University of Massachusetts Amherst
ScholarWorks@UMass Amherst

Open Access Dissertations

9-2010

Bounds on Service Quality for Networks Subject to Augmentation and Attack

George Dean Bissias
University of Massachusetts Amherst, gbiss@cs.umass.edu

Follow this and additional works at: https://scholarworks.umass.edu/open_access_dissertations

 Part of the Computer Sciences Commons

Recommended Citation
Bissias, George Dean, "Bounds on Service Quality for Networks Subject to Augmentation and Attack" (2010). *Open Access Dissertations*. 264.
https://scholarworks.umass.edu/open_access_dissertations/264

This Open Access Dissertation is brought to you for free and open access by ScholarWorks@UMass Amherst. It has been accepted for inclusion in Open Access Dissertations by an authorized administrator of ScholarWorks@UMass Amherst. For more information, please contact scholarworks@library.umass.edu.

BOUNDS ON SERVICE QUALITY FOR NETWORKS SUBJECT TO AUGMENTATION AND ATTACK

A Dissertation Presented

by

GEORGE DEAN BISSIAS

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2010

Computer Science

© Copyright by George Dean Bissias 2010

All Rights Reserved

BOUNDS ON SERVICE QUALITY FOR NETWORKS SUBJECT TO AUGMENTATION AND ATTACK

A Dissertation Presented

by

GEORGE DEAN BISSIAS

Approved as to style and content by:

Brian Neil Levine, Chair

Lixin Gao, Member

Ramesh Kumar Sitaraman, Member

Don Towsley, Member

Andrew Barto, Department Chair
Computer Science

ACKNOWLEDGMENTS

This work is the product of collaborations with many students and faculty at the University of Massachusetts. Professor Arnold L. Rosenberg provided invaluable guidance during my initial investigation into graph vulnerability. Professor Ramesh K. Sitaraman spent many hours with me discussing graph vulnerability and augmentation, and he helped guide the formulation and treatment of the Dynamic Network Augmentation Problem. I would like to thank Professor Mark Corner and John Burgess for allowing me to contribute to their work on DTN attack vulnerability. Finally, for my advisor Brian Neil Levine, there is no way to fully convey his contribution to this thesis or my professional development. He has devoted time, funding, and patience to my case for the past seven years, and has always been supportive of my ideas: good and bad. He is not manager or an employer. He is a scholar and *true* mentor.

ABSTRACT

BOUNDS ON SERVICE QUALITY FOR NETWORKS SUBJECT TO AUGMENTATION AND ATTACK

SEPTEMBER 2010

GEORGE DEAN BISSIAS

B.Sc., UNIVERSITY OF NEW MEXICO

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Brian Neil Levine

Assessing a network's vulnerability to attack and random failure is a difficult and important problem that changes with network application and representation. We furnish algorithms that bound the robustness of a network under attack. We utilize both static graph-based and dynamic trace-driven representations to construct solutions appropriate for different scenarios.

For static graphs we first introduce a spectral technique for developing a lower bound on the number of connected pairs of vertices in a graph after edge removal, which we apply to random graphs and the power grid of the Philippines. To address the problem of resource availability in networks we develop a second technique for

bounding the number of nominally designated *client* vertices that can be disconnected from all *server* vertices after either edge or vertex removal (or both). This algorithm is also tested on the power grid and a wireless mesh network, the Internet AS level graph, and the highway systems of Iowa and Michigan.

Dynamic networks are modeled as disruption tolerant networks (DTNs). DTNs are composed of mobile nodes that are intermittently connected via short-range wireless radios. In the context of both human and vehicular mobility networks we study both the effect of targeted node removal and the effect of augmentation with stationary relays.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Graph *vulnerability* has been defined in many ways. It can refer to random failure or intentional disruption, may be measured according to countless metrics, and be constrained to vertex or edge removal. Depending on the definition, the problem can be shown to be NP-complete or polynomial in terms of time complexity. The diversity of problem statements reflects the diversity of motivation for its study. A chemist may wish to measure the porosity of a material, an electrical engineer the redundancy of a circuit, a biologist the robustness of a neural system, or a computer scientist the fault tolerance of a network. We define vulnerability problems that are well suited to studying the latter problem, network fault tolerance. However, the problem is general enough that its impact likely extends beyond the networking community. We also investigate how graphs respond to *augmentation* in the context of network throughput. Graph augmentation is a dual problem to the vulnerability problem that addresses the addition of vertices or edges to a graph.

Many results concerning graph vulnerability have come from the mathematics literature. Percolation theory deals with random failure in random graphs, usually of infinite size. The theory of graph separators offers bounds on the number of vertices or edges that must be removed in order to achieve a certain balance in component

sizes. Unfortunately, these techniques fail to directly capture the vulnerability of most real-world graphs. We introduce algorithms that address the shortcomings of earlier work. In particular, we offer heuristics for deriving upper and lower bounds on vulnerability in *any* graph and subject to a variety of metrics with broad appeal to the networking community. To measure the quality and provide context for these heuristics, our work is validated empirically and is accordingly supplemented with experiments on real and synthetic graphs.

There are numerous network models that operate with varying degrees of sophistication. At one extreme are standard graph models, which are capable of representing the connectivity of static networks. On the end, are network simulators, which represent the dynamics of both connectivity and packet movement in a given network. Each is appropriate under different circumstances that depend on network topology, dynamics, and deployment. We study vulnerability and augmentation problems in three classes of network model: the standard graph $G = (V, E)$, client-server graph $G = (S \cup C, E)$, and disruption tolerant network (DTN) labeled $\mathcal{D} = (N, K, C)$. We use standard notation for graphs, each vertex in the set V corresponds to a node in the network and each edge in the set E corresponds to an undirected link. Client-server graphs are identical to standard graphs except that the vertex set is divided into a set of servers S and clients C. Under this model, we assume that some homogenous resource is available at every vertex in S and all clients in C desire access to this resource. The graph is not bipartite, that is to say, clients are generally adjacent to both servers and other clients. Both standard and client server graphs are most appropriate for modeling connectivity in static networks.

A DTN is a network comprised of mobile nodes that are only intermittently connected. Applications that can tolerate short disruptions can be implemented over such networks using routing algorithms that are robust to packet delay and route failure. Here we define DTNs by a collection of nodes N, node positions K, and connection events C. Positions are triples with $(n, k, t) \in K$ indicating that node n is at location k at time t. Two nodes in $n_1, n_2 \in N$ are considered to be *connected* when they share the same location at a given time. This occurs when there exists some time t and location k such that both $(n_1, k, t) \in K$ and $(n_2, k, t) \in K$. We call such occurrences *connection events* and label them with quadruples $(n_1, n_2, b, t) \in C$ indicating that nodes n_1 and n_2 can pass b bits of data at time t (in both directions).

Each of the models: graph, client-server graph, and DTN operate with a unique currency, which is the metric that best captures service quality in that model. We utilize the following metrics throughout this document,

- **Standard Graphs:** $Pairs(G)$, the number of pairs of vertices that are connected in graph G.
- **Client-Server Graphs:** $Conn(G)$, the number of clients connected to some server in client server graph G.
- **Disruption Tolerant Networks:**
  - $Thru(\mathcal{D}, M)$, the maximum aggregate throughput between nodes in DTN $\mathcal{D}$ given end-to-end demand matrix M.

– $Delay(\mathcal{D}, M, P^*)$, the minimum aggregate delay between nodes in DTN $\mathcal{D}$ given end-to-end demand matrix M, and minimum aggregate throughput P^*.

To begin, we present formal statements for each of the problems studied, and introduce notation crucial to our analysis. Our work is divided between the study of connectivity in graphs on one hand and throughput and delay in trace driven simulations on the other. The main difference is that graphs are most appropriate for modeling static properties of a network, while trace driven simulation is more appropriate in modeling the dynamics of a network.

1.1 Affecting Connectivity in Graphs

The first half of this thesis explores connectivity in static graphs where we focus on the communication capability of nodes in a given communication network. Specifically, we seek to assess how much of the network is still able to communicate after removing a given number of nodes or links. It is convenient to think of such a network as a graph $G(V, E)$ where each vertex in the set V corresponds to a node in the network and each edge in the set E corresponds to a link. By $\mathcal{V}(H)$ and $\mathcal{E}(H)$ we denote the vertices and edges associated with graph H. The graphs $G \ominus S$ and $G \oplus S$ denote the graph G with elements of $S \in \mathcal{S}$ added or removed respectively, where $\mathcal{S}$ is the power set of $V \cup E$. In other words, set $S \in \mathcal{S}$ may be any combination of edges, vertices, or both. In terms of element removal, the vertices and edges in S need not be consistent. That is to say, if vertex v is in S, then the edges adjacent to v are assumed to also be removed and we do not require that those edges be explicitly included in S.

1.1.1 Standard Graphs

We first tackle the problem of measuring connectivity in a graph subject to edge removal. This model treats all vertices equally and implicitly defines well-connected graphs as those with few and large connected components. Its concern is with the *existence* of a connection between pairs of vertices rather than the *quality* of that connection. Our goal is to solve the following problem.

PROBLEM 1: $MinPairs(G, \epsilon)$

- **Given:** graph $G = (V, E)$ and $\epsilon \in \mathbb{R}$.
- **Find:** minimum $Pairs(G \ominus R)$, over all R where $R \subseteq E$ and $|R| = \epsilon$.

Our approach to solving Problem *MinPairs* is to examine the spectrum of a graph's Laplacian matrix, which allows us to develop a lower bound on the solution to *MinPairs*. We compare this bound to actual cuts made by the METIS [96] graph partitioning library and find that the two bounds are close for relatively small graphs. We test this technique on two different families of random graphs: the Erdos Renyi (ER) and Barabasi-Albert (BA) graphs. Each family spans multiple graph sizes as well as quantities of edges. To the best of our knowledge, this is the first systematic characterization of the worst-case vulnerability of synthetic graphs subject to edge removal. Our results show that ER and BA graphs have roughly similar robustness properties when the graphs are small and comparably sized. Results are seen to weaken significantly when the size of the graph, the total number of edges, or number of edges removed increases. We also apply our bounds to the Power Grid of the Philippines. Our bounds are somewhat weak but informative nevertheless. The

results show that while the grid can be disconnected relatively easily, it is difficult to create a significant number of small components. We show that even after any 25 transmission lines are removed, each station is connected to a third of all other stations on average. We consider this a positive result because among the 404 total stations in the grid, there are 44 generators. Therefore, all stations have power so long as they are connected to at least one generator. We hypothesize that the typical station will thus be connected to at least one generator as it seems likely that at least one generator will exists in a large component. Unfortunately, the standard graph model is not capable of providing any further insight into this problem. This limitation and others motivates our study of client-server graphs.

Finally, we also investigate how our lower bound will change after graph augmentation. Specifically, we show how our spectral lower bound can be modified to anticipate how it will change subject to *any* augmentation of a fixed number of edges.

1.1.2 Client-Server Graphs

Our analysis on standard graphs is lacking because it cannot capture the hierarchy between nodes that is inherent to most networks. One way to enrich the basic graph is to differentiate between *client* and *server* vertices. Networks fundamentally facilitate the dispersion of resources. For a given resource, there are server and client nodes. Servers host a resource, while clients can consume or propagate the resource. In the presence of many different resources, each node may act as either client or server depending on the context. We focus on *client-server* graphs, or those graphs that exhibit a clear delineation between client and server vertices. We further assume that all servers are hosting the same resource.

The power grid provides an example. Generator nodes serve electricity and substation nodes receive and distribute it. If any substation is disconnected from *all* generators, then it and all of its customers will lose power. Wireless mesh networks can also be modeled as client-server graphs. In such a network, a small number of *gateways* (servers) are directly connected to the Internet, while a large number of *relays* (clients) connect only to gateways or other relays. Internet connectivity is established for any given relay only when there exists some path to a gateway. There are many other examples of client-server graphs that affect the lives of millions every day: telecommunications networks, Internet autonomous systems, and interstate highways are just a few.

A client-server graph is formulated as a graph $G = (V, E)$ where the vertex set V is further decomposed into two sets $V = S \cup C$ denoting server and client vertices. By determining how many clients can be disconnected from all servers we can characterize the ability of a network to disperse resources after vertex or edge removal. Specifically, our objective is to count the minimum number of clients that remain connected to at least one server after the removal of some fixed number of edges or clients. In this context, we define a *service path* as any path in G that begins at a client vertex and ends at a server vertex. Client vertex v is said to have a service path when there exists a service path beginning at v. Vertex v is then said to be *serviceable*. The following is our formulation of the vulnerability problem in the context of client-server graphs.

PROBLEM 2: $MinConn(G, \nu, \epsilon)$

- **Given:** client-server graph $G = (S \cup C, E)$ and $\nu, \epsilon \in \mathbb{R}$.

- **Find:** minimum $Conn(G \ominus (A \cup B))$, over all A, B, where $A \subseteq C, |A| = \nu$, and $B \subseteq E, |B| = \epsilon$.

In Chapter 4, we solve Problem $MinConn(G, \nu, \epsilon)$ in three parts. We first cast it as a semidefinite program, which returns a fractional solution analogous to those provided by linear programming. Second, we introduce a branch-and-cut algorithm that strengthens the result of the semidefnite program. Finally, we describe a procedure for rounding the fractional solution to a concrete set of edges and vertices whose removal renders a nearly optimal objective value.

