with first term in Eq. (4.46) are $Z = \phi$, $\alpha = |y_k|! - (|y_k| - 1)!$, and $C = S_k - (i_1, j_1)$. The pseudocode for the permutation clustering approximation is as follows.

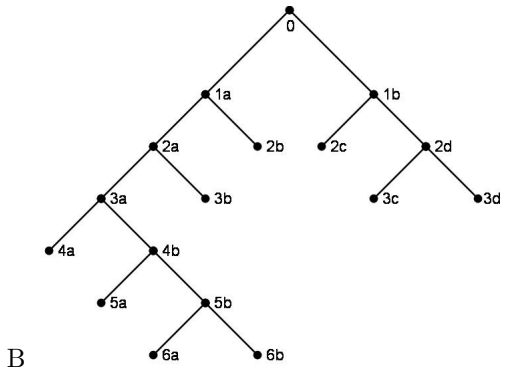
Algorithm 1. *:*

- *Given a parameter sequence $\tilde{S}_k$, define a tree root with characteristics $\alpha = |y_k|!$, $Z = \phi$, and $C = \tilde{S}_k$. Set $L = 1$.*
- *While $\tilde{S}_k$ is nonempty and $L \leq L_{\max}$:*
  - *Set $(i_*, j_*) = \tilde{S}_k(1)$ and $\tilde{S}_k = \tilde{S}_k - \tilde{S}_k(1)$.*
  - *For all leaves l such that $(i_*, j_*) \in C_l$ and $\alpha_l > 0$:*
    - *Add a left child to l with characteristics $Z = Z_l$ and $C = C_l - (i_*, j_*)$.*



A

Node	α	Z	C
0	2	ϕ	$\{(1,2),(1,d),(s,1),(2,1),(s,2),(2,d)\}$
1a	1	ϕ	$\{(1,d),(s,1),(2,1),(s,2),(2,d)\}$
1b	1	$\{(1,2)\}$	$\{(s,1),(2,d)\}$
2a	0	ϕ	$\{(s,1),(2,1),(s,2),(2,d)\}$
2b	1	$\{(1,d)\}$	$\{(2,1),(s,2)\}$
3a	0	$\{(1,2)\}$	$\{(2,d)\}$
3b	1	$\{(s,1),(1,2)\}$	$\{(2,d)\}$
4a	0	$\{(1,d)\}$	$\{(s,2)\}$
4b	1	$\{(2,1),(1,d)\}$	$\{(s,2)\}$
5a	0	$\{(2,1),(1,d)\}$	ϕ
5b	1	$\{(s,2),(2,1),(1,d)\}$	ϕ
6a	0	$\{(s,1),(1,2)\}$	ϕ
6b	1	$\{(s,1),(1,2),(2,d)\}$	ϕ



B

Node	α	Z	C
0	2	ϕ	$\{(1,2),(s,1),(2,d),(2,1),(s,2),(1,d)\}$
1a	1	ϕ	$\{(s,1),(2,d),(2,1),(s,2),(1,d)\}$
1b	1	$\{(1,2)\}$	$\{(s,1),(2,d)\}$
2a	1	ϕ	$\{(2,d),(2,1),(s,2),(1,d)\}$
2b	0	$\{(s,1)\}$	$\{(2,d)\}$
2c	0	$\{(1,2)\}$	$\{(2,d)\}$
2d	1	$\{(s,1),(1,2)\}$	$\{(2,d)\}$
3a	1	ϕ	$\{(2,1),(s,2),(1,d)\}$
3b	0	$\{(2,d)\}$	ϕ
3c	0	$\{(s,1),(1,2)\}$	ϕ
3d	1	$\{(s,1),(1,2),(2,d)\}$	ϕ
4a	0	ϕ	$\{(s,2),(1,d)\}$
4b	1	$\{(2,1)\}$	$\{(s,2),(1,d)\}$
5a	0	$\{(2,1)\}$	$\{(1,d)\}$
5b	1	$\{(s,2),(2,1)\}$	$\{(1,d)\}$
6a	0	$\{(s,2),(2,1)\}$	ϕ
6b	1	$\{(s,2),(2,1),(1,d)\}$	ϕ

Figure 4.5: Example permutation clustering trees. Here, just two sensors 1 and 2 are activated. The tree A utilizes the parameter sequence $\tilde{S}_k = ((1,2),(1,d),(s,1),(2,1),(s,2),(2,d))$, while tree B uses the sequence $\tilde{S}_k = ((1,2),(s,1),(2,d),(2,1),(s,2),(1,d))$. Nodes 1a and 1b are formed after the first element in the sequence ($(1,2)$ for both A and B) is appended, nodes 2a, 2b, 2c, and 2d are formed after the second element in the sequence ($(1,d)$ for A and $(s,1)$ for B) is appended, and so on. Each tree produces a complete enumeration of the permutation set with characteristic quantities given in the tables to the right. For tree A, nodes 5b and 6b give the complete permutation set, while nodes 3d and 6b are the complete permutation set in tree B. It is clear from this example that the order of the parameter sequence $\tilde{S}_k$ will have a large impact on the formation of the tree. A greedy heuristic for selecting this is described and justified in the next section.

  - $*$ *Add a right child to* l *with characteristics* $Z = Z_l \cup (i_*, j_*)$ *and* $C = \{(i,j) \in \tilde{S}_k \,|\, (i,j) \sim Z_l\}$.

- *Number all new leaves and those existing leaves with* $\alpha > 0$ *in order of increasing* $|Z|$ *with the integers* $1, 2, \ldots, L$.

- *For* $l = L, L-1, \ldots 1$*:*

  - $*$ *Set* $\alpha_l = (|y_k| - |Z_l|)! - \sum_{i|i>l, Z_l \subset Z_i} \alpha_i$.

- *Construct the sum represented by all leaves with* $\alpha > 0$.

Example binary trees produced by this algorithm for two different parameter se-

quences are given in Figure 4.5. Since at each step we only form new leaves that might be permutations, it is clear that all permutations will have been enumerated once $\tilde{S}_k$ is empty. One might contrast this method with the standard permutation generating tree due to [16]. The standard method enumerates all permutations of $\{1, 2, \ldots n\}$ by recursively forming children $(k+1, \pi_1, \pi_2, \ldots, \pi_k), (\pi_1, k+1, \pi_2, \ldots, \pi_k), \ldots (\pi_1, \pi_2, \ldots, \pi_k, k+1)$ to a given parent node $(\pi_1, \pi_2, \ldots, \pi_k)$. Note that our method is not strictly a generating tree, since the α characteristic of a child may depend on nodes other than its parent [99]. Our method is desirable, however, in that it allows more direct control over the order in which permutations (or partial permutations) are generated–through the ordering of the set $\tilde{S}_k$–without the need for complicated backtracking through the tree. This is important because we rarely generate the entire tree in the case of long paths. Indeed, once some maximum number of leaves $L_{\max}$ are accumulated, we truncate the tree to obtain the following approximation:

$$\sum_{\rho} \prod_{(i,j)\in\chi_{k,\rho}} v_{ij} \approx \sum_{l=1}^{L} \alpha_l \bar{v}_l^{|y_k|+1-|Z_l|} \prod_{(i,j)\in Z_l} v_{ij} \tag{4.47}$$

where $\bar{v}_l$ is the geometric mean of $\{v_{ij} \,|\, (i,j) \in C_l\}$.

In the numerators of Eqs. (4.15) and (4.23), it is necessary to compute restricted versions of the above sum; instead of summing over all ρ we only consider ρ such that some $(i,j) \in \chi_{k,\rho}$. The approximation is exactly as in Eq. (4.47) if it happens that the particular (i,j) has been removed from $\tilde{S}_k$ and added to Z_l for some leaf l in the tree before truncation. If we truncate before adding (i,j) to the tree, then we approximate by considering a weighted sum over leaves that might form a valid permutation with (i,j) (i.e. all leaves l such that $(i,j) \in C_l$). The restricted sum

approximation is therefore

$$\sum_{\rho|(i,j)\in\chi_{k,\rho}} \prod_{(i',j')\in\chi_{k,\rho}} v_{i'j'} \approx$$

$$\begin{cases} \sum_{l|(i,j)\in Z_l} \alpha_l \bar{v}_l^{|y_k|+1-|Z_l|} \prod_{(i',j')\in Z_l} v_{i'j'} & \text{if } (i,j) \notin \tilde{S}_k \\ \frac{(|y_k|-1)!}{\sum_{l|(i,j)\in C_l} \alpha_l} \sum_{l|(i,j)\in C_l} \alpha_l \bar{v}_l^{|y_k|+1-|Z_l|} \prod_{(i',j')\in Z_l} v_{i'j'} & \text{if } (i,j) \in \tilde{S}_k \end{cases} \tag{4.48}$$

When considering sums over permutations with different source/destinations, as in Eqs. (4.2) and (4.23), we can simply follow this procedure to approximate the sum over ρ for each pair (s,d).

4.5.2 Permutation Approximation Analysis

The sequence $\tilde{S}_k$ and the truncation limit $L_{\max}$ will determine approximations to the functions $c(\theta;k)$ and $g(\beta;k)$ as defined in Eqs. (4.15) and (4.23), respectively. Provided matching $\tilde{S}_k$ and $L_{\max}$ values are used for the recursive and a comparable non-recursive EM iteration as in Eqs. (4.43) and (4.42), one can easily verify that all necessary properties of the functions $c(\theta;k)$ and $g(\beta;k)$ hold to ensure validity of the previous convergence analysis. The permutation clustering approximation greatly decreases the complexity associated with computing $c(\theta;k)$ and $g(\beta;k)$. Full computation of these functions requires $O(|y_k|!)$ time. The permutation approximation reduces this to a low order polynomial in $|y_k|$. Suppose we fix $L_{\max}$ so that it does not grow with $|y_k|$. Note that each time an element from $\tilde{S}_k$ is added to the tree, the number of leaves at most doubles so that we can always ensure truncation before $L_{\max}$ is exceeded. The only operation that scales is computation of the characteristic C associated with new right children, and the geometric mean $\bar{v}$ corresponding to this set. In determining C, we need to check that the conditions defining a permutation are not violated if any of the pairs $(i,j) \in C$ are added to the set Z. This can be done recursively by simply removing any elements from the parent's C char-

acteristic that might result in any repetitions or incomplete paths from s to d after augmenting the parent's Z characteristic with (i_*, j_*). The cardinality of any C is at most $|y_k|^2 + 2|y_k|$. It follows that the permutation clustering approximation reduces the complexity from exponential in $|y_k|$ to $O(|y_k|^2)$.

We now develop some bounds on how well the permutation clustering approximation agrees with the full sum over all permutations. First note that all permutations ρ clustered into a given leaf l must satisfy $\chi_{k,\rho} \subset Z_l \cup C_l$, and $Z_l \cap C_l = \phi$ by definition of these characteristics. If we define $\bar{v}_{l,\min}$ as the geometric mean of the $|y_k| + 1 - |Z_l|$ smallest elements of $\{v_{ij} \,|\, (i,j) \in C_l\}$ and $\bar{v}_{l,\max}$ as the geometric mean of the $|y_k|+1-|Z_l|$ largest elements of this set, then we have the following inequalities for any ρ such that $\chi_{k,\rho} \subset Z_l \cup C_l$.

(4.49)
$$\begin{aligned} \bar{v}_{l,\min}^{|y_k|+1-|Z_l|} \prod_{(i,j)\in Z_l} v_{ij} &\leq \prod_{(i,j)\in\chi_{k,\rho}} v_{ij} = \left(\prod_{(i,j)\in\chi_{k,\rho}\cap C_l} v_{ij}\right)\left(\prod_{(i,j)\in Z_l} v_{ij}\right) \\ &\leq \bar{v}_{l,\max}^{|y_k|+1-|Z_l|} \prod_{(i,j)\in Z_l} v_{ij} \end{aligned}$$

It is also obvious that $\bar{v}_l^{|y_k|+1-|Z_l|} \prod_{(i,j)\in Z_l} v_{ij}$ lies within the bounds of Eq. (4.49), since $\bar{v}_l$ is the geometric mean of *all* elements in $\{v_{ij} \,|\, (i,j) \in C_l\}$. Now, the leaves represent a partition of the permutation set, so we have

$$\sum_{\rho} \prod_{(i,j)\in\chi_{k,\rho}} v_{ij} = \sum_{l} \sum_{\rho|\chi_{k,\rho}\subset Z_l\cup C_l} \prod_{(i,j)\in\chi_{k,\rho}} v_{ij} \tag{4.50}$$

We can then combine Eq. (4.50) with the inequalities in (4.49) and realize that α_l is the number of permutations ρ that satisfy $\chi_{k,\rho} \subset Z_l \cup C_l$ to arrive at the following bound on the approximation error in Eq. (4.47).

$$\begin{aligned} &\left|\textstyle\sum_\rho \prod_{(i,j)\in\chi_{k,\rho}} v_{ij} - \sum_{l=1}^{L} \alpha_l \bar{v}_l^{|y_k|+1-|Z_l|} \prod_{(i,j)\in Z_l} v_{ij}\right| \leq \\ &\textstyle\sum_{l=1}^{L} \alpha_l \left(\bar{v}_{l,\max}^{|y_k|+1-|Z_l|} - \bar{v}_{l,\min}^{|y_k|+1-|Z_l|}\right) \prod_{(i,j)\in Z_l} v_{ij} \end{aligned} \tag{4.51}$$

One can arrive at bounds for the approximation error associated with Eq. (4.48) in a similar fashion. When $(i,j) \notin \tilde{S}_k$, the form of the bound is almost identical to that

in Eq. (4.51) with the only difference being a restriction on the sum (only over l such that $(i,j) \in Z_l$). The bound is looser when $(i,j) \in \tilde{S}_k$ because we do not know the correct proportions for including each leaf in the sum. There is some loss associated with the weighted sum approximation in Eq. (4.48). It is straightforward to apply these results to determine bounds on the acutal estimators when the permutation clustering approximation is used. Clearly, there is no approximation error if the geometric mean of the $|y_k|+1-|Z_l|$ smallest elements of $\{v_{ij} \mid (i,j) \in C_l\}$ is equal to the geometric mean of the $|y_k|+1-|Z_l|$ largest elements of the set for all l. Since C_l is always a subset of $\tilde{S}_k$, this suggests a reasonable strategy is to choose the ordering $\tilde{S}_k$ so as to reduce the range of $\{v_{ij} \mid (i,j) \in \tilde{S}_k\}$ as much as possible each time an element is removed from $\tilde{S}_k$. This is a simple greedy approach to the problem of selecting the parameter sequence, however one might pose some optimization problem for selecting the sequence that is best in a nonmyopic setting. In most cases, such an optimization would result in additional online computational strain.

4.6 Online Probe Scheduling

Here we propose some methods for online probe scheduling. Previously, we assumed the training phase occurred before the monitoring phase. In this section, it is necessary to consider a different training paradigm wherein probes of the network are scheduled during observation downtime, that is, when a suspect observation is not observed. These consist of placing a call from some known source to some known destination and noting the activated sensor set and ordering distribution. Although they might seem secondary to observing suspects, probes are necessary for us to learn the routing parameters of the network. It might not be clear which are the best probes to make until we go online and begin recording measurements. A rapid,

online scheduling algorithm is certainly advantagesous in this paradigm.

We model the probe scheduling problem as a multiarmed bandit. Each different source/destination pair, that is each distinct element of $\Sigma \times \Delta$, is a separate arm of the bandit. The reward associated with scheduling some pair (s, d) is given by the information gained as a result of the probe. We use the change in entropy of the suspect endpoint posteriors as a measure of information gain. The reward r_{sd} for scheduling (s, d) is therefore given by

$$r_{sd} = \sum_t \lambda_t \Delta H(P(s, d|y_t, \hat{\theta}(k))) \tag{4.52}$$

where λ_t are constants that sum to one and allow a weighted average of the entropy change ΔH in all observed suspect posteriors. Given the reward function, we can directly apply the Exp3 algorithm of [3] for control of the multiarmed bandit. Exp3 uses a parameter $\delta \in (0, 1]$ and is based on the following recursions:

$$\begin{aligned} p_i &= (1-\delta)\frac{w_i(k-1)}{\sum_j w_j(k-1)} + \frac{\delta}{|\Sigma \times \Delta|} \\ w_i(k) &= \begin{cases} w_i(k-1)\exp\left(\frac{\delta r_{sd}}{p_{sd}|\Sigma \times \Delta|}\right) & \text{if } i = (s,d) \\ w_i(k-1) & \text{else} \end{cases} \end{aligned} \tag{4.53}$$

where $w_i(0) = 1$ for all $i \in \Sigma \times \Delta$. It is clear that p is a mixture distribution over the endpoint pairs consisting of a uniform component and a component shaped by the rewards. At each time step, the endpoint pair to be scheduled is chosen from p. There are several versions of the algorithm with slightly different asymptotic performance guarantees. We refer the reader to [3] for a thorough theoretical treatment.

4.7 Experimental Results

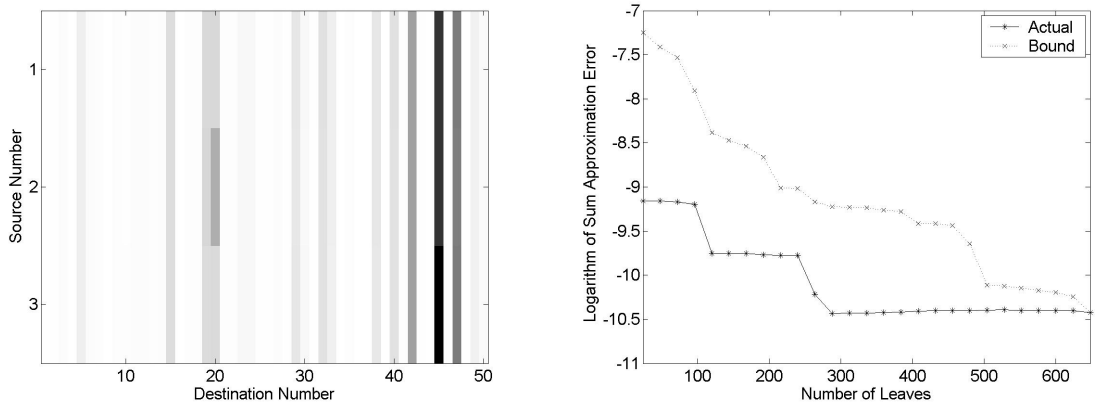
We applied the new online estimation methods to the `traceroute` data presented in [77]. The data was obtained from `traceroute` probes initiated on October 12,

2005 from three sources located at the University of Wisconsin-Madison, the Instituto Superior Tecnico in Lisbon, Portugal, and Rice University in Houston, Texas to fifty destination web servers of various companies, universities, and governments around the world. We treat the routers encountered as sensors, and ignore all ordering information, so that the ordering distributions $P_k(\rho)$ associated with all measurements ρ are uniform. After processing the data to collapse identical routers–that is routers that are always activated together across the 150 measurements–we were left with 241 routers and path lengths ranging from 2 to 14 hops, with an average of 7.5. In light of the long paths and uniform ordering distributions, enumeration of permutations was infeasible so we had to apply the permutation clustering approximations to compute parameter estimates. For purposes of initialization, we assumed all edge probabilities γ_{ij} were set to 0.5. Precision parameter values of $\gamma_0 = 0.0002$ and $\beta_0 = 1$ were used for all experiments.

Our first simulation illustrates the accuracy of the permutation clustering approximation that is used in subsequent experiments. We next present the core simulation of a moving suspect. The suspect moves through the network broadcasting from different sources, and we are able to track its position using the tracking parameter estimates along with the endpoint posteriors. In the next section, we simulate a sensor failure and show how the routing parameter estimates are able to detect this. Finally, we investigate the effectiveness of the multiarmed bandit scheduling algorithm by analyzing the evolution of the distribution from which scheduled probes are drawn.

4.7.1 Permutation Clustering Approximation Error

We devised an experiment to test the accuracy of the permutation clustering method for approximating combinatorial sums that arise in our estimators. After



A: True Endpoint Posterior B: Permutation Clustering Appx. Error

Figure 4.6: Illustration of the permutation sum approximation accuracy. In A, we have the exact endpoint posterior of a suspect transmission activating six sensors. Darker color indicates higher value in this two dimensional distribution. The true endpoints of this suspect were source 3 and destination 45; so we see that the correct destination is clearly pinpointed while there is a bit of ambiguity in the source estimate. Plot B shows the error (in a logarithmic scale) when permutation clustering is used to approximate this endpoint posterior. The number of leaves in the clustering tree were varied from 24 up to 648 in steps of 24. Asterisks connected by a solid line indicate the actual error (as on the left side of Eq. (4.51)), while X's connected by a dotted line indicate the error bound on the right side of Eq. (4.51).

initializing the system as described above, we trained using observed paths between all 150 possible source/destination combinations. Training was done with minimal forgetting–that is, a forgetting factor $a = 0.999999$. Then a single suspect transmission passed between source 3 and destination 45 was observed during the monitoring stage. This suspect activated a total of six sensors, thus we were able to compute its exact endpoint posterior as in Eq. (4.2) by summing over all 720 orderings. The exact posterior is shown in Figure 4.6A. We then used the permutation clustering method to approximate the endpoint posterior as in Eq. (4.47) with number of leaves L ranging from 24 up to 648 in steps of 24. The absolute error as on the left side of Eq. (4.51) is plotted in Figure 4.6B, along with the derived error bound.

We see that the permutation clustering approximation performs quite well, falling into the realm of round-off error after about 300 leaves are used in the tree. Furthermore, the actual approximation error is about a hundred times smaller than the

worst case bound for fewer leaves (24, 48, 72). This suggests it is reasonable to proceed with application of this method in the following simulations. In all remaining simulations, we utilize permutation clustering trees having at most 24 leaves for parameter updates and posterior computations.