We applied our solution to both a residential Wireless Mesh Network and the Power Grid of the Philippines. The results are two-fold: a lower bound on the number of clients connected to a server after vertex or edge removal, and a corresponding upper bound that additionally supplies the edge and vertex removal set. The results are particularly strong for the Wireless Mesh Network in that the upper and lower bounds remain quite close for intermediate values of nodes and links removed. We find interesting *watershed* moments during link removal where connectivity suddenly drops when one additional link is removed. Because there are two such moments, we hypothesize that these are moments when each of the gateways is completely disconnected from the rest of the network. Roughly half the clients are disconnected from all gateways after removing just 12 links ($< 5\%$). Direct client removal is less dramatic but more damaging. We show that removing 12 clients could potentially disconnect more than 70% of the remaining clients.

For the Philippine Power Grid, agreement between upper and lower bounds is not quite as strong. Nevertheless, we are able to draw significant conclusions for up to

50 links or clients removed. For example, removing just over 6% of the transmission lines will disconnect at least 20% but not more than 50% of the substations from all generators. On the other hand, we also find that almost 50% of the substations can be disconnected from all generators after strategically removing a certain 20 (approximately 5%) other substations.

1.1.3 Large Graph Solution

The basic branch-and-cut solution to Problem $MinConn(G, \nu, \epsilon)$ shows weakening performance as the size of the graph increases, but we would like to evaluate the connectivity properties of still larger graphs. To address this issue, Chapter 5 develops a more sophisticated algorithm that offers a better solution to Problem $MinConn(G, \nu, \epsilon)$ for large graphs. Our algorithm specifically solves Problem $MinConn(G, \nu, 0)$, that is, it does not address edge removal. This approach proceeds in two stages. First, an aggregation algorithm consolidates groups of vertices of the given graph into a single pair of numbers: weight and value. The second stage then models these weight-value paris as a version of the Knapsack Problem, whose solution is subsequently used to establish both lower and upper bounds on client serviceability.

We applied this approach to the Autonomous Systems (AS) level graph of the Internet and the highway systems of Iowa and Michigan. In the AS graph, two types of AS nodes are frequently distinguished: Tier 1 and Tier 2. Tier 1 nodes essentially form the backbone of the Internet while Tier 2 nodes act as downstream conduits. Therefore, it seems appropriate to model the AS graph as a client server graph with Tier 1 nodes being servers and Tier 2 nodes being clients. Our approach delivers tight upper and lower bounds on client connectivity to servers even for large

numbers of clients removed. Actually, its weakest performance comes with small numbers of clients removed, which arises from error introduced during the aggregation phase. Nevertheless, we are able to establish relatively tight bounds on Tier 2 node connectivity in the AS graph. Specifically, we show that the AS graph is quite robust to the removal of Tier 2 nodes. Even when 900 ($> 16\%$) Tier 2 nodes are removed, better than half the Internet remains connected.

We also used the large graph solution to analyze the number of people with highway access to airports in the states of Iowa and Michigan. For this application we solve a weighted version of *MinConn* called *WeightedMinConn*(G, ν, ϵ) where each $v \in C$ is assigned a weight $w(v)$. The objective is to minimize the weighted sum of the serviceable clients in $G \ominus (A \cup B)$, where $A \subseteq C$ and $B \subseteq E$, with $\sum_{v \in A} w(v) \leq \nu$ and $|B| = \epsilon$. The weighted variant of the problem is useful in our road network analysis where each vertex represents a geographical area and is weighted by its population density. We show that when 200 roadblocks are erected state-wide it is possible to disconnect between 500K and 1.7M people (15–50% total population) from every airport in Iowa. In Michigan we find that only 2.8M to 4.3M people (28–43% total population) will maintain airport connectivity after erecting 300 road blocks.

1.2 Service Quality in Dynamic Networks

Mobile networks are quickly becoming ubiquitous: smartphones, netbooks and in-dash vehicular computers have proliferated in recent years. For most applications running on these devices, Internet connectivity key. However, if mobile nodes are to receive uninterrupted wireless coverage over a large geographic area, they require the

support of a large infrastructure of cell towers or access points. Cellular data coverage is currently expensive for carriers to deploy and has a recurring cost of about USD$50 per device per month per user. WiMAX towers are tens of thousands of dollars and require allocation of a band, and therefore they similarly cannot be easily deployed by common users wishing to support mobile networking. More accessible are WiFi APs, but such nodes rely on an underlying high speed backbone, such as that provided by an institution or carrier. Studies have shown that WiFi networks typically provide intermittent access [65], even when deployed in an unbroken mesh [20]. An attractive solution is to develop protocols for *disruption tolerant networking* (DTN) whereby mobile devices share the burden of delivering packets in a network using a store, carry and forward paradigm.

Recall the definition of DTN $\mathcal{D} = (N, K, C)$. DTNs are much more descriptive than standard graphs, but they're also much more difficult to analyze because of this additional complexity. In Chapter 6, we begin by assessing how the metric $Thru(\mathcal{D})$, or maximum throughput, degrades as nodes in N are removed.

PROBLEM 3: $MinThru(\mathcal{D}, M, \nu)$

- **Given:** DTN $\mathcal{D}(N, K, C)$, demand matrix M, and $\nu \in \mathbb{R}$
- **Find:** minimum $Thru(\mathcal{D} \ominus R)$, over all R where $R \subseteq N$ and $|R \cap N| = \nu$.

Our approach is to first show experimentally that a greedy attack, removing the nodes implicated in the most contact events, is nearly optimal in a simplified, graph-based, version of Problem *MinThru*. We then apply this heuristic to traces taken from two real world DTNs: DieselNet [107] and the Haggle Project [88]. The

results show that the greedy attack is more damaging than a random attack, but that DTNs are fairly robust in either case. For both data sets, the greedy attack tends to lower throughput linearly in the number of nodes attacked.

1.2.1 DTN Augmentation

Disruption tolerant networks are uniquely robust to attack because nodes are mobile, and they are therefore constantly establishing new connections. The same reasoning suggests that adding some collection of stationary nodes, or relays, can bolster connectivity in a DTN and further protect the network from attack. In Chapter 7 we examine the following problem.

PROBLEM 4: $Placement(\mathcal{D}, L, M, m, P^*)$

- **Given:** DTN $\mathcal{D} = (N, K, C)$, a set of potential relay locations L, a demand matrix M, a number of relays $m \in \mathbb{N}$, and a minimum throughput P^*.
- **Find:** minimum $Delay(\mathcal{D} \oplus R, M, P^*)$, over all supplementary sets of stationary nodes $R \subseteq L$, $|R| \leq m$.

Relay nodes have the potential to augment DTNs by providing cheap additional infrastructure. Prior to our work, very little was known about the best relay placement strategy. Our major contributions are in devising a scheme to validate potential placement strategies and actually demonstrating the efficacy of several strategies. In particular, we first alleviate much complexity by defining a decoupled placement problem, which is accomplished by assuming a fixed routing protocol. The optimal solution to this problem is labeled *D-optimal*. This allows us to show that choosing

those locations that are most frequently visited is a highly effective strategy in DTNs with node hubs. For example, assuming RAPID routing [19] and choosing the 50 most frequented locations over 30 days of traces yields a placement that, when tested on the remaining 28 days of traces, delivers 40% of desired flow with D-optimal delay 40% of the time.

1.3 Data Sets

We apply our techniques to a variety of real world data in order to gain a deeper understanding of the underlying networks and also the quality of our algorithms themselves. Just as with our algorithms, the sets can be broken into two groups: static and dynamic. Among the static sets are the Net Equality Wireless Mesh network, the Power Grid of Philippines, the Internet Autonomous Systems graph, and the highway networks of Iowa and Michigan. While these graphs are not actually static — their topology does change over time — they are not considered intermittently connected as is the case for the second group: DTNs. We choose two very different types of DTNs to try to show the similarities that are inherent in all networks of this nature.

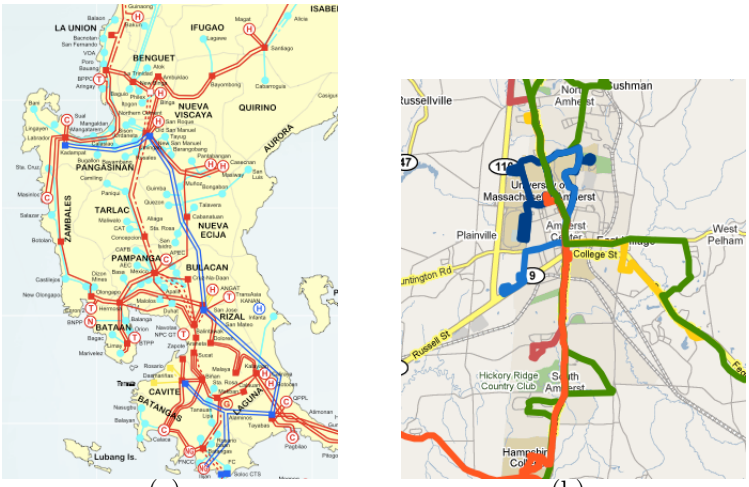
1.3.1 Net Equality

The Net Equality Project [5] works to provide Internet access to low-income communities where residents live in dwellings within close proximity, such as apartment complexes. They deploy a small number of gateway nodes with full access to the Internet and relatively inexpensive Meraki Mini access points (relays) around the community. A resident has access to the Internet wherever he or she is in range of

either a gateway or a relay that is ultimately connected to a gateway. We analyzed a snapshot of the Hacienda CDC project taken in 2007. At the time, 69 relays and 2 gateways were connected by 279 links serving approximately 1,200 residents.

1.3.2 Power Grid of the Philippines

Our Power Grid map of the Philippines was drawn in 2006 [6]. A portion is included in Figure 1.1(a). We extracted the underlying graph by observation from a large printed copy of the map. First, generators and substations were identified and labeled as servers and clients respectively. Then an edge was added to the graph for each connected pair of vertices. The result was a grid that comprised 44 power generators and 360 substations connected by 796 transmission lines.



(a) (b)

Figure 1.1. (a) A portion of the map of the power grid in the Philippines (b) DieselNet bus routes

1.3.3 Internet Autonomous Systems Graph

We analyzed a snapshot of the Internet at the Autonomous Systems (AS) level. ASes in the Internet are administrative domains managed by a business entity such as AT&T or Sprint. Two types of AS nodes are distinguished: Tier 1 and Tier 2. A Tier 1 AS is an autonomous system that connects to the rest of the Internet solely by means of *peer* relationships with other ASes. A peer relationship is essentially a business agreement between two ASes to exchange data *quid pro quo*. Accordingly, Tier 1 ASes have been said to form the *backbone* of the Internet [101]. All other ASes rely on some combination of peer and monetized relationships to provide full connectivity to the Internet. Such ASes are frequently labeled Tier 2. Being private agreements, peering relationships are not always advertised and are thus difficult to infer. Gao [73] has pioneered a technique for making such inferences, which form the basis of classification algorithm developed by CAIDA [1].

1.3.4 Highway Networks of Iowa and Michigan

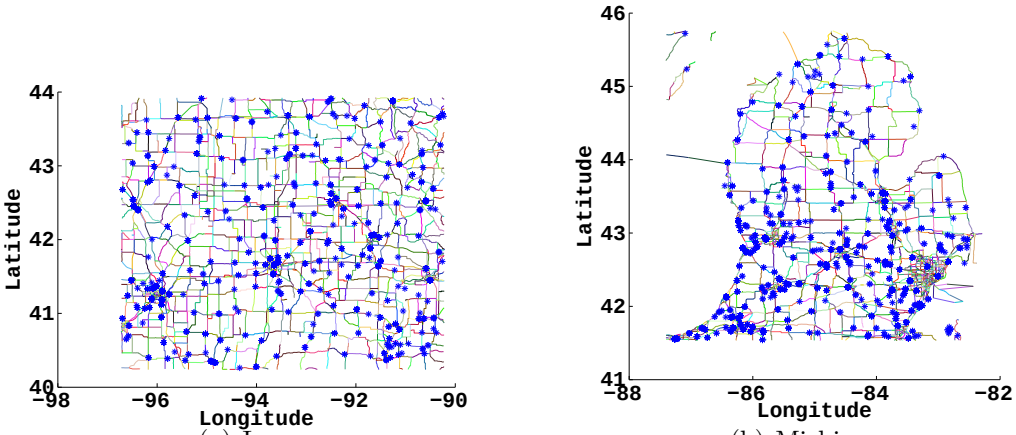
We also addressed airport connectivity in the states of Iowa and Michigan. To do so, we began by obtaining U.S. highway information from the Oak Ridge National Laboratory [3]. This was found in a single large file composed of more than 200K different routes in North America. Each route is delineated by a sequence of coordinates. We call these coordinates *waypoints*. From the Socioeconomic Data and Applications Center (SEDAC) [2], we obtained population density estimates from the 2000 census that are accurate to within several hundred meters.

For each state, we extracted the routes and corresponding waypoints that fall within the bounding box containing that state. A graph representing the highway

connectivity of each state was then generated by creating a client vertex for each waypoint and an edge for any two adjacent waypoints in a given route. Because one route will often end at the beginning of another, we consolidated waypoints that are within 100 meters of each other into a single vertex. Our next step was to create a server vertex for each airport, which we linked to all waypoints within a (roughly) six mile radius. The coordinates for airports in these states are found at the data warehouse site socrata.com [4]. The final step in graph construction was to assign a value to each waypoint corresponding to the estimated population density near that point using the SEDAC data. Since people might arrive at an initial waypoint near their place of residence by means other than automobile, we averaged the measured population density between all waypoints within approximately 15 miles of each other and assigned each waypoint a value corresponding to this population density. This means that the sum of all waypoint values in a given state is approximately equal to the population of that state. Figure 1.2 shows all the routes we use for each state as well as the available airports.