4.7.2 Suspect Tracking

This experiment simulates the movement of a suspect through the network and illustrates the tracking abilities of the proposed methods. As before, we begin by initializing the system and training it using all 150 source/destination pairs with minimal forgetting. Then, for the first 100 clock ticks of the monitoring phase, we observe suspect transmissions emanating from source 1 and terminating at random destinations. We observe transmissions from source 2 to random destinations for the next 100 clock ticks. For the final 100 clock ticks of monitoring, the suspect moves to source 3 and broadcasts to random destinations. A forgetting factor of $b = 0.9$ is used throughout in estimation of the tracking parameters β.

Our goal is to determine which source node the suspect is broadcasting from at each tick of the clock. One natural indicator of location is simply the instantaneous source posterior distribution given by

$$P_s(k) \propto \sum_d \sum_\rho \prod_{(i,j)\in\chi_{k,\rho}} \hat{\theta}_{ij}^d(k) \tag{4.54}$$

where a uniform endpoint prior $P(s, d)$ is assumed, and proportionality (rather than equality) is used because we have omitted a normalization constant. In addition to the instantaneous source posterior, one might look at the values of the tracking parameters associated with sensors that are exclusive to each source. In particular, we say a sensor is exclusive to source s if it is only activated when s is probed. We are able to use our probing measurements from the training phase to determine

the exclusivity of the various sensors. Based on this notion, we define the average entering probabilty $E_s(k)$ at time k associated with source s as follows.

$$E_s(k) \propto \frac{1}{|\{j|\ j \text{ is exclusive to } s\}|} \sum_{j|\ j \text{ is exclusive to } s} \sum_i \hat{\beta}_{ij}(k) \tag{4.55}$$

The name follows from the analogy to a Markov chain: since $\hat{\beta}(k)$ is the transition matrix of a Markov chain, the quantity defined in Eq. (4.55) can be interpreted as the probability of suspect measurements entering sensors exclusive to source s. Thus the larger $E_s(k)$ is, the more messages are entering sensors exclusive to s, and thus it is more likely that the suspect is broadcasting from s. Also, each row of the matrix $\hat{\beta}(k)$ is a probability distribution itself, with $\hat{\beta}_{ij}(k)$ reprensenting the probability a suspect measurement exits element i and arrives in element j. We can utilize the average exit distribution entropy $H_s(k)$ of sensors exclusive to s as another location indicator. This quantity is defined as

$$H_s(k) \propto \frac{1}{|\{i|\ i \text{ is exclusive to } s\}|} \sum_{i|\ i \text{ is exclusive to } s} \sum_j -\hat{\beta}_{ij}(k) \log \hat{\beta}_{ij}(k) \tag{4.56}$$

We interpret the value of $H_s(k)$ as follows: the smaller $H_s(k)$ is the more information we have about exit probabilities of sensors exclusive to s, this indicates more suspect messages are departing from sensors exclusive to s, and it is therefore more likely that the suspect is broadcasting from s. Recall that a uniform initialization is used (all $\gamma_{ij} = 0.5$) so that all exit distributions are nominally uniform in the absence of suspect measurements.

We repeated this experiment 30 times, each time chosing independent random destinations, in order to average over the effect of randomly chosen destinations. The indicator quantities of Eqs. (4.54), (4.55), and (4.56) were recorded and averaged over these 30 trials. The averaged quantities are plotted versus clock tick in

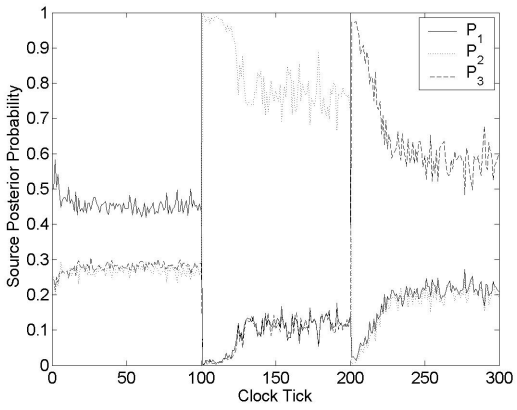
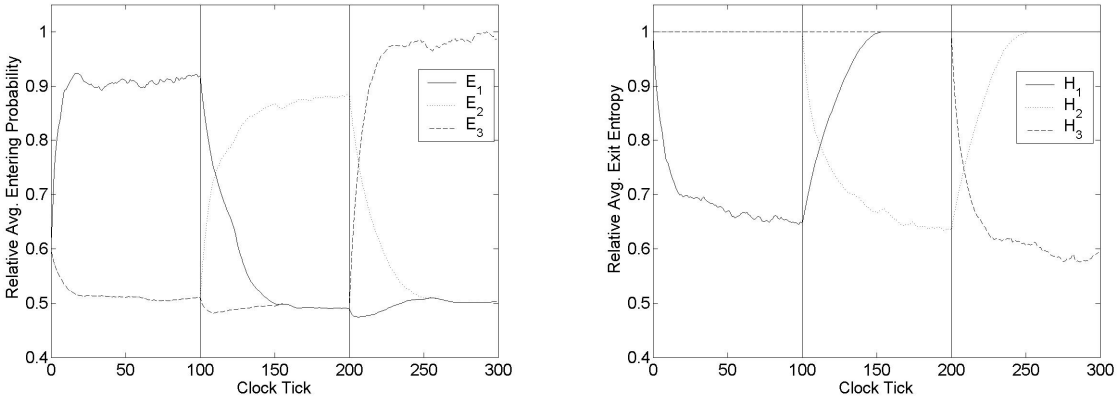


Figure 4.7: Instantaneous source posterior probability of Eq. (4.54) as a function of clock tick. The solid line represents $P_1(k)$, the dotted is $P_2(k)$, and the dashed is $P_3(k)$. Vertical lines are drawn at each transition time (from source 1 to source 2, and from source 2 to source 3). We see that the estimator is able to correctly locate the suspect at each point in time, as indicated by the larger value of $P_s(k)$ for the correct s.



A: Relative Avg. Entering Probability

B: Relative Avg. Exit Entropy

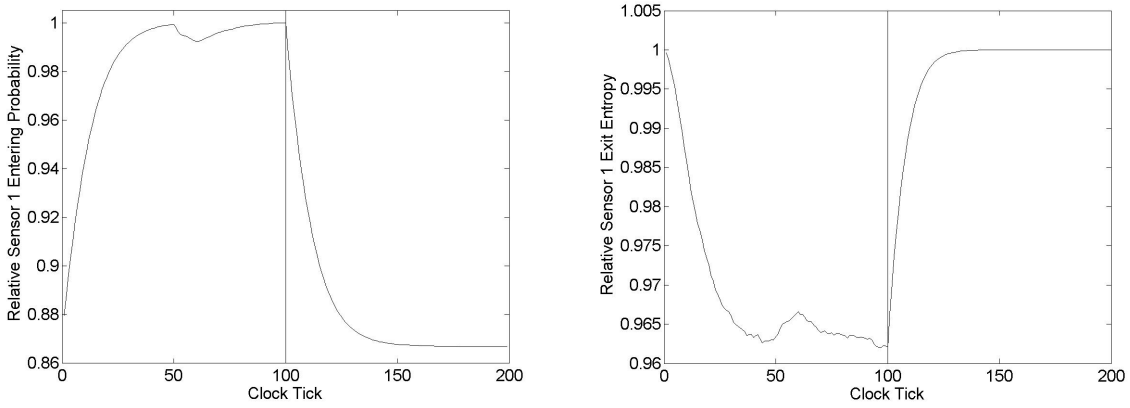
Figure 4.8: Plots of average entering probability in A and average exit distribution entropy in B as defined in Eqs. (4.55) and (4.56) respectively. The values are are normalized to the maximum in each plot. The solid line represents quantities associated with source 1, the dotted represents source 2, and the dashed represents source 3. Vertical lines are drawn at each transition time (from source 1 to source 2, and from source 2 to source 3). These indicators also point to the correct source location at the correct time as indicated by a rise in the appropriate entering probability $E_s(k)$, and a drop in the appropriate exit distribution entropy $H_s(k)$. At transition points, there is a decay of the previous extreme quantity with decay time determined by the forgetting factor b.

Figures 4.7 and 4.8. We see that the instantaneous source posterior probability pinpoints the correct suspect location during each 100-tick period. The average entering probability and exit entropy indicators also point to the correct source at the correct time. There is, however, some characteristic decay time in these quantities determined by the forgetting factor b. These simulations suggest our algorithms would be quite useful when applied to tracking problems.

4.7.3 Sensor Failure

This simulation shows how one might use the routing parameter estimates in an interleaved probing paradigm such as that described in the online scheduling section (where probes occur during clock ticks when suspects are not observed). If suspects are constantly arriving, one might not have enough downtime for excessive probing. In this case, it is useful to monitor the evolution of the routing parameters for significant deviation from their nominal values. A large change in the network, such as failure of a sensor, would prompt such a deviation. It is then necessary to halt monitoring long enough to train the parameters to the new routing dynamics in the network.

We simulate such a sensor failure in this example and show how the failure is reflected in the routing parameter estimates. We suppose probes of the network from source 1 to random destinations are scheduled for 200 consecutive clock ticks with a forgetting factor of $a = 0.9$. Sensor 1 is positioned such that transmissions from source 1 to any destination always pass it. At time 100, sensor 1 fails–meaning that messages are still routed through it, but it does not activate in response to their passing. We computed average entering probability $E(k)$ and exit distribution entropy $H(k)$ for sensor 1. Similar to the definitions in Eqs. (4.55) and (4.56), these



A: Relative Avg. Entering Probability

B: Relative Avg. Exit Entropy

Figure 4.9: Plots of average entering probability in A and average exit distribution entropy in B as defined in Eq. (4.57) for sensor 1. The values are are normalized to the maximum in each plot. A vertical line is drawn at the point where sensor 1 fails. We see a drop in the entering probability and a rise in the exit entropy beginning at the failure point; of course there is a decay time determined by the forgetting factor a.

are defined as

$$\begin{aligned} E(k) &\propto \textstyle\sum_d \sum_i \hat{\theta}^d_{i1}(k) \\ H(k) &\propto \textstyle\sum_d \sum_j -\hat{\theta}^d_{1j}(k) \log \hat{\theta}^d_{1j}(k) \end{aligned} \tag{4.57}$$

We averaged over the effect of random probe orders by repeating this experiment 30 times with independent random probes from source 1 and averaging the quantities in Eq. (4.57) over those 30 trials. The results are plotted in Figure 4.9.

We observe in Figure 4.9 a signficant change in the nominal values of entering probability and exit entropy associated with sensor 1 after the failure point. In the alternative training scheme discussed above, one might set some allowed tolerance around the nominal. Once this is exceeded, we would have to schedule several probes to learn the new routing dynamics of the network.

4.7.4 Online Scheduling

We investigate the utility of the multiarmed bandit control algorithm of [3] applied to online probe scheduling in this example. Here, a single suspect transmission is observed initially. Then the online scheduling algorithm as described in Section VI is

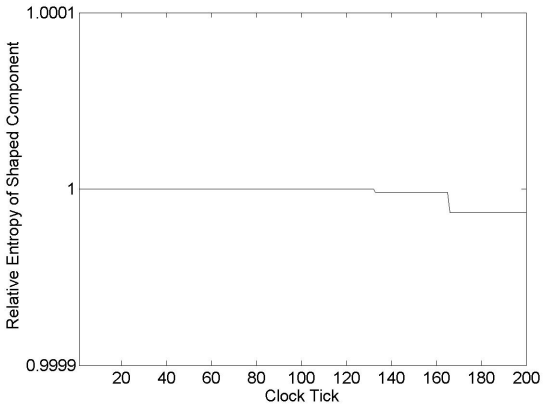


Figure 4.10: Reward shaped distribution entropy $-\sum_i w_i(k) \log w_i(k)$ as a function of clock tick were $w(k)$ is defined in Eq. (4.53). The entropy is normalized by its initial value. We see that the entropy deviates very little from its maximum value in 200 clock ticks. This indicates that probes are essentially drawn from a uniform distribution throughout the simulation (since the other component of p in Eq. (4.53) is uniform).

used to schedule probes for 200 clock ticks. At each probe, the reward (determined by the resulting change in entropy of the suspect endpoint posterior) is used in the recursions of Eq. (4.53) to update the distribution from which the next probe is drawn. Using a parameter $\delta = 0.1$, we are interested in how quickly the shaped component $w(k)$ of the distribution p is able to concentrate on the best probes to make for this particular suspect. We therefore plot the entropy of w (given by $-\sum_i w_i(k) \log w_i(k)$) as a function of clock tick to measure the concentration. This plot is shown in Figure 4.10.

We observe from Figure 4.10 that probes are essentially drawn from a uniform distribution through the entire simulation. The shaped component $w(k)$ concentrates slightly toward the end, as indicated by a small drop in entropy. One might suggest simply scaling the reward to speed up the process, however, the theory of [3] requires a reward between 0 and 1. This simulation seems to indicate that the proposed bandit scheduling algorithm requires a rather lengthy period of time to be effective. However, additional investigation of the utility of the algorithm is certainly in order

for future work.

4.8 Summary and Future Work

We have presented online techniques for adaptively estimating the source and destination of a suspect transmission through a network based on the activation pattern of sensors placed on network components. In addition to a thorough theoretical development, we applied the new methods to several tracking experiments involving real Internet data obtained using the `traceroute` command. Speedy and accurate results were observed.

In the way of future work, one might analyze further the permutation clustering algorithm; in particular, issues related to selection of the parameter sequence $\tilde{S}_k$ and tree truncation level. We suggested a heuristic for choosing $\tilde{S}_k$ based upon the derived performance bound. Also, we assumed a given number of allowed leaves $L_{\max}$ before truncating the tree. One might consider linking these two ($\tilde{S}_k$ and $L_{\max}$) and solving some optimization problem to give a parameter sequence and truncation level that balances approximation accuracy and computational burden. The methods presented here could be applied with few changes to perform topology inference online, as an alternative to the offline approach of [77]. The probe scheduling method might also be extended to topology inference with a graph edit distance used to reward source/destination pairs that activate network segments similar to some prior structure of interest [45].

CHAPTER V

Markov Chain Monte Carlo Approach to Endpoint Estimation

5.1 Introduction

We collect some theoretical results for a Markov Chain Monte Carlo approach to the endpoint localization problem in this chapter. We assume measurements are taken at discrete time instances specified by the index t, and T_τ denotes the set of all indices up to the current time τ ($T_\tau = \{1, 2, \ldots \tau\}$). Let Σ denote the set of all source nodes and Δ denote the set of all destination nodes. A transmission is sent in a network Γ_Θ between some source/destination pair $u_t \in \Sigma \times \Delta$ that results in the excitation of an ordered set of sensors y_t. The sensors are ordered in y_t depending on the order they were passed by the transmission due to u_t. The set of measurements performed on the network are partitioned into two types: active and passive. Active measurements up to and including the current time τ occur at times $t \in A_\tau \subseteq T_\tau$ and correspond to those where we select the source/destination pair u_t. All other measurements are passive measurements, where u_t is unknown but chosen from a known distribution $P_t(u)$.

At each time instant t, the unobserved ordered set y_t generates observations consisting of an unordered set of activated sensors $y_{t,R}$ along with a probability distribution over all possible orderings $P(R_t = \rho) = P_t(\rho)$. This distribution allows

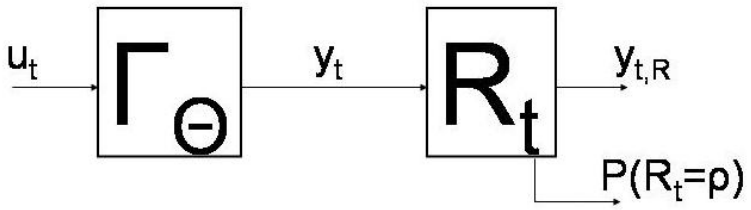


Figure 5.1: Model of the internally-sensed network tomography measurement system. u_t represents the source/destination pair used to excite the system at time t, y_t is the ordered set of sensors activated in the network Γ parameterized by some set of parameters Θ. R_t generates the observed elements: an unordered set of the elements of y_t, denoted $y_{t,R}$, along with a probability distribution on possible orderings $P(R_t = \rho) = P_t(\rho)$. Also, for $t \in A_\tau$ u_t is known, and for $t \in M_\tau = T_\tau - A_\tau$ u_t is unknown but chosen from the known distribution $P_t(u)$.

incorporation of any synchronization information available. For example, if $P_t(\rho)$ is uniform over all possible orderings we have essentially no ordering information, whereas if $P_t(\rho) = \delta(\rho - \rho_o)$ we know exactly the ordering of the sensors so that $y_t = y_{t,\rho_o}$. The random variable R_t depends on sensor reliability and any prior information about sensor configuration; it is therefore mutually independent with all other variables in the system. A diagram of the system that illustrates the various signals involved is given in Figure 5.1.

We assume that the network being monitored, denoted by Γ_Θ in Fig. 5.1, is a static system with respect to the measurement time index t. If Θ is a set of parameters that specifies all relevant routing mechanisms in the network Γ for producing y_t from u_t, we then have the conditional independence $P(v_t|w_{t'}, \Theta) = P(v_t|\Theta)$ for any random variables v_t and $w_{t'}$ with $t \neq t'$.

In endpoint estimation, we wish to estimate the endpoints of passive measurements $u_{M_\tau} = \{u_t | t \in M_\tau = T_\tau - A_\tau\}$. We approach this problem from a MAP framework, thus the posterior distribution given all measurements up to the current time τ is required. Using conditional independence implied by the static system Γ_Θ,

the endpoint posterior distribution may be written as

$$
\begin{aligned}
P(u_{M_\tau}|y_{T_\tau,R},u_{A_\tau}) &= \int P(u_{M_\tau}|y_{M_\tau},\Theta)P(\Theta|y_{T_\tau,R},u_{A_\tau})d\Theta \\
&\propto \int \left(\prod_{t\in T_\tau} P_t(u_t)\sum_\rho P(y_{t,\rho}|u_t,\Theta)P_t(\rho)\right)P(\Theta)d\Theta
\end{aligned} \tag{5.1}
$$

where $P(\Theta)$ is some prior distribution on the parameter set and $P_t(u_t) \equiv 1$ for $t \in A_\tau$.

Our strategy for solving the endpoint estimation problem is to generate Monte Carlo approximations of the posterior distribution by specifying a model for the fundamental distribution $P(y|u,\Theta)$ and using averages of the quantity in Eq. (5.1) evaluated at samples from the parameter prior $P(\Theta)$. The Metropolis-Hastings (M-H) algorithm is useful for generating samples from the parameter prior because it does not require a normalized distribution, and the parameter set will often be of high dimension so we may only know the prior to within a normalization constant [63, 37, 15]. We now proceed to describe some reasonable possibilities for the routing model $P(y|u,\Theta)$ and derive the corresponding M-H algorithms.

5.2 Shortest Path Routing Model

The shortest path routing model of [46] assumes a message transmitted from source $s = u^1$ to destination $d = u^2$ will most likely take the shortest path between s and d. This model has parameters θ^{yu}_{ij}, the weight associated with the logical link between source/destination/sensor i and source/destination/sensor j, and θ^u_{ij}, the probability a message transmitted from source i to destination j will take the shortest path between i and j. The shortest path routing model is thus given by

$$
P(y|u,\Theta) = \begin{cases} \theta^u_{u^1u^2} & \text{if } |y|_\theta = |y_u|_\theta \\ 1-\theta^u_{u^1u^2} & \text{if } |y_u|_\theta < |y|_\theta < \infty \\ 0 & \text{if } |y|_\theta = \infty \end{cases} \tag{5.2}
$$

where y_u is the shortest path between u^1 and u^2 with respect to the path length norm $|y|_\theta = \theta^{yu}_{u^1 y^1} + \sum_{j=2}^{N} \theta^{yu}_{y^{j-1}y^j} + \theta^{yu}_{y^N u^2}$. Presumably, the shortest path will be taken most often, so $\theta^{u}_{u^1 u^2}$ is near unity.

In order to develop an M-H algorithm for sampling the parameter prior, we desire candidate samples that are likely to have active and passive measurements corresponding to shortest paths. This will ensure a variety of distinct parameter samples around the mode of the posterior distribution can be obtained in a reasonable number of algorithm iterations. To enforce this, we prefer candidate samples wherein the subgraph induced by selecting elements involved in a single measurement has sensor nodes all of degree two (or in-degree and out-degree both one in the case of a directed network) and endpoint nodes both of degree one. This essentially rules out the possibility of 'short-cuts' in the measured paths present in our samples. The strategy for generating candidates for the parameters θ^{yu}_{ij} is then to first generate a 0-1 adjacency matrix $\bar{\theta}_{ij}$ so that the subgraph requirements are satisfied as much as possible. Then, if $\bar{\theta}_{ij} = 1$, we select θ^{yu}_{ij} from some distribution $P_o(\theta^{yu}_{ij}|\hat{\theta}^{yu}_{ij})$ that may depend on its previous value $\hat{\theta}^{yu}_{ij}$, otherwise θ^{yu}_{ij} is set to ∞.

The subgraph requirements may be expressed as linear equalities on the 0-1 parameter matrix $\bar{\theta}$. For active measurements $t \in A_t$, these are given by

$$\begin{aligned} \sum_{y^i \in y_t \cup \{u^1_t, u^2_t\}} \bar{\theta}_{y^i y^j_t} &= 2 \text{ for all } j = 1, \ldots, N \\ \sum_{y^i \in y_t} \bar{\theta}_{y^i u^1_t} &= 1 \\ \sum_{y^i \in y_t} \bar{\theta}_{y^i u^2_t} &= 1 \end{aligned} \tag{5.3}$$

For passive measurements $t \in M_t$, we need only introduce 'slack' endpoints (σ_t, δ_t) that correspond to the unknown endpoints of the transmission. As with active mea-

surements, for the undirected case the constraints may then be written as

$$\begin{aligned} \sum_{y^i \in y_t \cup \{\sigma_t, \delta_t\}} \bar{\theta}_{y^i y_t^j} &= 2 \text{ for all } j = 1, \ldots, N \\ \sum_{y^i \in y_t} \bar{\theta}_{y^i \sigma_t} &= 1 \\ \sum_{y^i \in y_t} \bar{\theta}_{y^i \delta_t} &= 1 \end{aligned} \tag{5.4}$$

We shall return to a rigorous derivation of the linear constraints later.