1.3.5 DieselNet DTN

The DieselNet [107] DTN is a network of buses serving a large area of western Massachusetts. The area of operation covers approximately 150 square miles. Traces came from a three-month period ranging from the beginning of February 2007 to the beginning of May 2007. All weekends and holidays were dropped to allow a homogeneous road schedule that left a total of 58 days. On each day, we eliminated buses that make no contact with other buses. The result was an average of more than 15 buses on the road each day. Figure 1.2 shows a map of the area of operation that



(a) Iowa (b) Michigan

Figure 1.2. Airports are shown as blue dots over highways in (a) Iowa and (b) Michigan. Colors vary between different routes. Waypoints are not shown.

is overlaid with lines depicting the routes upon which the DTN is active. Each bus is outfitted with a small computer comprising an 802.11 wireless transmitter/receiver and hard drive. During the course of the day, buses record their location at 10 second intervals as well as any contact events with other buses. During contact events, buses attempt to send as much data as possible via wireless transmission in duplex mode. The average transmission size is 1.7MB with a standard deviation of 1MB.

1.3.6 Haggle DTN

The Haggle traces were drawn from a human mobility experiment at Infocom 2005, using 41 volunteers carrying iMotes that connected to one another, as well as connecting to Bluetooth-capable devices in the environment [88]. To allow better comparison of Haggle data to DieselNet data, we removed connection events from

the Haggle data that lasted less than one second or involved the singular appearance of a node since meaningful data transfer is likely to require setup time and nodes incapable of routing data may be ignored. After these transformations, we were left with events involving 41 Class 1 devices. Finally, we note that Haggle data, unlike DieselNet data, only reports contact times and durations without sending any data. To compare the bandwidth distribution among nodes between the two networks, we assigned a constant transmission rate to Haggle data equal to median bandwidth observed in DieselNet.

1.4 Notation and Terminology

In our theoretical development, we assume the vertices of every graph are ordered and identified by the natural numbers. That is to say, we assume that for any graph $G = (V, E)$, each vertex is identified by an arbitrary and unique integer value between 1 and $|V|$. This practice is useful when, for example, indexing vertices in summations.

When dealing with matrix variables, we use Matlab notation. Specifically, for matrix variable X, $X(i, j)$ is the entry in row i and column j of X. The symbol ":" acts as a wildcard character so that $X(i, :)$ is the row vector equal to row i of X, and $X(:, j)$ is the column vector equal column j of X. We also make use of the diag operator as follows: for matrix X, $\text{diag}(X)$ is a vector containing the elements found along the diagonal of X. In contrast, for vector x, $\text{diag}(x)$ is the matrix of zeros with x along its diagonal. We also make use of the $\text{ones}(m, n)$ and $zeros(m, n)$ functions, which return an $m \times n$ matrix of all ones and zeros respectively.

A *formulation* is a mathematically rigorous problem statement that specifies an objective function and complete set of constraints, which define a problem that may or may not be solvable. A *program* is a formulation that is phrased as, and solvable by, a mathematical program.

1.5 Contributions

The thrust of the thesis is toward deriving and applying techniques for understanding vulnerability in real-world networks. In some cases, as with graph vulnerability, prior work does exist, but it does not provide a deep enough framing or analysis of the underlying problem. In other cases, as with the study of DTNs, there has simply never been an attempt to quantify the effect of addition or removal of nodes in such a network. Below we summarize our contributions in each of these areas.

- *Spectral Technique for Graph Vulnerability*: here we look at the graph vulnerability problem using a classical approach, but applying a novel measure, pair-wise connectivity.
  - offer a purely algebraic lower bound on the number of connected pairs of vertices in a graph after link removal that cannot be accurately captured by existing techniques.
  - project how this bound can improve after link augmentation
- *Vulnerability in Client-server Graphs*: this is a departure from the classical framework that allows us to model the distribution of services in a network.
  - provide a new framework for assessing vulnerability in graphs.

  - first to offer upper and lower bounds on vulnerability for *any* graph with simultaneous edge and vertex removal
  - address problems with large graphs by introducing a topological decomposition appropriate for many sparse networks
  - apply this technique in diverse settings revealing surprising properties of real-world networks that are essential to our modern society

- *Robustness of DTNs to Node Attack*: this work identified the intermittent connectivity of DTNs as the key to their resilience to node attack.
  - first to demonstrate that DTNs are relatively resilient to node attack as validated by experiments on two different real-world DTNs

- *DTN Augmentation with Relays*: with a small amount of prior work indicating the utility of adding relays to a DTN, we explored the optimal placement within a large geographical area.
  - the first to study optimal placement of relays in DTNs
  - offer bounds on optimal tradeoff between packet delivery rate and delay
  - validate the use of greedy heuristic for relay placement

CHAPTER 2

BACKGROUND

We begin by motivating each of the problems studied in this thesis. We subsequently elucidate areas of the literature that relate to each problem and provide a comparison to our work wherever it is appropriate.

2.1 Motivation

2.1.1 Node Attacks

Problems *MinConn* and *MinThru* evaluate node removal in networks. It's important to study such attacks because they are common in a wide variety of networks. Wireless networks have become increasingly popular as network devices become smaller, cheaper, and more reliable [7]. Also increasing in popularity are DTNs [68], which are wireless networks comprised of mobile nodes. Chakrabarti and Mishra [44], Yang et al. [146], and Perrig et al. [121] discus various security issues unique to wireless networks; they note specifically a susceptibility to denial-of-service attacks, spoofing attacks, and physical tampering. Arbaugh et al. [14] point out the inadequacy of existing wireless security measures such as WEP. Falsified updates can also bring down resources in unauthenticated routing protocols supporting intranets [133], mobile ad

hoc networks [86, 129], and sensor networks [95]. In wireless networks, disruption by jamming the physical layer is as easy as it is unpreventable [103, 145].

Power grids are crucial to the networking infrastructure as well as modern society at large. A major blackout in North America in 2003 ignited interest in the study of worst case fault tolerance in power grids. The current sociopolitical environment has also instigated fears of targeted attack. Amin [12] cites centralized ownership of regions of power grids as a reason for topological vulnerability. Cyber attacks on grid communications are also cited as being a serious threat to power control centers [13, 141]. The industry consensus is a need to identify attack and failure scenarios before they happen [12, 136].

The highway system in North America is another crucial piece of infrastructure. We rely on highways for connectivity to jobs, markets, hospitals, and airports. A report released by the Department of Homeland Security in 2006 identified eight unrelated suspicious events that seemed to indicate terrorist attention to major pieces of domestic highway infrastructure [84]. There is significant concern that such locations could be the target of traditional improvised explosive devices (IEDs) or vehicle borne IEDs. As part of a major coordinated effort towards securing our national highway system, the United States Government Accountability Office recommends the development of cost effective mechanisms for monitoring critical pieces of infrastructure [31]. Specifically, they call for individual vulnerability assessments on the nationally critical Tier 2 structures list.

2.1.2 Link Attacks

The threat of link removal by attack or failure is significant. Though nodes tend to be well secured, there are several ways to target links for physical destruction. In wired networks, fiber optic cables and other media are conspicuously susceptible to intentional or inadvertent destruction. It is impractical to provide the same level of physical security that is given to Internet routers to the tens of thousands of miles of Internet fiber in the ground. In fact, fibers are cut easily and commonly [123]. In 2004, earth excavators (e.g., backhoes) caused 7,795 reported incidents of accidental damage to Telecom and CATV facilities in the U.S. [15]. Moreover, the location of fiber is largely a matter of public record in order to avoid such accidents. For example, Gorman el al. [76] mapped the location of major fiber lines by researching publicly available records.

In addition to physical security concerns, both nodes and links in many scenarios are subject to protocol exploits. Part of our evaluation focuses on traces of the BGP routes available between Internet Autonomous Systems. The currently deployed BGP protocol is not secure; the vulnerabilities have been documented by numerous researchers for years [29, 30, 97, 116, 132]. BGP vulnerabilities include falsification of route updates, which can remove available links. BGP attacks are commonly performed: the addition of routes is a favorite tool of spammers [125] and deletion is not significantly more difficult. What is challenging is the deployment of a secure BGP protocol, and it is the focus of the latest research related to this problem [41,45,87,139]. Falsified updates can also bring down resources in unauthenticated routing protocols supporting intranets [133], mobile ad hoc networks [86, 129], and sensor networks [95].

In wireless networks, disruption by jamming the physical layer is as easy as it is unpreventable [103, 145].

2.1.3 Adding Network Relays

While still a new topic, a growing set of works has examined the performance gains that are possible in a sparse mobile network that results from adding infrastructure. Banerjee et al. [21] design and deploy energy-efficieny solar-powered relays and demonstrate the network improvement from using a single relay in a DTN. While that paper proposed a longer range, hailing radio and shorter range data radio, Jun et al. [93] examine the use of the opposite case. Nain et al. present the asymptotic routing performance for large numbers of relays [89]. Banerjee et al. [22] show that incorporating additional Internet access points (that can be placed anywhere in a geographic area) can be twice as valuable as adding the same number of mesh points (that can only be placed contiguously); and APs are five times as valuable as adding the same number of relays. However, relays do not require an underlying wired or wireless infrastructure, and are therefore cheaper and easier to deploy. In the same paper, Banerjee et al. show the performance of placing boxes according to locations most visited by mobile nodes, but those experiments did not validate the quality of such an approach other than against other infrastructure.

2.2 Graph Vulnerability

Problems *MinPairs* and *MinConn* consider edge and vertex removal in graphs and report the minimum number of pairs of connected vertices and serviceable clients

respectively. They share common ground in a vast subfield termed graph vulnerability (see Grubesic et al. for a recent review [78]). In general, graph vulnerability problems consider how a certain graph metric or invariant changes with the removal of edges or vertices chosen according to some strategy. Problem *MinConn* is unique both in its phrasing and in its solution. Problems very similar to *MinPairs* have recently gained attention from other authors. Dinh et al. [60] introduce the β-edge and β-vertex disruptor problems that seek a subset of edges or vertices that disconnect a fraction of at least (1-β) strongly connected pairs of vertices. The undirected and unweighted version of the edge and vertex β-disruptor problems are very similar to the duals of problems $MinPairs(G, e)$ and $MinPairs(G, v)$ respectively. Unfortunately, their approach is not appropriate for worst-case analysis. They use recursive bisection (or near bisection) to find the best disruptor. For many graphs, this approach is suitable, but it cannot be used to develop tight lower bounds for our problems. This is easily seen by considering a pathological example where a graph, $G = (V, E)$, is composed of three equal sized cliques each connected to the others by a single edge. In this case, it's possible to lower the overall pair-wise connectivity in the graph by a factor of 3 by removing just 3 edges, but recursive bisection will never find these edges. Instead, the best cut of comparable balance will be of order $(n/3)^2$ edges. Arulselvan et al. [17] focus on minimizing pair-wise connectivity after node deletion. Their approach was similar to the approach we took to solve *MinPairs* in that they began by bounding the objective from below and sought to solve the resulting relaxation using mathematical programming. Unfortunately their solution was computationally intense (evaluating graphs with about 100 nodes took over an hour) and does not yield a lower bound.

2.2.1 Similar Edge and Vertex Deletion Problems

Early theoretical work from Chung [50, 51] offered theoretical bounds on the diameter of a graph after edge removal. Diameter is a useful metric when the graph in question never becomes disconnected. Dekker and Colbert [57] pointed out that symmetry is crucial to robustness in the face of adversarial attack. More recently, interest in preferential attachment models in the context of network vulnerability has yielded theoretical results specific to the BA model [42, 53] and the mathematically distinct LCD model [35, 36]. The consensus is that, relative to the ER graph, graphs that are grown according to a preferential attachment rule are more robust to random failure, but more vulnerable to targeted attacks. Recently, Kleinberg [99, 100] developed a unique framework for *detecting* failure in a network. His strategy was to calculate the size of cut (link or node) that would separate a witness pair of connected nodes.

A traditional approach to studying the vulnerability of a complex network is to use graph partitioning techniques such as finding graph separators of small width [128]. An α-separator of a graph $G = (V, E)$ is a collection of either edges or vertices whose removal separates a graph into two disconnected subgraphs, each having size at most $\lceil (1 - \alpha)|V| \rceil$. The number of edges or vertices in the cut is called the separation *width*. Finding either an α-edge-separator or α-vertex-separator of minimum width is NP-hard. Bui and Jones have shown it is NP-hard even to find a good approximation to the α-separator problem [117]. The objective of graph separation is to completely partition the graph into appropriately-sized pieces by removing vertices and edges. In

contrast, our goal is not to partition the graph completely, but to study the manner in which service availability degrades as a function of the number of failed components.

Even though standard graph partitioning does not appear to be directly applicable, some of the techniques developed in that context have been key to our approach. Specifically, Wolkowicz and Zhao [144] developed techniques for using semidefinite programming for graph partitioning. We utilize some of those ideas in our algorithm, specifically the notion of a lift matrix to represent our block decomposition of the client-server graph.

Problem $MinPairs(G, 0, e)$ can be weakly bounded by the α-separators problem. Because the vertices in the separated half of the graph may or may not be connected we can derive the simple bound stated in Proposition 1.

PROPOSITION 1: For edge separators, if G is initially connected and α is the largest bipartition coefficient with separation width less than or equal to k in G, then

$$\alpha|V| + (1-\alpha)^2|V|^2 \leq MinPairs(G,k) \leq \alpha^2|V|^2 + (1-\alpha)^2|V|^2.$$

So the greatest α that renders a separation width of no greater than k provides $\alpha|V|$-approximation for *MinPairs*. This ratio is exact when the partition of size $\alpha|V|$ is a single connected component and no other vertices are connected.

Though graph theoretical techniques are effective for finding graph separators in some specific types of graphs [128], there are two major approaches to constructing separators in arbitrary graphs. Spectral techniques [10, 32, 46, 52, 54, 77] exploit numerical properties of the graphs adjacency matrix. Flow techniques [16, 98, 106, 131]

generalize the concept of min-cut, for example multicommodity min-cut, to optimize α.