The SDP hyperplane rounding method of [46] may then be used to produce $\bar{\theta}_{ij}$ that approximately satisfy the constraints. We must make a slight modification, however, so that all $\bar{\theta}_{ij}$, and therefore all θ_{ij}^{yu}, are independent. Recall that the sample element produced by this method is given by

$$\bar{\theta}_{ij} = \begin{cases} 1 & \text{if } sign(v_{ij} \cdot r) = sign(v_o \cdot r) \\ 0 & \text{if } sign(v_{ij} \cdot r) \neq sign(v_o \cdot r) \end{cases} \tag{5.5}$$

where v_{ij} is the column corresponding to variable $\bar{\theta}_{ij}$ in the Cholesky factorization of the solution to the SDP, v_o is the column for the extra variable introduced, and r is a random vector from the uniform distribution on the surface of the unit hypersphere. We can ensure independence if instead of using a single random vector r for all i, j, we choose an independent r_{ij} for each $\bar{\theta}_{ij}$. The distribution of $\bar{\theta}_{ij}$ is given by

$$P(\bar{\theta}_{ij}) = \begin{cases} 1 - \frac{1}{\pi} \arccos(v_{ij} \cdot v_o) & \text{if } \bar{\theta}_{ij} = 1 \\ \frac{1}{\pi} \arccos(v_{ij} \cdot v_o) & \text{if } \bar{\theta}_{ij} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

The distribution of the parameters of interest θ_{ij}^{yu} may then be expanded using the law of total probability as

$$P(\theta_{ij}^{yu} | \hat{\theta}_{ij}^{yu}) = I_{\Re}(\theta_{ij}^{yu}) P_o(\theta_{ij}^{yu} | \hat{\theta}_{ij}^{yu})(1 - \frac{1}{\pi} \arccos(v_{ij} \cdot v_o)) + I_{\infty}(\theta_{ij}^{yu}) \frac{1}{\pi} \arccos(v_{ij} \cdot v_o) \tag{5.7}$$

where $I : \Re \rightarrow \{0, 1\}$ is the indicator function. Finally, we need candidate values for the parameters θ_{ij}^u. These may simply be chosen independently from some distribution that depends on the previous sample $\hat{\theta}_{ij}^u$. Thus the candidate generating distribution becomes

$$P(\Theta|\hat{\Theta}) = \left(\prod_{ij} P(\theta_{ij}^{yu}|\hat{\theta}_{ij}^{yu})\right)\left(\prod_{ij} P(\theta_{ij}^{u}|\hat{\theta}_{ij}^{u})\right) \tag{5.8}$$

The M-H algorithm also utilizes a probability of move $\alpha(\hat{\Theta}, \Theta)$; this is defined as

$$\alpha(\hat{\Theta}, \Theta) = \min\left[\frac{P(\Theta)P(\hat{\Theta}|\Theta)}{P(\hat{\Theta})P(\Theta|\hat{\Theta})}, 1\right] \tag{5.9}$$

where $P(\Theta)$ is the parameter prior from which we wish to sample [15].

The M-H algorithm is then

- Choose some initial iterate Θ_0
- for $k = 1, 2, \ldots M$
  - Generate Θ from $P(\Theta|\Theta_k)$ and a from $Unif[0, 1]$.
  - if $a \leq \alpha(\Theta_k, \Theta)$
    - $\Theta_{k+1} = \Theta$
  - else
    - $\Theta_{k+1} = \Theta_k$
- Return $\{\Theta_k\}$.

Early samples corresponding to the transient stage of the Markov chain should of course be discarded. Once convergence to the invariant distribution occurs, samples will come from the desired parameter prior.

Note that drawing an independent vector r_{ij} for rounding each element of the adjacency matrix $\bar{\theta}_{ij}$ requires some adjustment to the error analysis of the SDP

rounded solution performed in [46], where it was assumed that the same vector r was used for all i, j. Recall, in this development Q, b summarized the linear equalities on x, the vectorized version of the constrained parameters $\bar{\theta}_{ij}$, so that we desire x to satisfy $Qx = b$. Let the matrix Z be defined as before so that $Z_{ij} = \arccos(Y^*_{ij})$ for the optimal solution of the SDP Y^*, and the constant matrices derived from Q and b are B, C, and D. This new rounding method then results in the following expected error:

(5.10)

$$E\left[\|Qx - b\|_W^2\right] = \|Qe - b\|_W^2 - \frac{1}{2\pi} Tr\left[(C - B)Z\right] + \left(\frac{1}{2\pi} Tr[DZ_n] - \frac{1}{4}\|Qe\|_W^2\right)$$

where e is a vector of ones and the terms not in parentheses comprise the error value when a single r is chosen for all i, j. The term in parentheses can be shown to lie between $-\frac{1}{4}\|Qe\|_W^2$ and $\frac{1}{4}\|Qe\|_W^2$. It follows that the solution independent error bound is in this case given by

$$E\left[\|Qx - b\|_W^2\right] \leq (1 - \alpha)\left(\|Qe\|_W^2 + \|b\|_W^2\right) + \frac{1}{4}\|Qe\|_W^2 \tag{5.11}$$

5.3 Controlled Markov Routing Model

We now derive an M-H algorithm when the controlled Markov model of [47] is used. Consider a message transmitted through the network from source $s = u^1$ to destination $d = u^2$ and along the way passing y^1 then y^2 then y^3 and so on until y^N is the final sensor passed before d is reached. The basic idea is that when a message is sitting at sensor y^i, its choice about where to go next only depends on y^i and d. The model is therefore Markovian because given y^i and d, y^{i+1} is independent of all other y^j (for notational consistency, define y^0 to be s and y^{N+1} to be d). We can use the parameters Θ to specify this model by letting $\theta^d_{y^i y^j}$ be the probability a message will go next to y^j given it is currently at y^i and its ultimate destination is d. Thus Θ

can be represented as a set of d stochastic matrices, and the required model is given by

$$P(y|u,\Theta) = \prod_{j=1}^{N} \theta^{u^2}_{y^{j-1}y^j} \tag{5.12}$$

where y^0 is taken as u^1.

We now proceed to develop an M-H algorithm for sampling from the parameter prior using the quasi-Markov routing model. This requires specifying a candidate generating distribution $P(\Theta|\hat{\Theta})$ from which the d stochastic matrices θ^d may be drawn. Instead of choosing it directly, we will describe a simple means for drawing the sample Θ given the previous sample $\hat{\Theta}$ and derive the $P(\Theta|\hat{\Theta})$ that results from this scheme.

Θ is generated from $\hat{\Theta}$ as follows:

1. Draw $\bar{\theta}^d_{ij}$ from $P(x|\hat{\theta}^d_{ij})$ independently for each d, i, j.
2. Produce θ^d for each d by setting $\theta^d_{ij} = \frac{\bar{\theta}^d_{ij}}{\sum_k \bar{\theta}^d_{ik}}$.

It is clear that this method will produce the d stochastic matrices required. In order to derive the candidate generating distribution resulting from this method, we first note that independence holds over the indices d and i, so we have

$$P(\Theta|\hat{\Theta}) = \prod_d \prod_i P(\theta^d_i|\hat{\theta}^d_i) \tag{5.13}$$

Correlation over the j index is introduced because of normalizing over columns to produce a stochastic matrix. The factors $P(\theta^d_i|\hat{\theta}^d_i)$ may be derived as follows. First define a random vector V as

$$V = \begin{pmatrix} \bar{\theta}^d_{i1} & \bar{\theta}^d_{i2} & \dots & \bar{\theta}^d_{in} & \sum_k \bar{\theta}^d_{ik} \end{pmatrix} \tag{5.14}$$

Assuming Θ was generated as described above, the distribution of V is then given by

$$P_V(v) = P(v_{n+1}|v_{1:n})P(v_{1:n}) = I_{v_{n+1}}(\sum_{k=1}^{n} v_k)\prod_{j=1}^{n} P(v_j|\hat{\theta}_{ij}^d) \tag{5.15}$$

Where $I : \Re_+ \rightarrow \{0, 1\}$ is the indicator function. The random vector W is then generated from V via the invertible transformation G so that $W = G(V)$ and G is given by

$$G(V) = \begin{pmatrix} \frac{V_1}{V_{n+1}} & \frac{V_2}{V_{n+1}} & \cdots & \frac{V_n}{V_{n+1}} & V_{n+1} \end{pmatrix} \tag{5.16}$$

and G^{-1} is given by

$$G^{-1}(W) = \begin{pmatrix} W_1W_{n+1} & W_2W_{n+1} & \cdots & W_nW_{n+1} & W_{n+1} \end{pmatrix} \tag{5.17}$$

Using a change of variables, it is simple to show that the distribution of W may be written as

$$P_W(w) = P_V(G^{-1}(w))|J(G^{-1}(w))| \tag{5.18}$$

where $|J(G^{-1}(w))|$ is the magnitude of the Jacobian of G^{-1} evaluated at w as follows

$$|J(G^{-1}(w))| = \begin{vmatrix} w_{n+1}I_n & w_{1:n} \\ 0 & 1 \end{vmatrix} = w_{n+1}^n \tag{5.19}$$

Where I_n is an identity matrix of dimension n. Since we are interested in the joint distribution of only the first n elements of W, we need only substitute the necessary ingredients into Eq. (5.18) and marginalize over W_{n+1}, thus giving

$$P_{W_{1:n}}(w) = I_1(\sum_k w_k)\int_0^\infty x^n \prod_j P(w_j x|\hat{\theta}_{ij}^d)dx \tag{5.20}$$

Resuming the original variables Θ, we substitute into Eq. (5.13) to obtain the candidate generating distribution

$$P(\Theta|\hat{\Theta}) = \prod_d \prod_i I_1(\sum_k \theta_{ik}^d)\int_0^\infty x^n \prod_j P(\theta_{ij}^d x|\hat{\theta}_{ij}^d)dx \tag{5.21}$$

With this candidate generating distribution, we define the probability of move as in Eq. (5.9). The M-H algorithm then proceeds exactly as before.

5.4 Derivation of Linear Constraints on the Adjacency Matrix

In the shortest path routing model, the linear equalities of Eqs. (5.3) and (5.4) were intuitively derived from internally sensed measurements and used to constrain sample adjacency matrices. When the adjacency matrix representation of a graph is used, one typically must consider powers of the matrix to determine existence of some specified path. For example, the existence of some n hop path between vertices (i, j) can be accertained by considering the (i, j) element of the adjacency matrix A raised to the power n. We formally derive the linear constraints in this section and show that they are necessary and sufficient for the existence of any specified path in the network.

We consider undirected graphs $G_A(V_A, E_A)$ (i.e. there is no distinction made between the edges (v_i, v_j) and (v_j, v_i)). The vertex set V_A is partitioned into the disjoint sets E_I and T, so that $V_A = E_I \cup T$ and $E_I \cap T = \phi$. Vertices in E_I are referred to as *sensor vertices*, and those in T are called *external vertices*. It is assumed that no two external vertices are adjacent, so that for all $s, d \in T$, $(s, d) \notin E_A$. In order to simplify the indexing notation in the development that follows, we assume the vertices in V_A are mapped to the natural numbers so $E_I = \{1, 2, \ldots, |E_I|\}$ and $T = \{|E_I| + 1, |E_I| + 2, \ldots, |E_I| + |T|\}$. We will utilize the adjacency matrix representation of the graph G_A, which is given by

$$A^{v_i v_j} = \begin{cases} 1 & \text{if and only if } (v_i, v_j) \in E_A \\ 0 & \text{otherwise} \end{cases} \tag{5.22}$$

It follows from the characterization of undirected edges, that the adjacency matrix is

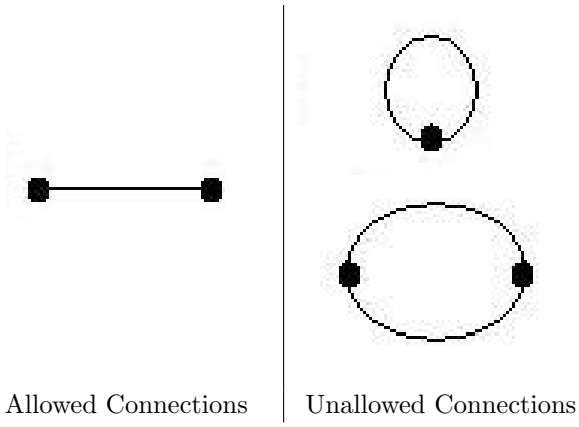


Figure 5.2: Examples of allowed and unallowed connections in undirected graphs that are uniquely represented by an adjacency matrix with all zeros along the diagonal.

always symmetric. In this chapter, we will only consider graphs G_A that are uniquely described by an adjacency matrix A with all zeros along the diagonal. Figure 5.2 illustrates some allowed and unallowed connections in such graphs.

Given some set of distinct sensor vertices p_{sd}, we wish to determine necessary and sufficient conditions on the adjacency matrix A so that p_{sd} describes a path between distinct external vertices s and d. These conditions are based on the following definitions of a path and the related concept of a cycle.

Definition V.1. Let $p_{sd} = \{v_i\}_{i=1}^{h} \subseteq E_I$ be a set of distinct sensor vertices associated with the external vertices $s, d \in T$ with $s \neq d$. In the case that $h = 1$, p_{sd} is a **path** if $(s, v_1) \in E_A$ and $(v_1, d) \in E_A$. In the case that $h > 1$, p_{sd} is a **path** if there is some permutation vector π such that $(s, v_{\pi^1}) \in E_A$, $(s, v) \notin E_A$ for $v \in p_{sd} - v_{\pi^1}$, $(v_{\pi^h}, d) \in E_A$, $(v, d) \notin E_A$ for $v \in p_{sd} - v_{\pi^h}$, $(v_{\pi^i}, v_{\pi^{i+1}}) \in E_A$ for $i = 1, \ldots, h-1$, and $(v_{\pi^i}, v_{\pi^j}) \notin E_A$ for $|i - j| > 1$.

Definition V.2. Let $p = \{v_i\}_{i=1}^{h} \subseteq E_I$ be a set of distinct sensor vertices. p is a **cycle** if $h \geq 3$ and there is some permutation vector π such that $(v_{\pi^i}, v_{\pi^{(i \mod h)+1}}) \in E_A$ for $i = 1, 2, \ldots, h$ and $(v_{\pi^i}, v_{\pi^j}) \notin E_A$ for $|(i \mod h) - (j \mod h)| > 1$.

Note that in the preceding definitions and in the results to follow, superscripts

are used to index vector elements. For example, if the permutation vector is given by $\pi = (3, 1, 4, 5, 2)$, then $\pi^1 = 3$, $\pi^2 = 1$, $\pi^3 = 4$, etc. A dichotomy between a paths and cycles is observed in Lemma V.3.

Lemma V.3. *If p_{sd} is a path, there is no subset of p_{sd} that is a cycle.*

Proof. We need only consider connected subsets containing at least three elements as candidates for cycles. However, the only such subsets from p_{sd} are of the form $\{v_{\pi^i}, v_{\pi^{i+1}}, \ldots, v_{\pi^{i+k}}\}$ with $k \geq 2$, and Definition V.1 requires $(v_{\pi^{i+k}}, v_{\pi^i}) \notin E_A$. Hence no subset of p_{sd} is a cycle. □

Without much effort, we can use Definition V.1 to write down linear constraints on the adjacency matrix implied by the existence of a path. In fact, when $h = |p_{sd}| \leq 3$ these constraints are necessary and sufficient for p_{sd} to be a path. This result is given in Theorem V.4.

Theorem V.4. *If $p_{sd} = \{v_i\}_{i=1}^h$ is a path, then the following linear constraints on the adjacency matrix A are satisfied:*

$$\begin{aligned} &\sum_{v \in p_{sd} \cup \{s,d\}} A^{vv_i} = 2 \text{ for all } i = 1, \ldots, h \\ &\sum_{v \in p_{sd}} A^{sv} = 1 \\ &\sum_{v \in p_{sd}} A^{vd} = 1 \end{aligned} \tag{5.23}$$

If $h \leq 3$ and the constraints in Eq. (5.23) are satisfied, then p_{sd} is a path.

Proof. It is trivial to show the forward implication holds in the case of $h = 1$, so consider when $h > 1$. The only nonzero term in the second equation is $A^{sv_{\pi^1}}$, and the only nonzero term in the third equation is $A^{v_{\pi^h}d}$ so these constraints readily follow. The first constraint for v_{π^1} has nonzero terms $A^{sv_{\pi^1}}$ and $A^{v_{\pi^2}v_{\pi^1}}$. For v_{π^h},

the first constraint has nonzero terms $A^{dv_{\pi^h}}$ and $A^{v_{\pi^{h-1}}v_{\pi^h}}$. If $h > 2$, then for each $i = 2, \ldots, h-1$, the nonzero terms are $A^{v_{\pi^{i-1}}v_{\pi^i}}$ and $A^{v_{\pi^{i+1}}v_{\pi^i}}$.

In order to show the reverse implication, note that the second constraint implies there is exactly one element from p_{sd}, call it v_{π^1}, such that $(s, v_{\pi^1}) \in E_A$. Define the set $V_1 = p_{sd} - v_{\pi^1}$. If V_1 is nonempty and there is an element from V_1 adjacent to v_{π^1}, call this element v_{π^2}–the first constraint ensures there is at most one element from V_1 adjacenct to v_{π^1} since we have already established s is adjacent to v_{π^1}–and define $V_2 = V_1 - v_{\pi^2}$. If V_2 is nonempty and there is an element from V_2 adjacent to v_{π^2}, call this element v_{π^3} and define $V_3 = V_2 - v_{\pi^3}$. Suppose this process terminates after assigning v_{π^j} for $1 \leq j \leq 3$ either because V_j is empty or there is no element from V_j adjacent to v_{π^j}. Since the process has terminated and each vertex $s, v_{\pi^1}, \ldots, v_{\pi^{j-1}}$ has already had its adjacent vertices assigned, the first constraint for v_{π^j} requires that $(v_{\pi^j}, d) \in E_A$. It then must be that the process terminated because of V_j being empty since the fact that $|V_j| \leq 2$ prevents the first constraint from being satisfied for members of V_j. We have thus constructed a path. □

The crucial point in proving the reverse implication of Theorem V.4 required the set V_j have no more than two members. This prevents the sum over only elements of V_j in the first constraint of Eq. (5.23) from totaling two. If V_j were to have three or more members, this equality implies the existence of cycles, as indicated by Lemma V.5. We must then add additional constraints to prevent cycles for necessary and sufficient conditions in the case that $h > 3$.

Lemma V.5. *If $p \subseteq E_I$ is a set of distinct vertices in G_A for which the following equality is satisfied, then some subset of p is a cycle.*

$$\sum_{v \in p} A^{vv_i} = 2 \text{ for all } v_i \in p \tag{5.24}$$

Proof. It must be the case that $|p| \geq 3$ for the contraints in Eq. (5.24) to hold. Choose an arbitrary vertex from p, call it v_{π^1}, and define the set $V_1 = p - v_{\pi^1}$. Select v_{π^2} as a vertex from V_1 with $(v_{\pi^1}, v_{\pi^2}) \in E_A$, and define $V_2 = V_1 - v_{\pi^2}$. There is now exactly one vertex in V_2 that is adjacent to v_{π^2} since $A^{v_{\pi^1} v_{\pi^2}}$ accounts for one nonzero term in the constraint of Eq. (5.24) for v_{π^2}. Let this vertex be v_{π^3} and define $V_3 = V_2 - v_{\pi^3}$. Continue this process of choosing $v_{\pi^{i+1}}$ from V_i so that $(v_{\pi^i}, v_{\pi^{i+1}}) \in E_A$ and setting $V_{i+1} = V_i - v_{i+1}$ until termination at some step j with $3 \leq j \leq |p|$–at which point either V_j is empty or there is no vertex $v \in V_j$ with $(v_{\pi^j}, v) \in E_A$. It must then be the case that $(v_{\pi^j}, v_{\pi^1}) \in E_A$ since each vertex $v_{\pi^2}, v_{\pi^3}, \ldots, v_{\pi^{j-1}}$ has already had its adjacent vertices assigned. We have then constructed a cycle using elements from p. □

We now specify linear conditions on the adjacency matrix A that eliminates every conceivable cycle using elements from p_{sd}. In order to enumerate every possible cycle, we use the vectors $\pi_{lm|hk}$, where l and m index the particular cycle with k distinct vertices selected from $h = |p_{sd}|$ total vertices. Index m indicates the specific choice of k distinct vertices included in the cycle and therefore ranges from 1 to $\binom{h}{k}$, while index l indicates the particular cycle for those k vertices and thus ranges from 1 to $\frac{1}{2}(k-1)!$ for undirected graphs. For example, suppose $h = 4$ and we wish to enumerate all cycles consisting of three distinct vertices ($k = 3$) using this notation. These are given in Eq. (5.25).

$$\pi_{11|43} = (1,2,3) \quad \pi_{12|43} = (1,2,4) \quad \pi_{13|43} = (1,3,4) \quad \pi_{14|43} = (2,3,4) \tag{5.25}$$

With this notation, we prove necessary and sufficient conditions on the adjacency matrix for preventing cycles.