2.2.2 Experimental Approaches

Synthetic graphs offer a means of testing algorithms as well as generalizing graph properties. In the latter case, one can assess the vulnerability of a static *real* network by comparing it to a similar synthetic graph, with known properties. The idea is that a measured similarity in structure suggests a similarity in vulnerability as well. This technique is fundamentally different from the techniques we use to assess vulnerability in static networks because our graphs are constructed to be isomorphic to the networks we study while synthetic graphs only seek approximate similarity. The literature is full of measures of vulnerability applied to particular models. For these reasons we review several network models chosen to reflect recent developments in the literature. See Bonato [37] for a survey. Faloutsos et al. [67] showed experimentally that the degree distribution of the nodes of the Internet obeys a power law. That is, the probability of a node having degree k is proportional to $k^{-\beta}$, for some β. Barabasi and Albert [24] proposed a new random graph (commonly called the BA graph) that models this phenomenon. A BA graph is *grown* by *preferentially* attaching new nodes to existing nodes with probability proportional to the existing node's degree. This graph differs significantly from the common ER random graph (i.e., $G_{n,p}$ [24]) of Erdos and Renyi [34] in both construction and degree distribution. The authors showed that a BA graph is scale free in that the degree distribution will remain linear with a constant slope on a log-log scale as the size of the graph increases. The

implication of Barabasi and Albert's work is that the similarity of degree distributions between the Internet and BA graph indicates similar topological properties [24].

Lai et al. [102] evaluated link removal in BA and ER graphs subject to removal of short-range and long-range links. Neither attack is very effective: the average shortest path length of 1.0 increased by no more than 0.1 links when as many as 4% of the links were removed. Guillaume et al. [80] evaluated the vulnerability of scale free graphs of a fixed degree distribution as well as ER graphs subject to a randomized link attack in terms of the largest connected component size (LCCS). The strategy involved randomly removing links between nodes of degree at least two. In this context, they found that the scale free graph had approximately the same robustness as the ER graph. Ding et al. [59] used spectral techniques to decompose the web graph, while Gkantsidis et al. [75] performed similar analysis on the Internet AS graph. Magoni and Zhou et al. [113, 148] independently analyzed the Internet AS graph and showed that by removing approximately 3% of the ASes (including Tier-1's and Tier-2's) the size of the largest connected component decreases by a factor of 3. Flaxman et al. [72] provides a framework for strengthening existing graphs by supplementing them with regular subgraphs. Cohen et al. [55] provided a modification to classical percolation theory in order to study the effects of removing vertices with highest degree, and demonstrated that even though scale-free graphs are robust to random failure they are highly sensitive to targeted attack.

The Internet AS level [61, 75, 118, 138, 148, 149] and router level [113] graphs have been found consistently to have low resilience, in terms of the LCCS, against the *highest degree first node attack* (HDN). Similar results have been established for

the Internet AS level graph in terms of graph diameter [8]. The BA graph displays somewhat higher resilience than the AS graph when subjected to the HDN attack in terms of the LCCS [72, 85, 149]. For the same attack and metric, Park et al. [118] quantified the BA graph's vulnerability, and Bollobas [36] and Holme [85] showed that the BA graph is more vulnerable than the ER graph. Lee and Kim [104] studied the affect of random node and path removal in both the AS and router level Internet. Path attacks had universally higher failure rates, where nodes were said to have failed when they had no neighbors. In contrast to the commercial Internet, Internet2 was demonstrated to be rather robust subject to the HDN attack in terms of the LCCS [38].

2.3 Other Analysis on Attacking DTNs

The field of disruption tolerant networking (DTN) is relatively new, and therefore, the study of attacks on DTNs has been limited. Jain et al. [90] study random failure in DTNs and offer a variety of techniques to compensate for such failures including message splitting and replication and use of erasure coding. To the best of our knowledge, this is the only work predating ours to systematically examine failure in DTNs. Our work is distinct in that it considers intentional attack as opposed as random failure.

Subsequent work has focused on protocol exploits and equitable routing. Li et al. [108] examine how malicious nodes in a DTN can *trick* many routing protocols that exploit periodic node movements. They argue that by altering its movement, a malicious node can create a more central role for itself in the DTN. Their solution is

to form a web of trust among honest nodes in the DTN so as to eliminate the threat form malicious ones. Uddin et al. [137] introduce several denial-of-service attacks achievable in DTNs. The go on to introduce countermeasures for those attacks and evaluate the tradeoff that exists between security and performance. Zhu et al. [150], consider malicious nodes to be those members of the DTN that would prefer not to forward messages. They implement an incentive scheme to encourage participation among these reluctant entities. Solis et al. [134] look at nodes that behave in opposite fashion by hogging resources. They propose a stratification of DTN users in order to assign messages sent by *hogs* a lower priority.

2.4 Similar Efforts in DTN Augmentation

Two papers are most relevant to our work. Zhao et al. [147], who specifically address the placement problem, introduce a heuristic that has the goal of increasing throughput; however, that analysis did not address delivery delay, and it lacked validation and evaluation based on a real network scenario. We could not extend this technique for our analysis as it became computationally intractable as packet delay constraints were added to the model. Lochert et al. [112] examine the use of stationary relays in VANETs. They propose a genetic algorithm for placement and evaluate this algorithm via simulated mobility in a large city using ns-2. A bit vector is formed with one entry for each potential relay location, after which they conduct an entire round of simulations each time the vector is mutated. Their approach is very computationally intense and seeks only to optimize delay, never the less, we compare their solution to ours in Chapter 7. We find very little difference between

their solution and a frequency-based placement in terms of both packet delivery rate and delay.

Another set of works add mobile ferries to the network. For example, Jun et al. [94] examine how nodes can optimize routing by leveraging the known schedule of a ferry. And Burns et al. [40] design a system where ferries move autonomously to meet demand in a DTN. A series of DTN routing protocols have been proposed based on historical mobility patterns [18, 56, 105, 135]. Jain et al. [91] show that routing in a DTN is not as simple as choosing among available paths, and show how to encode DTN properties into a linear program. We specifically use the RAPID protocol [19] in this paper to evaluate relays. RAPID aims to directly affect QoS metrics by tailoring routing accordingly. Routing decisions at local nodes are based on the estimated probability that a given node can satisfy the metric of interest.

A set of other work has examined coding [49, 111]. For example, Wang et al. [140] compared erasure-coding routing in DTNs to packet flooding, simple replication, and a history-based method. They observed that overall delay was lowest for the history-based strategy, while the coding strategy ensured the best worst-case performance. And Widmer and Le Boudec [142] show that coding with replication provides a higher delivery rate than replication alone, but at the cost of higher delay.

2.5 Foundations of Analysis

2.5.1 The Laplacian

From a numerical perspective it is difficult to avoid the graph *Laplacian*, which is the difference of the graph's adjacency matrix, A, and a diagonal matrix comprising

the row sums of A. The Laplacian differentiates edges *internal* to and *external* to any subset of vertices. In seminal work, Fiedler [70, 71] related the ratio of the size of a graph separator to the size of the largest partitioned *block* (i.e., subgraph). This ratio is often called the *graph expansion*. Donath and Hoffman [62] developed a generalization of graph expansion to an arbitrary number of separators, which was subsequently generalized by Barnes and Hoffman [28]. Pothen et al. [122] elaborated on the connection to graph separators. Most work since has found an audience in the VLSI and parallel computing communities. Spectral techniques [64,81] are most popular, but are not always very accurate, as was noted by Guattery and Miller [79]. Alternatively, Wolkowitcz and Zhao [144] use the Laplacian as a constraint in a Semidefinite program, a technique of great important to our analysis. See Mohar [114, 115], Alon [10, 11], or Seary and Richards [130] for a review of Laplacian techniques.

2.5.2 Separation Algorithms

Several available programming libraries [83, 96, 119, 124] implement the most recent algorithms for graph partitioning due to link removal. In particular, we use the Metis library [96] to provide upper bounds for our analysis. Each package employs a variant of *multilevel graph partitioning* [96], which operates in three nominal stages. During the first stage, the nodes of the graph are merged to create a relatively small representational graph. The second stage uses a heuristic to create a nearly optimal partition. In the third stage, a heuristic is used to balance the partitions, which are subsequently expanded. The process is repeated until a separated graph of full size remains with a small separator of specified balance.

CHAPTER 3

GRAPH CONNECTIVITY AFTER EDGE REMOVAL

Many static networks are considered functional only when the vast majority of pairs of nodes in those networks are connected by one or more paths. This is true of the power grid, Internet, and wired telephone networks. After link deletion, paths are disrupted and pair-wise connectivity among nodes breaks down. We model such a network by a graph $G = (V, E)$ with vertices in V corresponding to nodes and edges in E corresponding to links. Problem $\mathit{MinPairs}(G, \epsilon)$ gives the number of connected pairs of vertices after the removal of at most ϵ edges. Such a measurement gives a good idea of the communication capability of the underlying network after attack. Unfortunately, we show in Chapter 8 that *MinPairs* is NP-hard, so the focus of this chapter is in developing a lower bound on the solution to the problem. We begin by introducing a framework for representing *MinPairs*, which will subsequently be honed into solution for this problem.

For any $G = (V, E)$, a *block* B is a subset of V such that $G(B)$ is disconnected from $G(V \setminus B)$. Suppose that there are m vertices in an initially connected graph G and we wish to create a partition into p disconnected blocks by removing at most ϵ edges. This is accomplished by placing each vertex in V into one of the p blocks, $B_1, \ldots, B_p$, so that no more than ϵ edges pass between those blocks (see Figure 3.1).

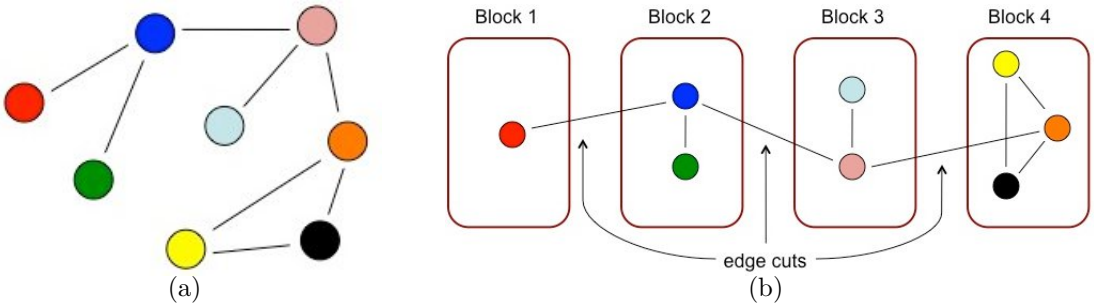


Figure 3.1. (a) Graph representation of a simple network. (b) Decomposition into four blocks.

This set of edges passing between blocks must be removed in order to realize the desired partition. We label those edges the *edge cut*. For the resulting partition, assume that block i has size m_i with $m_1 \leq \ldots \leq m_p$ and $\sum_{j=1}^{p} m_j \leq m$. Let A be the adjacency matrix of G, D be the matrix with row sums of A along the diagonal, and $L = D - A$. Also, let X be an $m \times p$ *block indicator matrix* where,

$$X(i,j) = \begin{cases} 1, & \text{node } i \in B_j \\ 0, & \text{otherwise} \end{cases}. \tag{3.1}$$

It follows immediately that

$$\sum_j X(:,j) = m_j \text{ and } \sum_i X(i,:) = 1,$$

where the operator ":" is defined as in Section 1.4. For a given partition X let the the edge cut set be given by $\mathcal{C}_E$. Problem $MinPairs(G, \epsilon)$ can be reformulated as follows.

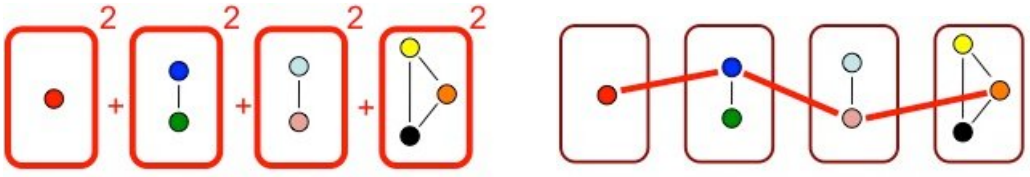
FORMULATION 1: *MinPairsForm*(G, ϵ)

- **Given:** graph $G = (V, E)$ and $\epsilon \in \mathbb{R}$.
- **Find:** minimum $\sum_{i=1,\ldots,p} \| X(:,i) \|^2$ such that,

(1) $\sum_{i=1}^{m} \sum_{j=1}^{p} X(i,j) \leq m.$ (Vertex Count)

(2) $|\mathcal{C}_E| \leq \epsilon.$ (Edge Cut Size)

(3) $X(i,j) \in \{0,1\}$ (0–1 property)



(a) Minimize (b) Subject To

Figure 3.2. *MinPairs* problem formulation.

Each column vector, $X(:,i)$, is in indicator vector for the vertices included in block B_i. The objective of Formulation *MinPairsForm* seeks to minimize the sum of squares of these vector lengths because we would like to minimize the sum of squares of the sizes of the blocks $B_1, \ldots, B_p$. This quantity is equivalent to the number of pairs of vertices communicating *within* all blocks but not between them. The optimization is subject to the second constraint, which specifies that the number of edges passing between blocks, $|\mathcal{C}_E|$, is no greater than ϵ. The first and third constraints ensure that the block indicator matrix X represents a valid partition. Figure 3.2 shows how the objective and second constraints relate to our original example.

3.1 Spectral Decomposition

Our approach to solving the problem will be to first relax the edge cut constraint, and formulate a quadratic programming problem whose solution bounds $MinPairsForm$ from below by means of spectral decomposition. We begin with a lower bound on $|\mathcal{C}_E|$ due to Donath and Hoffman [62], which we ultimately use to relax $MinPairsForm$. The matrix L is known as the graph Laplacian of G. A great deal is known about the eigenvalues $\lambda_1 \leq, \ldots, \leq \lambda_m$ and corresponding eigenvectors $u_1, \ldots, u_m$ of L. Since L is symmetric and positive semidefinite, its eigenvalues are real and non-negative. Its eigenvectors form a complete orthonormal basis so the spectral theorem dictates that $L = \sum_{i=1}^{m} \lambda_i u_i u_i^T$. Therefore, $|\mathcal{C}_E|$ can be rewritten as

$$\begin{aligned} |\mathcal{C}_E| &= \frac{1}{2} \sum_{j=1}^{p} X(:,j)^T \left(\sum_{i=1}^{m} \lambda_i u_i u_i^T \right) X(:,j) \\ &= \frac{1}{2} \sum_{j=1}^{p} \sum_{i=1}^{m} \lambda_i X(:,j)^T (u_i u_i^T) X(:,j) \\ &= \frac{1}{2} \sum_{j=1}^{p} \sum_{i=1}^{m} \lambda_i (u_i^T X(:,j))^2 \end{aligned} \tag{3.2}$$

Let $s_{ij} = (u_i x_j)^2 / m_j$. Then, after substituting s_{ij} we have

$$\begin{aligned} |\mathcal{C}_E| &= \frac{1}{2} \sum_{j=1}^{p} \sum_{i=1}^{m} \lambda_i (u_i^T X(:,j))^2 \\ &= \frac{1}{2} \sum_{j=1}^{p} m_j \sum_{i} \lambda_i s_{ij} \end{aligned} \tag{3.3}$$

It can be verified that

$$\sum_i s_{ij} = 1 \quad \text{and} \quad \sum_j s_{ij} \leq 1.$$

Consider the second expression in Equation 3.3, which is minimized when $s_{11} = 1$ since m_1 and λ_1 are smallest. But $\sum_i s_{ij} = 1$ so $s_{i1} = 0, \forall i > 1$. The same logic shows that letting $s_{jj} = 1$ for all $j \leq p$ while forcing the remaining $s_{ij} = 0$ when $i \neq j$ will minimize the entire sum. Finally we arrive at the spectral bound

$$\frac{1}{2}\sum_j^p \lambda_j m_j \leq \frac{1}{2}\sum_{j=1}^p X(:,j)^T L X(:,j) = |\mathcal{C}_E|. \tag{3.4}$$

If we equate $m_j = |X(:,j)|$, then the solution to Formulation *MinPairsForm* can be bounded from below by the following quadratic program.

PROGRAM 1: *MinPairsProg*(G, ϵ)

- **Given:** Graph $G = (V, E)$ and $\epsilon \in \mathbb{R}$
- **Find:** minimum $\sum_j m_j^2$ such that,

  (1) $\sum_{j=1}^p m_j \leq m$ (Vertex Count).

  (2) $\frac{1}{2}\sum_j^p \lambda_j m_j \leq \epsilon$ (Edge Cut Size).

  (3) $m_j \in \mathbb{R}^+$ (Block Size Positivity).

3.1.1 Applying Program *MinPairsProg*(G, ϵ)

Program *MinPairsProg* can be solved by means of quadratic programming, or more generally, as a convex program. We show here, however, that the problem

is simple to solve using the method of Lagrange Multipliers, which yields a single algebraic solution. We evaluate the quality of the spectral bound on two random graphs and the Power Grid of the Philippines. This is accomplished by comparing its lower bound to an upper bound achieved by means of the METIS graph partitioning suite. To begin, consider the vector formulation of Problem $MinPairsProg$ where $\boldsymbol{m}$ and $\boldsymbol{\lambda}$ are the vectors with index i equal to the size of partition m_i and eigenvalue λ_i, respectively.

FORMULATION 2: $FunctionalForm(G, \epsilon, k)$

- **Given:** Graph $G = (V, E)$ and $\epsilon \in \mathbb{R}$
- **Find:** minimum $f(\boldsymbol{m})$ where $f, g, h : \mathbb{R}^k \to \mathbb{R}$ and such that
  1. $f(\boldsymbol{m}) = \boldsymbol{m}^T\boldsymbol{m}$,
  2. $g(\boldsymbol{m}) = \frac{1}{2}\boldsymbol{m}^T\boldsymbol{\lambda} - \epsilon$,
  3. $h(\boldsymbol{m}) = \boldsymbol{m}^T\mathbf{1} - m$,
  4. $g(\boldsymbol{m}) = 0$
  5. $h(\boldsymbol{m}) = 0$

The variable k designates the number of blocks of G after ϵ edges are removed, and m_i denotes the size of ith block. When G is initially connected, the number of blocks cannot exceed $\epsilon + 1$ after removing up to ϵ edges. So, for a given ϵ, we begin by fixing $k \equiv \epsilon + 1$.

In our solution, we seek to identify the values of each entry, m_i, by the method of Lagrange Multipliers (LM). LM looks for places on the intersection of level curves

$g(\boldsymbol{m}) = 0$ and $h(\boldsymbol{m}) = 0$ where the gradients of g, h, and f have the same orientation. At such points, f is also on a level curve and thus has a critical point. Therefore, we seek the solution to the following problem.

PROBLEM 5: $Lagrange(G, \epsilon, k)$

- **Given:** Graph $G = (V, E)$, $\epsilon \in \mathbb{R}$, and f, g, and h as defined in $FunctionalForm(G, \epsilon, k)$
- **Find:** solution to the following system of equations,

  1. $\boldsymbol{f}^{\boldsymbol{\nabla}} = \nabla f(\boldsymbol{m})$
  2. $\boldsymbol{g}^{\boldsymbol{\nabla}} = \nabla g(\boldsymbol{m})$
  3. $\boldsymbol{h}^{\boldsymbol{\nabla}} = \nabla h(\boldsymbol{m})$
  4. $g(\boldsymbol{m}) = 0$
  5. $h(\boldsymbol{m}) = 0$
  6. $\boldsymbol{\phi} = \boldsymbol{f}^{\boldsymbol{\nabla}} - \alpha \boldsymbol{g}^{\boldsymbol{\nabla}} - \beta \boldsymbol{h}^{\boldsymbol{\nabla}} = 0$

LEMMA 1: There is exactly one optimal solution, labeled $\mathcal{F}(G, \epsilon, k)$, to Problem $Lagrange(G, \epsilon, k)$.

PROOF: We may deduce from Problem 2 that:

$$\boldsymbol{f}^{\boldsymbol{\nabla}} = 2\boldsymbol{m} \tag{3.5}$$

$$\boldsymbol{g}^{\boldsymbol{\nabla}} = \frac{1}{2}\boldsymbol{\lambda} \tag{3.6}$$

$$\boldsymbol{h}^{\boldsymbol{\nabla}} = \mathbf{1} \tag{3.7}$$

The eigenvalues λ_i are derived from a real symmetric matrix and are, therefore, all non-negative. It is well known (see Fiedler [69]) that the multiplicity of the zero eigenvalue equals the number of connected components in G. It follows that, $\lambda_0 = 0$ for all G. So together with condition $\phi_0 = 0$ we know that $\beta = m_0/2$. Substituting this value for β into each of the remaining conditions for ϕ_i we have

$$m_i = \frac{\alpha\lambda_i}{4} + m_0.$$

Substituting these values into constraint $h(\boldsymbol{m})$ we have m_0 in terms of α,

$$m_0 = \frac{m}{k} - \frac{\alpha}{4k}\sum_{i=1}^{k}\lambda_i.$$

Let $x = \sum_{i=1}^{k}\lambda_i$ and $y = \sum_{i=1}^{k}\lambda_i^2$. Using constraint $g(\boldsymbol{m}) = 0$ we can solve for α,

$$\alpha = \frac{4(2\epsilon k - mx)}{yk - x^2}.$$

□

Point $\mathcal{F}(G, \epsilon, k)$ will not always correspond to a reasonable partition. In some cases, an index m_i may be less than zero. Here we cannot make sense of the given partition, but we can conclude that no solution with all positive indices can render a lower objective value whilst satisfying the cut size constraint.

3.1.2 Real Cuts with METIS

The METIS graph partitioning library is a software suite that finds small, but not the smallest, edge cuts in graphs according to various conditions. We use a variant

that seeks to achieve a specific quantity of blocks in the modified graph where each block is further restricted to a specified size. This information is supplied by the user to METIS as a *balance vector*, $\boldsymbol{b}$, where $|\boldsymbol{b}|$ is the number of desired blocks and b_i is the desired size of block i. For each graph we randomly generated 100,000 balance vectors (a random number of blocks each with randomly chosen size) and then calculated the best edge cut for $\boldsymbol{b}$ in G using METIS. An upper bound on $MinPairs(G, \epsilon)$ was formed for each value of ϵ by choosing the lowest balance vector $\boldsymbol{b}$ that returned an edge cut less than or equal to ϵ. We call this process the *Real Cut Algorithm*.

3.1.3 Random Graph Evaluation

In this section, we use our upper and lower bounds to evaluate the robustness of two common random graph models. The first graph is the Erdos-Renyi (ER) graph, G_{np}, which contains n vertices where every pair of vertices is connected with probability p by an undirected edge. The second graph is the Barabasi-Albert (BA) graph, G_{nm}, which contains n vertices that are *preferentially* attached by undirected edges according the following algorithm. Initially the graph consists of an m-vertex clique. New vertices are added to the graph one-by-one and are attached to m existing vertices each chosen randomly with probability proportional to their degree. Bollobas [36] and Holme [85] have shown using asymptotic analysis that the BA graph is more vulnerable than the ER graph to removal of the highest degree vertex, but currently little is known about how the two models relate in small finite graphs and subject to edge removal.

We applied our bounds to a family of ER graphs generated for various quantities of vertices, n, and edge probabilities, p. We also generated bounds for a BA family

with the same values for n and with m chosen so that each BA graph has roughly the same number of edges as the corresponding ER graph. Figures 3.3 – 3.6 show the results of 500 trial runs for each set of parameters. They are box-and-whisker plots with boxes representing the center 50% of the data (between the 25^{th} and 75^{th} percentiles) and whiskers representing the extent of the remaining data.

Several conclusions can be drawn from these plots. The first relates to the general performance of our bounds. We see that the bounds universally weaken as *i)* the number of edges removed increases, *ii)* the size of the graph, n, grows larger, and *iii)* the total number of edges in the graph increases. In each case, this weakening can be attributed to the increase in the number of degrees of freedom in the solution space. That is to say, there are more possible solutions, therefore, the value of the optimal solution becomes more difficult to characterize.

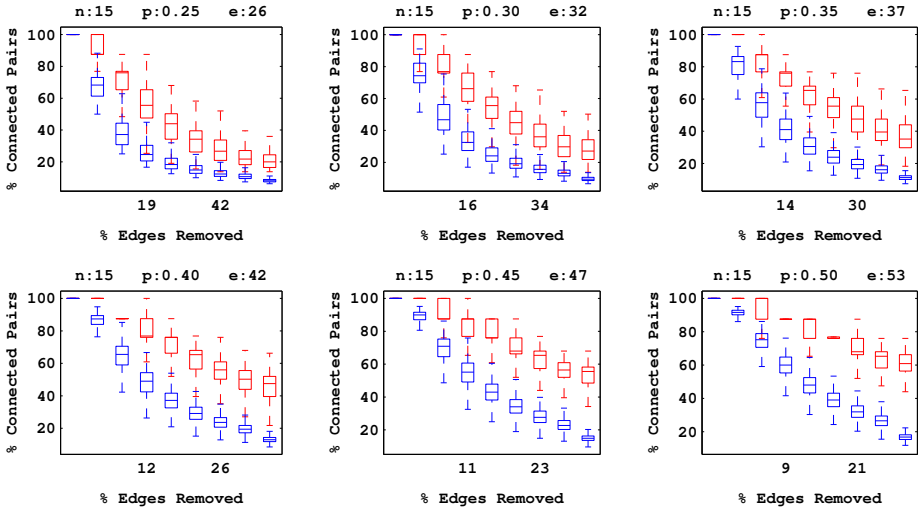
The second conclusion we draw is that the deviation between individual trials is higher in the ER graphs than in the BA graphs, particularly for smaller values of n. This difference indicates that the robustness of BA graphs is remarkably consistent for a fixed size and edge quantity relative to the ER graphs. Unfortunately, it's difficult to determine a mechanism behind this phenomenon, though it could indicate that the algorithm that generates BA graphs is somewhat *less random* than the ER algorithm. That is to say, a relatively small number of variations of BA graphs may dominate the set of most likely constructions.