Lemma V.6. *Let $p_{sd} = \{v_i\}_{i=1}^h$ be a set of distinct vertices in G_A with $h \geq 4$.*

There does not exist a cycle consisting only of $h-1$ or fewer elements from p_{sd} if and only if the following conditions on the adjacency matrix A are satisfied for all $m = 1, 2, \ldots, \binom{h}{k}$, $l = 1, 2, \ldots, \frac{1}{2}(k-1)!$, and $k = 3, 4, \ldots, h-1$.

$$\sum_{i=1}^{k} A^{v_{\pi^i_{lm|hk}} v_{\pi^{(i \bmod k)+1}_{lm|hk}}} \leq k-1 \tag{5.26}$$

Proof. For the forward implication, suppose the inequality in Eq. (5.26) is violated for some l_*, m_*, k_*. It follows that $A^{v_{\pi^i_{l_*m_*|hk_*}} v_{\pi^{(i \bmod k_*)+1}_{l_*m_*|hk_*}}} = 1$ for all $i = 1, 2, \ldots, k_*$. Thus a cycle consisting of k_* unique elements from p_{sd} is given by $\{v_{\pi^1_{l_*m_*|hk_*}}, v_{\pi^2_{l_*m_*|hk_*}}, \ldots, v_{\pi^{k_*}_{l_*m_*|hk_*}}\}$.

For the reverse implication, suppose p is a cycle consisting only of elements from p_{sd}. Since all distinct cycles were enumerated by the π vectors, it must be the case that there is some l_*, m_* such that this cycle is specified by $\pi_{l_*m_*|hk_*}$. It follows that $A^{v_{\pi^i_{l_*m_*|hk_*}} v_{\pi^{(i \bmod k_*)+1}_{l_*m_*|hk_*}}} = 1$ for all $i = 1, 2, \ldots, k_*$ and so $\sum_{i=1}^{k_*} A^{v_{\pi^i_{l_*m_*|hk_*}} v_{\pi^{(i \bmod k_*)+1}_{l_*m_*|hk_*}}} = k_*$, thus violating the inequality constraint. □

Augmenting the constraints in Eq. (5.23) with those in Eq. (5.26) then provides necessary and sufficient linear constraints on the adjacency matrix for p_{sd} to be a path in the case that $|p_{sd}| \geq 4$. This result is given in Theorem V.7.

Theorem V.7. *Let $p_{sd} = \{v_i\}_{i=1}^{h}$ with $h \geq 4$ be a set of sensor vertices in G_A. p_{sd} is a path if and only if the constraints in Eq. (5.23) and Eq. (5.26) are satisfied for the adjacency matrix A.*

Proof. Suppose p_{sd} is a path, Theorem V.4 then gives that the constraints in Eq. (5.23) are satisfied. Also, from Lemma V.3, the fact that p_{sd} is a path prevents any subset of p_{sd} from being a cycle, thus implying the conditions in Eq. (5.26).

For the reverse implication, proceed exactly as in the corresponding proof in Theorem V.4 to select the vertices v_{π^1} and v_{π^2} and define the sets V_1 and V_2. Continue

this process of choosing $v_{\pi^{i+1}}$ from V_i so that $(v_{\pi^i}, v_{\pi^{i+1}}) \in E_A$ and setting $V_{i+1} = V_i - v_{i+1}$ until termination at some step j–at which point either V_j is empty or there is no vertex $v \in V_j$ with $(v_{\pi^j}, v) \in E_A$. As argued in Theorem V.4, it must be that $(v_{\pi^j}, d) \in E_A$, $|V_j| \neq 1$, and $|V_j| \neq 2$. It also cannot be that $|V_j| \geq 3$, because in this case the result of Lemma V.5 would imply a cycle and thus, by Lemma V.6, violate the constraints in Eq. (5.26). We must have that V_j is empty, and therefore p_{sd} is a path. □

It is desired to express the problem of finding an adjacency matrix consistent with some internally sensed measurements as follows

$$\begin{aligned} &\text{find } x \in \{0,1\}^n \\ &\text{such that } Qx = b \end{aligned} \tag{5.27}$$

where Q and b are derived from the constraints in Eq. (5.23) and (if $|p_{sd}| \geq 4$) Eq. (5.26). We introduce 0-1 slack variables to adapt the inequality constraints of Eq. (5.26) into this form. For each distinct inequality constraint specified by a permutation $\pi_{lm|hk}$, $k-1$ 0-1 slacks must be introduced. Thus the number of linear equality constraints introduced by each measured path $p_{sd} = \{v_i\}_{i=1}^h$ is given by

$$h + 2 + \frac{1}{2}\sum_{k=3}^{h-1} \frac{h!}{k(h-k)!} \tag{5.28}$$

where the second term is only included if $h \geq 4$. Also, in the case that 0-1 slack variables are needed, the number of such variables is given by

$$\frac{1}{2}\sum_{k=3}^{h-1} \frac{(k-1)h!}{k(h-k)!} \tag{5.29}$$

The formulas in Eqs. (5.28) and Eqs. (5.28) are evaluated for $h = 1, 2, \ldots, 7$ in Table 5.1. For $h > 5$, the number of constraints/slacks starts becoming unreasonable.

No. of Hops, h	No. of Constraints	No. of 0-1 Slacks
1	3	0
2	4	0
3	5	0
4	10	8
5	32	65
6	145	463
7	821	3493

Table 5.1: Number of equality constraints on the adjacency matrix and 0-1 slack variables that must be added to ensure p_{sd} is a path. Due to the combinatorial nature of the constraints in Eq. (5.26), the problem becomes too large for $h = |p_{sd}| > 5$.

5.5 Conclusion

This chapter derives Markov chain Monte Carlo methods for network endpoint estimation based on models proposed in [46] and [47]. Although MCMC algorithms are frequently used tools for estimation, a major drawback that must be kept in mind is the massive computational burden. The M-H methods in this chapter, however, might be investigated further in the future and compared to existing endpoint localization techniques.

CHAPTER VI

Conclusion

6.1 Remarks

This thesis developed several models and algorithms for network endpoint localization. Chapter 2 focused on the derivation and computation of a general metric for labeled networks. Chapters 3, 4, and 5 proposed methods specifically for the problem of endpoint estimation from partially ordered sensor activation sets. The focus of Chapter 3 is on a Monte Carlo approach driven by a semidefinite programming method. Chapter 4 develops faster, recursive tools that are suitable for online implementation. Finally, Chapter 5 gives an outline of Markov chain Monte Carlo methods based on the models of the previous chapters. It would be very exciting to the author if any or all of the tools described in this thesis found their way into real systems. It is not an unreal possibility, given that these results sprang from an application of interest to our sponsors.

6.2 Future Work

The concluding sections of each individual chapter often provides some directions for future work. As we see it, the most pressing of these is the issue of probe scheduling. This is alluded to in Chapters 3 and 4, with a concrete approach based on a multiarmed bandit model given in 4. Careful probe scheduling may result in

marked improvement of endpoint estimates, a point we hope will be corroborated down the road by others interested in this problem.

Dissertations are not finished; they are abandoned.

–Prof. Fred Brooks, University of North Carolina

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

[2] H.A. Almohamad and S.O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(5):522–525, May 1993.

[3] P. Auer, N. Cesa-Bianchi, Y. Freund, and R.E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

[4] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, Oct. 1999.

[5] E. Barcucci, A. Del Lungo, E. Pergola, and R. Pinzani. ECO: a methodology for the enumeration of combinatorial objects. *Journal of Difference Equations and Applications*, 5:435–490, 1999.

[6] P. Bergamini, L. Cinque, A.D.J. Cross, and E.R. Hancock. Efficient alignment and correspondence using edit distance. *SSPR/SPR*, pages 246–255, 2000.

[7] M. Berkelaar, K. Eikland, and P. Notebaert. lp_solve: open source (mixed-integer) linear programming system, May 2004.

[8] G. Beylkin and M. Mohlenkamp. Numerical operator calculus in higher dimensions. *Proc. National Academy of Sciences*, 99(16):10246–10251, Aug. 2002.

[9] B. Borchers. CSDP: A C library for semidefinite programming. *Optimization Methods and Software*, 11(1):613–623, 1999.

[10] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, 2004.

[11] H. Bunke. Error correcting graph matching: on the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):917–922, Sept. 1999.

[12] H. Bunke. Recent developments in graph matching. *Proc. 15th Intl. Conf. on Pattern Recognition*, 2:117–124, Sept. 2000.

[13] H. Bunke and K. Shearer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19:255–259, 1998.

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

[15] S. Chib and E. Greenberg. Understanding the Metropolis-Hastings algorithm. *The American Statistician*, 49(4):327–335, Nov. 1995.

[16] F. Chung, R. Graham, V. Hoggatt, and M. Kleiman. The number of Baxter permutations. *Journal of Combinatorial Theory, Ser. A*, 24:382–394, 1978.

[17] M. Coates, R. Castro, and R. Nowak. Maximum likelihood network topology identification from edge-based unicast measurements. *ACM Sigmetric 2002*, June 2002.

[18] M. Coates, A.O. Hero, R. Nowak, and B. Yu. Internet tomography. *IEEE Signal Processing Magazine*, 19(3):47–65, May 2002.

[19] M.M. Cone, R. Venkataraghavan, and F.W. McLafferty. Molecular structure comparison program for the identification of maximal common substructures. *J. American Chem. Society*, 99(23):7668–7671, Nov. 1977.

[20] J.H. Conway and N.J.A. Sloane. *Sphere Packings, Lattices and Groups*. Springer-Verlag, New York, 1988.

[21] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1367–1372, Oct. 2004.

[22] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[23] G.M. Downs and P. Willett. Similarity searching in databases of chemical structures. In K.B. Lipkowitz and D.B. Boyd, editors, *Reviews in Computational Chemistry, Vol. 7*, pages 1–66. VCH Publishers, Inc., New York, 1996.

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

[25] B.H. Dunford-Shore, W. Sulaman, B. Feng, F. Fabrizio, J. Holcomb, W. Wise, and T. Kazic. Klotho: Biochemical compounds declarative database, 2002.

[26] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the Internet topology. *Proc. of ACM SIGCOMM'99*, pages 251–262, Aug. 1999.

[27] M.-L. Fernandez and G. Valiente. A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22:753–758, 2001.

[28] M. Figueiredo and R. Nowak. An EM algorithm for wavelet-based image reconstruction. *IEEE Transactions on Image Processing*, 12(8):906–916, Aug. 2003.

[29] M. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, San Francisco, CA, 1979.

[30] M.X. Goemans and F. Rendl. Semidefinite programming in combinatorial optimzation. In H. Wolkowicz, R. Saigal, and L. Vandenberghe, editors, *Handbook of Semidefinite Programming*, pages 343–360. Kluwer Academic Publishers, 2000.

[31] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, Nov. 1995.

[32] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(4):377–387, Apr. 1996.

[33] S.M. Gomez and A. Rzhetsky. Towards the prediction of complete protein-protein interaction networks. *Proc. Pacific Symp. on Biocomputing*, pages 413–424, 2002.

[34] M. Gori, M. Maggini, and L. Sarti. Exact and approximate graph matching using random walks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1100–1111, July 2005.

[35] T.R. Hagadone. Molecular substructure similarity searching: efficient retrieval in two-dimensional structure databases. *J. Chem. Inf. Comput. Sci.*, 32:515–521, 1992.

[36] G. Harper, G.S. Bravi, S.D. Pickett, J. Hussain, and D.V.S. Green. The reduced graph descriptor in virtual screening and data-driven clustering of high-throughput screening data. *J. Chem. Inf. Comput. Sci.*, 44:2145–2156, 2004.