Finally, we see that, to within the accuracy provided by our bounds, the ER and BA graphs exhibit almost identical robustness to edge removal in terms of the quantity of connected pairs of vertices. To the best of our knowledge, ours is the

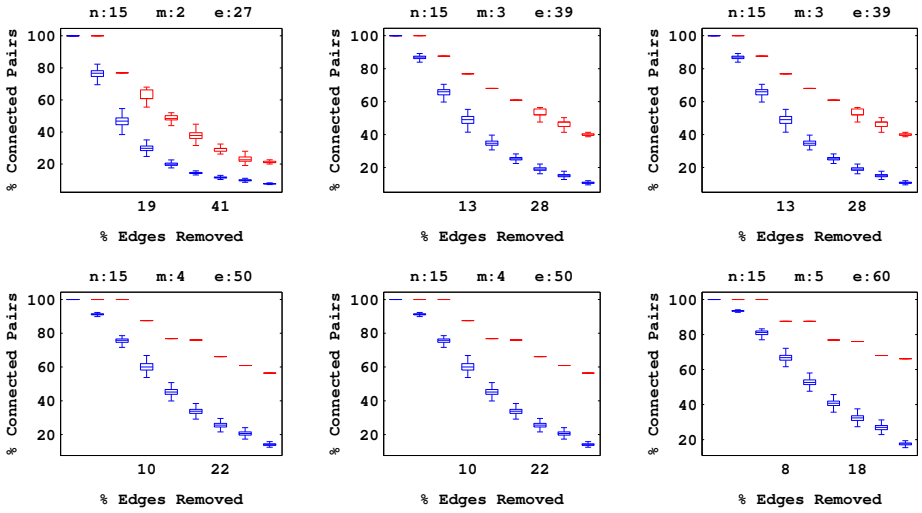
first work to demonstrate this similarity for edge removal and over a wide variety of graph parameters. We emphasize that these results are drawn for small graphs and are not necessarily indicative of asymptotic performance. The similarity is interesting because the ER and BA graphs are constructed according to two very different procedures. Specifically, the expected degree distributions will be very different as n grows large. An ER graph will tend toward a Poisson distribution while a BA graph will tend toward a power-law distribution. However, as Figures 3.7 and 3.8 show, these distributions are not realized for the small graphs that we evaluate. For 25 and 50 nodes graphs, the degree distributions are very similar, which implies that the topologies are similar as well. As n grows large, these distribution will deviate, and we expect that the similarity in robustness will vanish. It remains an open question to identify what random graph model will emerge as the most robust in that case.

3.1.4 Evaluation on a power grid

We also tested our technique on the graph corresponding to the Power Grid of the Philippines (see Section 1.3). From Figure 3.9 we see that the lower bound stayed roughly within a factor of three of the upper bound. The results themselves seem to indicate that the Power Grid of the Philippines is susceptible to partitioning by power line removal. When 50 lines are removed, even the upper bound shows only about 80,000 pairs of stations being connected. This is down more than 100% from the total pair-wise connectivity found before removal. It also implies that the size of the largest connected component cannot be larger than about 290 stations, which is only about 70% of the entire grid. But in order for all stations to receive power the grid does not need total connectivity. There are some 44 generators among the

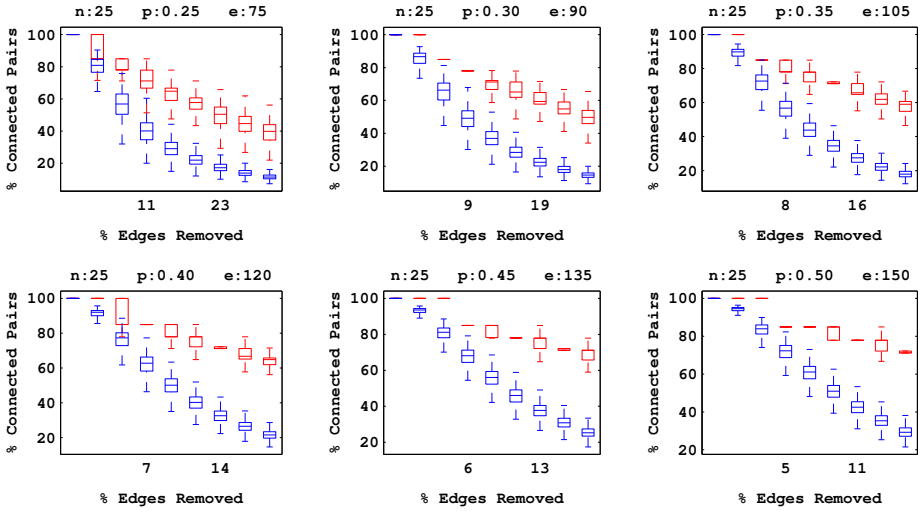


(a) ER: 15 vertices

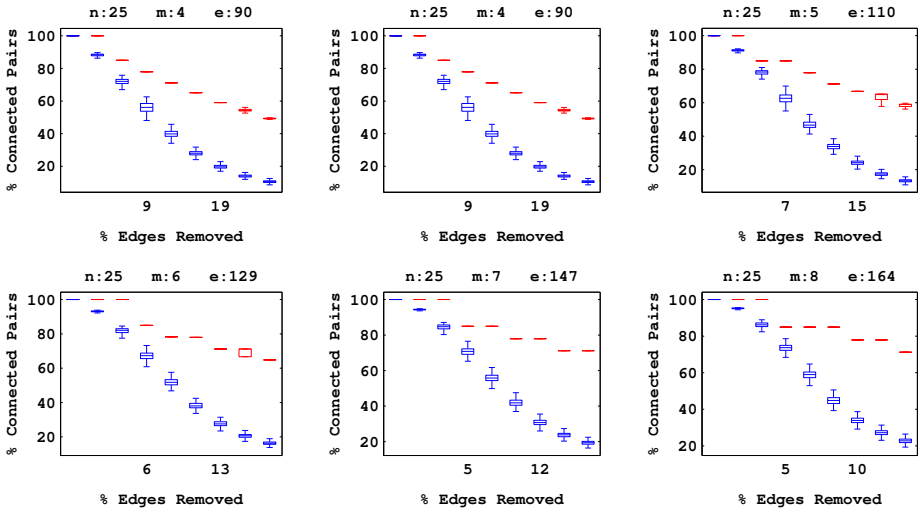


(b) BA: 15 vertices

Figure 3.3. Upper and lower bound on edge removal in 15 node random graphs of the (a) Erdos-Renyi and (b) Barabasi-Albert varieties over 500 trials.

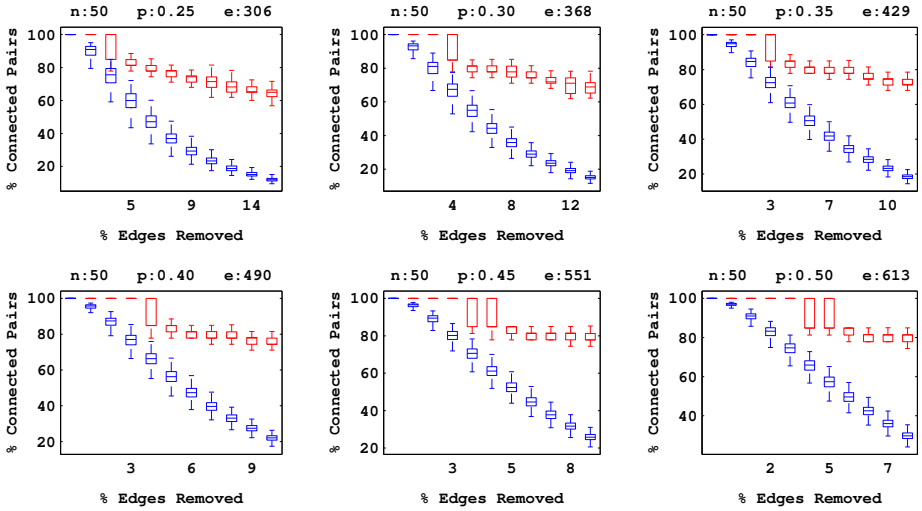


(a) ER: 25 vertices

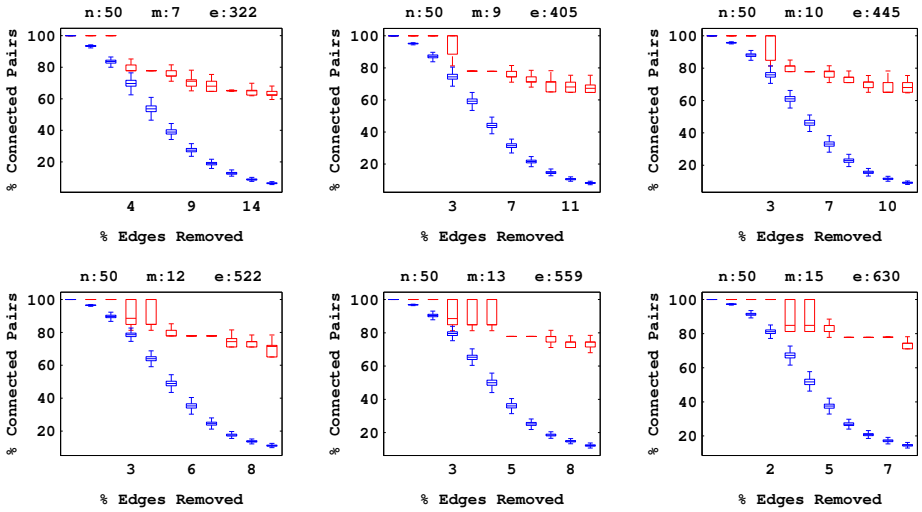


(b) BA: 25 vertices

Figure 3.4. Upper and lower bound on edge removal in 25 node random graphs of the (a) Erdos-Renyi and (b) Barabasi-Albert varieties over 500 trials.

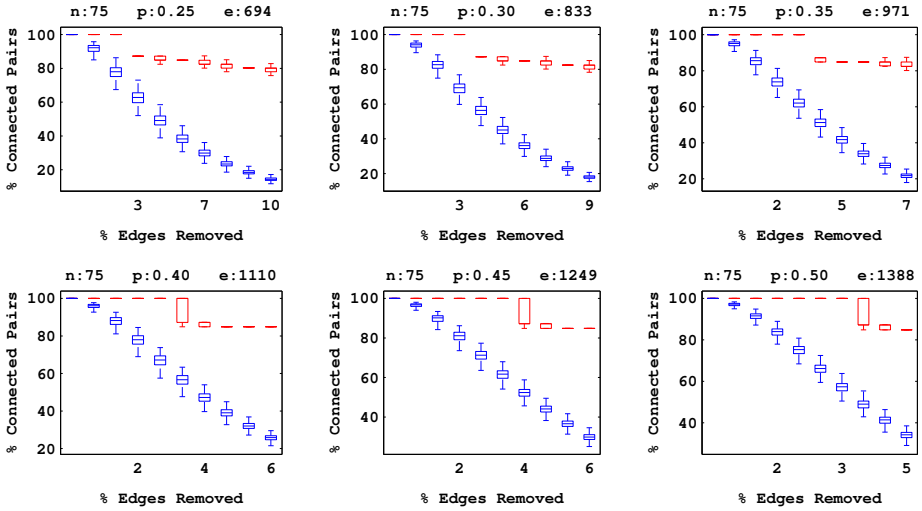


(a) ER: 50 vertices

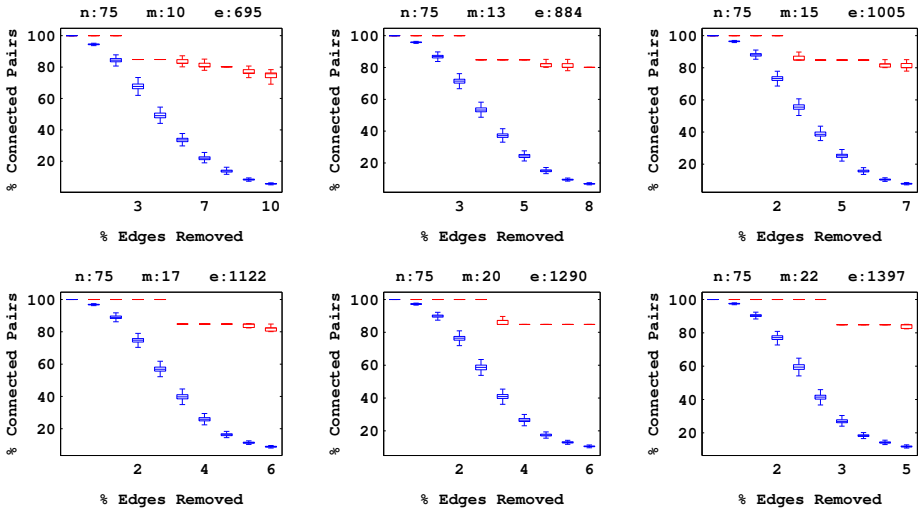


(b) BA: 50 vertices

Figure 3.5. Upper and lower bound on edge removal in 50 node random graphs of the (a) Erdos-Renyi and (b) Barabasi-Albert varieties over 500 trials.

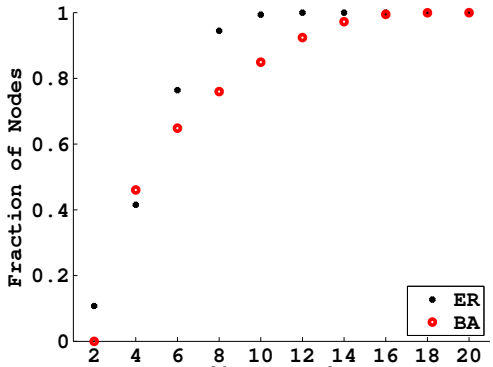


(a) ER: 75 vertices

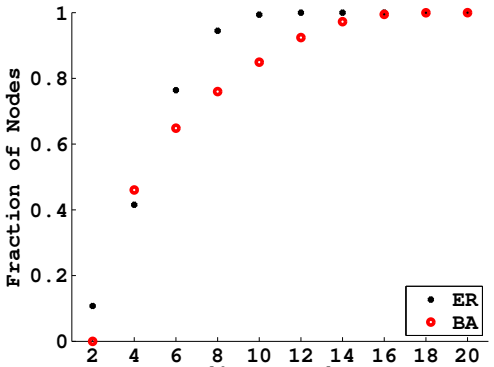


(b) BA: 75 vertices

Figure 3.6. Upper and lower bound on edge removal in 75 node random graphs of the (a) Erdos-Renyi and (b) Barabasi-Albert varieties over 500 trials.

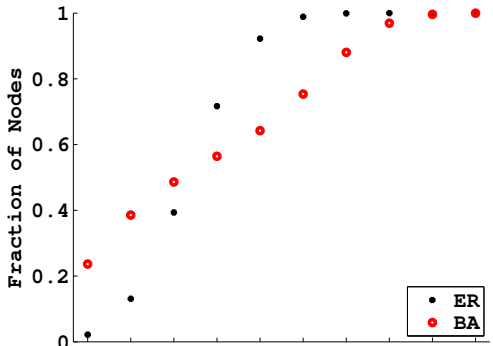


(a) ~ 75 edges

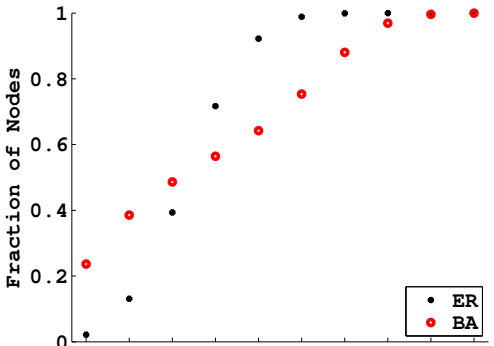


(b) ~ 150 edges

Figure 3.7. Degree distributions for 25 node graphs with (a) ~ 75 edges and (b) ~ 150 edges.



(a) ~ 306 edges



(b) ~ 612 edges

Figure 3.8. Degree distributions for 50 node graphs with (a) ~ 306 edges and (b) ~ 612 edges.

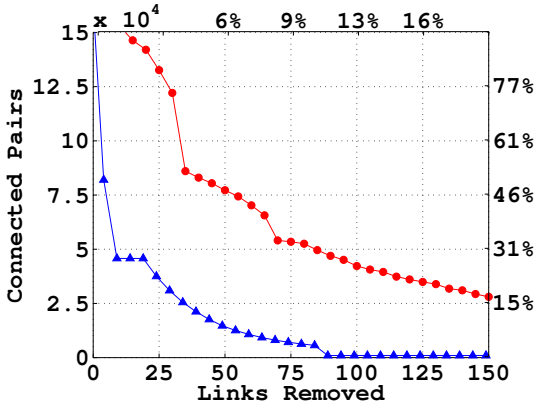


Figure 3.9. Disconnecting the Power Grid of the Philippines. The red line (with circles) indicates the result of an actual segmentation. The blue line (with squares) is our spectral lower bound.

404 total stations that can actually provide power to the remaining substations. So even if substantial division can be created within the grid, most substations will continue to receive power. The lower bound tells us that even when any 25 power lines are removed, there is still significant cohesion among the connected components with each station maintaining connectivity with 1/3 of all other stations on average. This suggests that the Power Grid of the Philippines may actually be quite robust to attack, but looking at basic connectivity between pairs of stations does not provide the full picture. In Chapter 4 we will introduce a new graph model that is capable of distinguishing between power generators and substations and will allow us to better understand the robustness of the grid.

3.2 Bolstering a Graph Prior to Attack

In many applications, it's possible to augment a graph knowing that it may experience either intentional or accidental edge loss in the future. Using the Power Grid of the Philippines as an example, the local government might be aware that the lines are aging and therefore they would expect some quantity to fail with high probability in the near future. Instead of replacing all lines, they might wish to augment the existing grid so as to add a measure of redundancy. In a different scenario, the government may instead know that some subversive group is planning to attack the grid but that this group has limited resources. In this section we show how to use our existing spectral lower bound on $MinPairs(G, \epsilon)$ to provide a lower bound on $MiniMaxPairs(G, \epsilon_1, \epsilon_2)$. The new lower bound can be interpreted as follows, if $MiniMaxPairs(G, \epsilon_1, \epsilon_2) = c$, then even after choosing the best ϵ_2 new edges to augment G, we will not be able to guarantee connectivity greater than c connected pairs after ϵ_1 edges have been removed. We begin with the following classical result regarding the eigenvalues of the Laplacian of a graph.

THEOREM 1: (Eigenvalues Interlace, see Mohar [114])

Let $G = (V, E)$ be a graph and $G^1 = G \oplus \{d\}$ be the same graph with any single edge d added. If the ordered and nondecreasing eigenvalues of the Laplacian matrix associated with G and G^1 are given by $\lambda_1, \lambda_2 \ldots$ and $\lambda_1^1, \lambda_2^1 \ldots$ respectively, then

$$\lambda_1 = \lambda_1^1 \leq \lambda_2 \leq \lambda_2^1 \leq \lambda_3 \leq \lambda_3^1 \ldots$$

Informally, the theorem states that the spectrum of G^1 is bounded above by the spectrum of G shifted over by one index. Let G^j denote a graph that is the same as G but with j arbitrarily added edges, and label its ordered and nondecreasing eigenvalues $\lambda_1^j, \lambda_2^j \ldots$, so that we have,

COROLLARY 1: Let G^j be a j-edge augmentation of graph G. For any edge cut producing block sizes $m_1, m_2, \ldots, m_k$ in G^j we have,

$$\frac{1}{2}\sum_{i=1}^{k} m_i \lambda_i^j \leq \frac{1}{2}\sum_{i=j+2}^{k+j} m_i \lambda_i$$

PROOF: The first eigenvalue remains zero because the first eigenvalue of the Laplacian of a connected graph is always 0 [69]. The other eigenvalues can be bounded by applying Theorem 1 multiple times so that $\lambda_i^j \leq \lambda_{i+j}$ for every $i \leq |V| - j$.

□

Using Corollary 1 we can place an upper bound on the edge cut constraint in Program *MinPairsProg* after *any* edge augmentation of a fixed size. The intuition is that we know that adding edges will make it harder to create more blocks and more diffuse blocks after edge removal. Our new bound gives the most optimistic bound on block size and quantity after edge augmentation. The following program incorporates the new edge cut bound so that its solution to this program is a relaxation of the solution to Problem *MiniMaxPairs*.

PROGRAM 2: $MiniMaxPairsProg(G, \epsilon_1, \epsilon_2)$

- **Given:** Graph $G = (V, E)$, number of edges to remove $\epsilon_1 \in \mathbb{R}$, and number of edges to add $\epsilon_2 \in \mathbb{R}$

- **Find:** minimum $\sum_i m_i^2$ such that,

  (1) $\sum_{i=1}^{p} m_i \leq m$ (Vertex Count).

  (2) $\frac{1}{2} \sum_{i=\epsilon_2+2}^{p+\epsilon_2} \lambda_i m_i \leq \epsilon_1$ (Edge Cut Size).

  (3) $m_i \in \mathbb{R}^+$ (Block Size Positivity).

Program *MiniMaxPairsProg* quantifies the extent to which edge augmentation can improve our earlier bound on Problem *MinPairs*. If we were to actually augment G with all possible combinations of ϵ_2 new edges to form graph H, then the greatest bound found by Program $MinPairsProg(H, \epsilon_1)$ will not be greater than the bound returned by $MiniMaxPairsProg(G, \epsilon_1, \epsilon_2)$. This new program is identical to Program *MinPairsProg* in every respect except that the spectrum has been shifted. Accordingly, it can be solved using the same techniques developed for solving Program *MinPairsProg*.

CHAPTER 4

CLIENT CONNECTIVITY AFTER MIXED REMOVAL

Program *MinPairsProg* has two failing points. First, it trades exact information about partition cost for weaker, spectral, information. In fact, it only uses the eigenvalues of L. Second, in deriving the spectral bound on the size of the edge cut, it searches the space of block sizes as opposed to actual vertex assignments. The ramification of this abstraction is an inability to handle vertex removal. We next set out to address both problems. To do so, we first extend Formulation 1 to handle metric $Conn(G)$ and cast it in terms of client-server graphs. Our ultimate goal is to derive upper and lower bounds on $MinConn(G, \nu, \epsilon)$ by means of a semidefinite program (SDP). Recall that the metric $Conn(G)$ operates on a client-server graph $G = (V, E)$, such that $V = S \cup C$, where it measures the quantity of clients in C with some path to at least one server in S. Problem $MinConn(G, \nu, \epsilon)$ seeks the smallest value of $Conn$ after some combination of ν clients and ϵ edges have been removed from G.

For Problem *MinConn* we also use the block indicator matrix X first introduced in Chapter 3 except in this instance we designate just three blocks: B_1, B_2, and B_3, with $|B_i| = m_i$. Again, we assume that there are a total of m vertices (i.e., $|V| = m$). The block indicator matrix for G is a $|V| \times 3$ matrix X, where

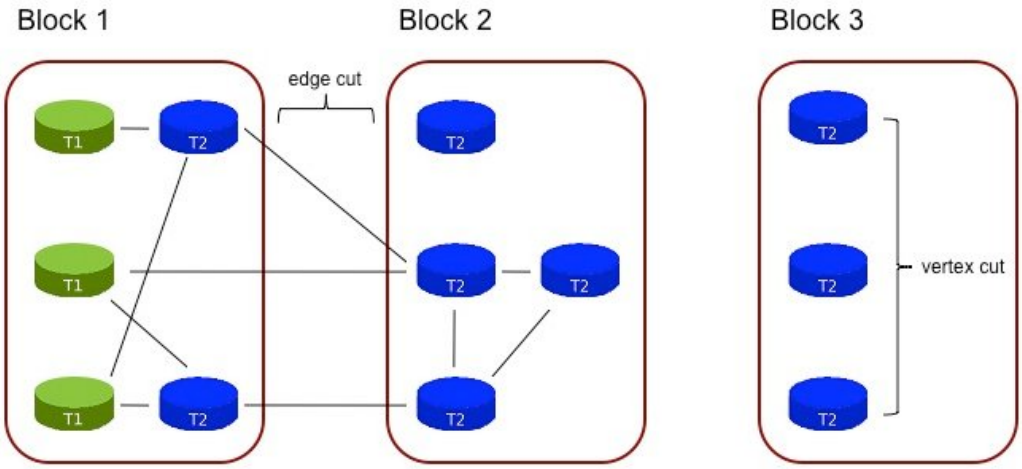


Figure 4.1. Separating the AS graph into blocks. Tier 1 ASes (in green) are servers and Tier 2 ASes (in blue) are clients.

$$X(i,j) = \begin{cases} 1, & \text{node } i \in B_j \\ 0, & \text{otherwise} \end{cases} . \tag{4.1}$$

It follows that

$$\sum_{i=1}^{m} X(i,:) = 1,$$

and

$$\sum_{j=1}^{3} X(:,j) = m_j,$$

with the operator ":" being used as described in Section 1.4.

We next seek to frame the *MinConn* problem in terms of the three blocks we have created above, which is illustrated in Fig. 4.1. These blocks will correspond to a segmentation of serviceable, non-serviceable, and removed clients in G. Specifically, the first block is the server block, to which all server vertices are fixed, and clients

with paths to these servers are placed. Clients without paths to servers are placed in the second block. Finally, the third block is the *null* block, which is where clients that have been removed from the graph are placed. The set of removed vertices and edges are called the vertex and edge *cuts*, and they are labeled $\mathcal{C}_V$ and $\mathcal{C}_E$ respectively. Figure 4.1 provides an example for the AS graph where Tier 1 nodes are regarded as servers and Tier 2 nodes as clients. We have the following QCQP formulation.

FORMULATION 3: $MinConnQCQP(G, \nu, \epsilon)$

- **Given:** client-server graph $G = (S \cup C, E)$ and $\nu, \epsilon \in \mathbb{R}$.
- **Find:** minimum $\parallel X(:,1) \parallel$ such that,

(1) $X(i,1) = 1, \ \forall i \in S.$ (All servers in Block 1)

(2) $\sum_{i=1}^{|V|} \sum_{j=1}^{3} X(i,j) = |V|.$ (Vertex Count)

(3) $|\mathcal{C}_V| \leq \nu.$ (Vertex Cut Size)

(4) $|\mathcal{C}_E| \leq \epsilon.$ (Edge Cut Size)

(5) $X(i,j) \in \{0,1\}$ (0–1 property)

Note that $\mathcal{C}_V \equiv B_3$ so

$$|\mathcal{C}_V| = \sum_{i=1}^{|V|} X(:,3). \tag{4.2}$$

Finding an expression for $|\mathcal{C}_E|$ is just a bit more complicated. We begin with a short digression. Vector $X(:,j)$ indicates the vertex membership of block j. That is to say, $X(i,j) = 1$ if vertex i is in block j and is 0 otherwise. For an arbitrary matrix M,

the quantity $X(:,j)^T M X(:,j)$ gives the sum of entries in the sub-matrix indicated by $X(:,j)$. Define D to be the matrix with row sums of adjacency matrix A along its diagonal, and further define the *Laplacian* matrix $L = D - A$. The quantity $X(:,j)^T D X(:,j)$ gives twice the *total* number of edges incident to vertices indicated by $X(:,j)$, and $X(:,j)^T A X(:,j)$ gives twice the number of edges *fully contained* in block j. It follows that $X(:,j)^T L X(:,j)$ gives twice the number of edges incident to, but not fully contained in, block j.

To quantify $|\mathcal{C}_E|$, we seek to bound the number of edges between blocks 1 and 2, that is, all edges connecting the two blocks. This quantity is captured by the following

$$|\mathcal{C}_E| = \frac{1}{2}\sum_{j=1}^{2} X(:,j)^T L X(:,j) - \frac{1}{2} X(:,3)^T L X(:,3). \tag{4.3}$$

The first term gives the total number of edges that lie between *any* of the three blocks. The second term subtracts the number of edges that pass between the null block and the others.