[37] W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[38] C. Helmberg. Fixing variables in semidefinite relaxations. *SIAM J. Matrix Anal. and Apps.*, 21(3):952–969, 2000.

[39] C. Helmberg, F. Rendl, R.J. Vanderbei, and H. Wolkowicz. An interior-point method for semidefinite programming. *SIAM J. Optimization*, 6(2):342–361, May 1996.

[40] K.J. Hintz. A measure of the information gain attributable to cueing. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(2):237–244, 1991.

[41] A. Hlaoui and S. Wang. A new algorithm for inexact graph matching. *Proc. 16th Intl. Conf. on Pattern Recognition*, 4:180–183, 2002.

[42] L. Jianzhuang and L.Y. Tsui. Graph-based method for face identification from a single 2d line drawing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1106–1119, 2000.

[43] M. Johnson, M. Naim, V. Nicholson, and C.C. Tsai. Unique mathematical features of the substructure metric approach to quantitative molecular similarity analysis. In R.B. King and D.H. Rouvray, editors, *Graph Theory and Topology in Chemistry*, pages 219–225, Mar. 1987.

[44] M.A. Johnson and G.M. Maggiora, editors. *Concepts and Applications of Molecular Similarity*. John Wiley and Sons, New York, 1990.

[45] D. Justice and A. Hero. A binary linear programming formulation of the graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1200–1214, Aug. 2006.

[46] D. Justice and A. Hero. Estimation of message source and destination from network intercepts. *IEEE Transactions on Information Forensics and Security*, 1(3), Sept. 2006.

[47] D. Justice and A. Hero. Online methods for network endpoint localization. Technical report, Dept. of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, July 2006.

[48] H. Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, NJ, 3 edition, 2002.

[49] P. Klein, S. Tirthapura, D. Sharvit, and B. Kimia. A tree-edit distance algorithm for comparing simple, closed shapes. *Proc. ACM-SIAM Symp. Disc. Algorithms*, pages 696–704, 2000.

[50] C. Kreucher, K. Kastella, and A. Hero. Sensor management using an active sensing approach. *Signal Processing*, 85(3):607–624, Mar. 2005.

[51] P.R. Kumar and P. Varaiya. *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1986.

[52] E. Lawrence, G. Michailidis, and V. Nair. Local area network analysis using end-to-end delay tomography. *Performance Evaluation Review*, 33:39–45, 2005.

[53] E. Lawrence, G. Michailidis, V. Nair, and B. Xi. Network tomography: A review and recent developments. In Fan and Koul, editors, *Frontiers in Statistics*, 2006.

[54] D. Lee and M. Yannakakis. Principles and methods of testing finite state matchines - a survey. *Proc. of the IEEE*, 84(8):1090–1123, Aug. 1996.

[55] G. Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcolo*, 9:341–352, 1972.

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

[57] C.-J. Lin and R. Saigal. A predictor corrector method for semidefinite linear programming. Technical Report TR95-20, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, Oct. 1995.

[58] J. Llados, E. Marti, and J.J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(10):1137–1143, 2001.

[59] L. Lovasz and A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1(2):166–190, 1991.

[60] D.G. Luenberger. *Optimization by Vector Space Methods.* John Wiley and Sons, Inc., New York, 1969.

[61] B.D. McKay. Practical graph isomorphism. *Congressus Numerantium*, 30:45–87, 1981.

[62] B. Messmer and H. Bunke. Error-correcting graph isomorphism using decision trees. *Int. Journal of Pattern Recognition and Art. Intelligence*, 12:721–742, 1998.

[63] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Physics*, 21:1087–1092, 1953.

[64] S. Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[65] E.F. Moore. Gedanken-experiments on sequential machines. In *Automata Studies, Annals of Mathematics Studies*, number 34, pages 129–153. Princeton University Press, Princeton, N.J., 1956.

[66] Q.D. Morris, B.J. Frey, and C.J. Paige. Denoising and untangling graphs using degree priors. *Proc. Neural Info. Proc. Sys.*, (16), 2003.

[67] R. Myers, R.C. Wilson, and E.R. Hancock. Bayesian graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):628–635, June 2000.

[68] M.V. Nayakkankuppam and Y. Tymofyeyev. A parallel implementation of the spectral bundle method for large-scale semidefinite programs. *Proc. of the 8th SIAM Conf. on App. Lin. Alg.*, 2003.

[69] M. Neuhaus and H. Bunke. A probabilistic approach to learning costs for graph edit distance. *Proc. 17th Intl. Conf. on Pattern Recognition*, 3:389–393, 2004.

[70] M.E.J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[71] P. Ostergard. A fast algorithm for the maximum clique problem. *Discrete Appl. Math.*, 120:197–207, 2002.

[72] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1982.

[73] A. Pasztor and D. Veitch. A precision infrastructure for active probing. *Proc. Workshop on Passive and Active Networking*, Apr 2001.

[74] A. Pasztor and D. Veitch. PC based precision timing without GPS. *Proc. ACM SIGMETRICS*, June 2002.

[75] M. Pavel. *Fundamentals of Pattern Recognition*. Marcel Dekker, New York, 1989.

[76] T. Pavlidis. *Structural Pattern Recognition*. Springer-Verlag, New York, 1977.

[77] M. Rabbat, M.A.T. Figueiredo, and R. Nowak. Network inference from co-occurrences. Technical Report ECE-06-2, Department of Electrical and Computer Engineering, Univ. of Wisconsin, Madison, WI, April 2006.

[78] M. Rabbat and R. Nowak. Telephone network topology inference. Technical report, Univ. of Wisconsin, Madison, WI, Dec 2004.

[79] J.W. Raymond, E.J. Gardiner, and P. Willett. Rascal: calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, 45(6):631–644, 2002.

[80] J.W. Raymond and P. Willett. Effectiveness of graph-based and fingerprint-based similarity measures for virtual screening of 2d chemical structure databases. *J. Computer-Aided Molecular Design*, 16:59–71, 2002.

[81] J.W. Raymond and P. Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *J. Computer-Aided Molecular Design*, 16:521–533, 2002.

[82] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, New York, 1996.

[83] H. Robbins. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.*, 55:527–535, 1952.

[84] A. Robles-Kelly and E.R. Hancock. Graph edit distance from spectral seriation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(3):365–378, Mar. 2005.

[85] R. Saigal. *Linear Programming: A Modern Integrated Analysis*. Kluwer Academic Publishers, Boston, 1995.

[86] T.B. Sebastian, P.N. Klein, and B.B. Kimia. Recognition of shapes by editing their shock graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):550–571, May 2004.

[87] D. Shasha, J.T.L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *Proc. 21st ACM SIGMOD-SIGACT-SIGART*, Madison, WI, June 2005.

[88] M.-F. Shih and A.O. Hero. Unicast-based inference of network link delay distributions with finite mixture models. *IEEE Transactions on Signal Processing*, 51(8):2219–2228, Aug. 2003.

[89] H. Steck and T.S. Jaakkola. On the Dirichlet prior and Bayesian regularization. Technical Report 2002-014, Artificial Intelligence Lab, MIT, Cambridge, MA, Sept. 2002.

[90] A.S. Tanenbaum. *Computer Networks*. Prentice Hall PTR, Upper Saddle River, NJ, 3 edition, 1996.

[91] D.M. Titterington. Recursive parameter estimation using incomplete data. *J. Royal Statistical Society, Series B*, 46(2):257–267, 1984.

[92] A. Torsello, D. Hidovic-Rowe, and M. Pelillo. Polynomial-time metrics for attributed trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1087–1099, July 2005.

[93] J.R. Treichler, M.G. Larimore, S.L. Wood, and M. Rabbat. Determining the topology of a telephone system using internally sensed network tomography. *Proc. of 11th Digital Signal Processing Workshop*, Aug. 2004.

[94] W. Tsai and K. Fu. Error-correcting isomorphisms of attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 9:757–768, 1979.

[95] Y. Tsang, M. Coates, and R. Nowak. Network delay tomography. *IEEE Transactions on Signal Processing*, 51(8):2125–2136, Aug. 2003.

[96] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):695–703, Sept. 1988.

[97] B.J. van Wyk and M.A. van Wyk. A pocs-based graph matching algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1526–1530, Nov. 2004.

[98] Y. Vardi. Network tomography: estimating the source-destination traffic intensities from link data. *J. Amer. Stat. Assoc.*, 91:365–377, 1996.

[99] V. Vatter. Finitely labeled generating trees and restricted permutations. *Journal of Symbolic Computation*, 41:559–572, 2006.

[100] R. Wagner and M. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.

[101] W.D. Wallis, P. Shoubridge, M. Kraetz, and D. Ray. Graph distances using graph union. *Pattern Recognition Letters*, 22:701–704, 2001.

[102] Z. Wang and K. Zhang. Alignment between two rna structures. *MFCS*, pages 690–702, 2001.

[103] P. Willett. *Clustering in Chemical Information Systems*. Research Studies Press, Letchworth, 1987.

[104] P. Willett. Matching of chemical and biological structures using subgraph and maximal common subgraph isomorphism algorithms. *IMA Vol. Math. Appl.*, 108:11–38, 1999.

[105] P. Willett and V. Winterman. A comparison of some measures for the determination of inter-molecular stuctural similarity. *Quant. Struct.-Act. Relat.*, 5, 1986.

[106] C. Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95–103, Mar. 1983.

[107] K. Zhang. A constrained edit distance between unordered labeled trees. *Algorithmica*, 15(6):205–222, 1996.

ABSTRACT

Inference Methods for Message Endpoint Localization in Networks

by

Derek H. Justice

Chair: Alfred O. Hero III

People often build or organize networks in order to establish lines of communication. The subjects might utilize a telephone or computer network, or perhaps even something much more low-tech where certain individuals are designated to deliver messages in person. When certain parties of interest are communicating, it is desirable to monitor these networks in order to discover their motives, identities, and locations. Presently, government and private agencies are investing heavily in the development of equipment for network surveillance and algorithms for gleaning useful information from collected data. This thesis develops several tools for inference in networks with a focus on determining the locations of the sender and receiver of an intercepted message.

We begin by deriving a distance metric that allows comparisons between different network topologies. The metric quantifies the distance between networks by the total cost of edit operations (such as node or link insertion or deletion) necessary to make

the two networks isomorphic. We derive this graph edit distance through a sort of embedding scheme, and show how to compute it with a binary linear program. Upper and lower bounds are computable in polynomial time through relaxation to an assignment problem and standard linear programming, respectively.

We move next to the estimation of an intercepted message's source and destination in a network of unknown topology. Sensors placed on some links or nodes in the network are capable of indicating whenever a specific message passes their assigned elements with a limited degree of timing precision. The source and destination (endpoints) are localized using a possibly unordered sensor activation pattern along with some prior information on the unknown network topology. We first use a semidefinite programming driven Monte Carlo approach to build approximate endpoint posterior distributions. Maximum a posteriori endpoint estimates can then be read directly from these. Next we utilize a hierarchical Bayesian model and a recursive expectation-maximization algorithm to develop online techniques for endpoint localization. Finally, some preliminary derivations are given for the application of well-known Markov chain Monte Carlo algorithms to this problem.