Figure 4.2 demonstrates the objective and edge and client cut constraints delineated by Formulation *MinConnQCQP*. The objective is to move as many client vertices out of the first block as possible without violating these two constraints. The edge cut constraint can be visualized as the number of edges that pass between the first and second blocks. The client cut constraint is much simpler, it is the quantity of clients placed in block three.

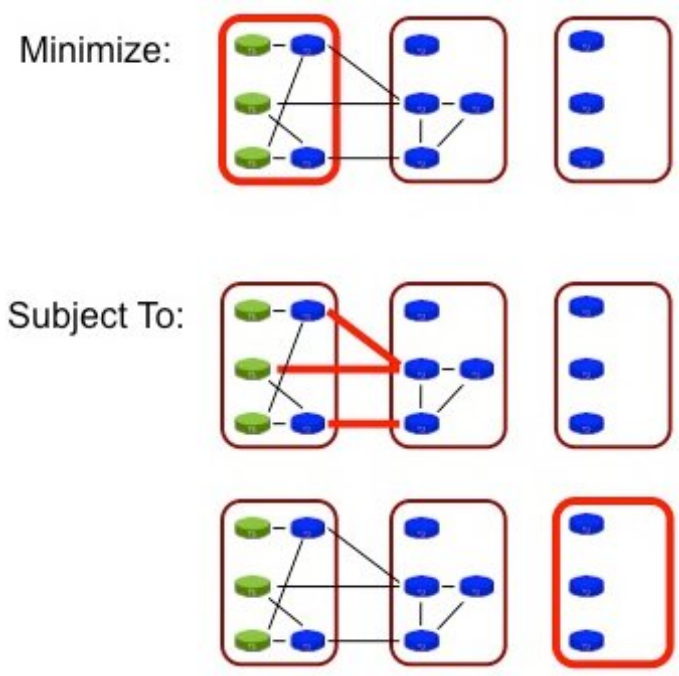


Figure 4.2. *MinConn* problem formulation.

4.1 SDP Formulation

The QCQP formulation cannot be solved directly because there is no mechanism to enforce orthogonality between block vectors; hence there is no way to force each vertex into a single partition. Directly enforcing orthogonality is not possible in a QCQP. Therefore, we must graduate to a richer language, that of semidefinite programming. Every semidefinite program with an $N \times N$ constraint matrix can be solved in time $O(N^3)$ [82]. A semidefinite program is any problem of the form

Minimize: $C \bullet Z$

Subject to: $A \bullet Z = b$, and $Z \succeq 0$,

where Z is the optimization variable and A, C, and b are arbitrary constants. Dot notation is used to indicate the *Frobenius inner product*: $A \bullet B = trace(A^T B) = \sum_{ij} A_{ij} B_{ij}$. The expression $Z \succeq 0$ indicates that Z is a positive semidefinite matrix. That the optimization variable, Z, is a positive semidefinite matrix is the fundamental assumption in semidefinite programming. A positive semidefinite matrix is any matrix M such that $x^T M x \geq 0$ for all real vectors x. Our analysis relies on the fact that for all real vectors y, the outer product matrix $Y = yy^T$ is always positive semidefinite.

The Frobenius inner product gives the component-wise product of two matrices. So, for example, the ijth entry of $A \bullet Z$ is equal to the product of the ijth entry of A and the ijth entry of Z. Our goal is to implement each of the constraints in Formulation $MinConnForm$ in terms of a semidefinite optimization variable. To that end, let vec(X) be the vector formed by concatenating the columns of X. Let

$$y = \begin{bmatrix} 1 \\ \text{vec}(X) \end{bmatrix}. \tag{4.4}$$

Define the *lift matrix* associated with X as $Y = yy^T$. Wolcowicz and Zhao [144] were the first to use this representation in the context of graph separators. First order terms are found along the first row and column, while second order terms are easily extracted from the remainder of the matrix. For notational simplicity, we number the rows and columns of Y beginning with index zero. We will also use the function $\phi(i, b) = i + |V|(b - 1)$, which maps the entry $X(i, b)$ to its corresponding entry in vec($X(i, b)$). The matrix Y comprises all pairwise products of block indices of the form $X(i, b_1)X(j, b_2)$, as well as single indices $X(i, b_1)$. Term $X(i, b_1)X(j, b_2)$

indicates that vertex i appears in block b_1 *and* vertex j appears in block b_2. We have, $Y(\phi(i,b_1),\phi(j,b_2)) = X(i,b_1)X(j,b_2)$, while $Y(0,\phi(i,b_1)) = Y(\phi(i,b_1),0) = X(i,b_1)$. Now suppose that for some matrix B, $B(\phi(i,b_1),\phi(j,b_2)) = \beta$, then the term $\beta X(i,b_1)X(j,b_2)$ will appear in row $\phi(i,b_1)$ and column $\phi(j,b_2)$ of matrix $B \bullet Y$. Constraints can thus be built as scaled single and pairwise products of block indices.

THEOREM 2: There exists an SDP whose solution provides a lower bound to the solution to our QCQP formulation.

PROOF: It will suffice to rewrite the objective and each of the constraints of our QCQP formulation as an SDP in terms of the lift matrix Y, where Y is relaxed to the space of all semidefinite matrices. To that end let $n = |V|$, and define $1_{i\times j}$ and $0_{i\times j}$ to be the matrices of all ones and zeros, respectively, and having size $i \times j$. The operator *diag* is defined as in Matlab so that it either extracts the diagonal of a matrix or forms a diagonal matrix from a vector. Also, let $I_{i\times i}$ be the identity matrix of size $i \times i$ and the symbol "$\otimes$" denote the Kronecker product of two matrices. The Kronecker product $A \otimes B$ is equal to A after replacing each of the ij^{th} entries, a_{ij}, of A with the matrix $a_{ij}B$. The following is a suitable SDP formulation.

PROGRAM 3: *MinConnSDP*(G, ν, ϵ)

- **Given:** graph $G = (S \cup C, E)$ and $\nu, \epsilon \in \mathbb{R}$.
- **Find:** minimum $\mathcal{O} \bullet Y$ such that,

(1) $\mathcal{F}_{i1} \bullet Y,\ \forall i \in S.$ (Servers in Block 1)

(2) $\mathcal{S} \bullet Y = n.$ (Vertex Count)

(3) $\mathcal{V} \bullet Y = \nu.$ (Vertex Cut Size)

(4) $\mathcal{E} \bullet Y = \epsilon.$ (Edge Cut Size)

(5) $\mathcal{A}_{ij} \bullet Y,\ \forall i, j.$ (0–1 property)

(6) $\mathcal{U} \bullet Y = 1.$ **(Upper Left Equal 1)**

(7) $\mathcal{Z}_i \bullet Y,\ \forall i.$ **(Sum Vertex Values is 1)**

where

- $\mathcal{O} = diag(\begin{bmatrix} 0 & 1_{1\times n} & 0_{1\times 2n} \end{bmatrix})$
- $\mathcal{S} = diag(1_{|y|\times|y|})$ except that $\mathcal{S}(0,0) = 0$
- $\mathcal{V} = diag(\begin{bmatrix} 0 & 0_{1\times 2n} & 1_{1\times n} \end{bmatrix})$
- $\mathcal{E} = \begin{bmatrix} 0 & 0_{1\times 2n} & 0_{1\times n} \\ 0_{2n\times 1} & \frac{1}{2} I_{2\times 2} \otimes L & 0_{2n\times n} \\ 0_{n\times n} & 0_{n\times 2n} & -\frac{1}{2} L \end{bmatrix}$
- $\mathcal{A}_{ij} = 0_{|y|\times|y|}$ except that $A_{ij}(\phi(i,j), 0) = 1$ and $A_{ij}(\phi(i,j), \phi(i,j)) = -1$
- $\mathcal{F}_{ij} = 0_{|y|\times|y|}$ except that $F_{ij}(\phi(i,j), \phi(i,j)) = 1$
- $\mathcal{U} = diag(1_{|y|\times|y|})$ except that $\mathcal{U}(0,0) = 1$
- $\mathcal{Z}_i = 0_{|y|\times|y|}$ except that $Z_i(\phi(i,j), \phi(i,j)) = 1$ for all $j \in \{1,2,3\}$

Constraints for the edge cut and the 0-1 property were first introduced by Wolcowicz and Zhao [144], but we have modified the edge cut constraint to subtract the number of edges passing to the null block. The last two constraints, in bold, are new; we have introduced them for the purposes of numerical stability. When translating constraints for mathematical programs, it is often necessary to use decompositions that involve simpler and sometimes redundant (in the space of binary vectors) constraints.

□

4.2 Solving the SDP Formulation

Semidefinite Programs can be solved by standard software suites. We use SDPT3 for MATLAB. This yields a tight lower bound for very small graphs. However, as the size of the input graph increases, the solver fails to assign each vertex to a single block. Instead, vertices are fractionally assigned to all blocks. We address this issue by fixing a small number of vertices to a single block as part of a branch-and-cut algorithm. Our solution procedure is described as follows.

PROCEDURE 1: *MinConnBC*(G, ν, ϵ)

- Solve Program *MinConnSDP*(G, ν, ϵ) using SDPT3.
- Use branch-and-cut procedure to find a lower bound.
- Derive Upper bound by rounding.

The next section provides the details of this approach.

4.2.1 Lower Bound via Branch and Cut

Let $\mathcal{S}$ be the *value* of the semidefinite programming solution from Section 4.1. $\mathcal{S}$ offers a lower bound on the optimal solution to $MinConn(G, \nu, \epsilon)$. This bound can be strengthened by fixing a small set of vertices T, each to one of the three blocks. In doing so, we move the semidefinite variable Y closer to being a binary matrix, which shrinks the search space and ultimately leads to a tighter bound. On the other hand, we cannot be certain that the resulting solution is optimal because it's possible that the optimal solution does not admit the placement we have chosen. However, if we evaluate *all* $|Y|^3$ possible permutations of vertices T, then we can be certain that the lowest solution among them is no greater than optimal.

We choose the vertices of T one-by-one. The first vertex, v_0, is chosen to be that whose placement is most ambivalent in $\mathcal{S}$ — that is, closet to one-third in each block. For each branch on v_0, the next vertex, v_1, is chosen as the most ambivalent vertex assignment in the given branch. This process is continued until the entire set T is formed. Any given vertex i is fixed to block j by adding the constraint $\mathcal{F}_{ij} \bullet Y = 1$ to the SDP. To stem the exponential growth of solution branches, we can leverage the structure of the graph G itself to prune some of these branches. First, we can eliminate any branch that places more than ϵ pairs of connected vertices in different blocks (not including the null block). Second, we can cut any branch that places a vertex i in block 2 when i is adjacent to some server.

4.2.2 Upper Bound via Rounding

Our goal is to round some solution matrix Y^* from the solution tree formed via branch-and-cut into a concrete solution to Problem $MinConn(G, \nu, \epsilon)$. In principle,

any partial solution from the tree is a valid candidate, but in practice we find it difficult to round most solutions while simultaneously keeping the objective value low and satisfying the edge cut constraint. So we try multiple solution matrices and choose the best rounded solution among them. The best candidates are those matrices Y^* that are nearly binary, and for small graphs these are readily found in the solution tree. However, for large graphs, a binary solution does not emerge immediately, so we choose the best branches at the last level of the solution tree and follow them exclusively until a nearly binary matrix Y^* is found.

We now proceed under the assumption that a suitable matrix Y^* has been identified. Given Y^*, define Z^* to be the block matrix formed by extracting the diagonal of Y^* and being fashioned in a manner analogous to the block matrix X. Matrix Z^* is therefore an $|V| \times 3$ matrix with each row corresponding to a single vertex and possessing net weight 1. Entry $Z^*(i, b)$ indicates the presence of vertex i in block b, which may be fractional. Rounding Z^* means rounding each row so that a single column in that row is equal to 1. This constitutes an unambiguous assignment for each vertex.

Finally, we apply the Kernighan-Lin Algorithm [128] to reduce the number of edges in the cut. The the Kernighan-Lin Algorithm is an iterative procedure that begins by swapping every possible pair of vertices (v_0, v_1) one-at-a-time where v_1 is in block 1 and v_2 is in block 2. The pair that lowers the edge cut by the most is chosen and the vertices in the pair are *fixed* to their respective blocks. This procedure is repeated until there are no unfixed vertices remaining in one or both blocks. This

procedure is attractive because it preserves the objective value and is known to produce small edge cuts when the initial partition is a good one.

4.3 Empirical Evaluation

We compared the lower and upper bounds found by executing Procedure *Min-ConnBC*(G, ν, ϵ) in two different real world networks: a wireless mesh network from the Net Equality Project and the Philippine Power Grid.

4.3.1 Net Equality Wireless Mesh Network

In the context of a wireless mesh network, vertex removal implies that a relay has been disabled while link removal implies that the connection between two nodes has been disrupted. Here, a relay node is a client and gateway node a server; our trace had no information about the number of users (laptops, desktops, etc.) connected to clients. We investigated how the number of serviceable clients decreases as other clients and links are removed from the network.

The Net Equality Project [5] works to provide Internet access to low-income communities where residents live in dwellings within close proximity, such as apartment complexes. They deploy a small number of gateway nodes that provide direct access to the Internet and relatively inexpensive Meraki Mini access points (relays) around the community, where most relays are multiple hops from gateways. A resident has access to the Internet wherever he/she is in range of either a gateway or a relay that is ultimately connected to a gateway. We analyzed a snapshot of the Hacienda CDC housing development taken in 2007. At the time, 69 relays and 2 gateways

were connected by 279 links serving approximately 1,200 residents. We studied the removal of up to 18 clients and 20 links in the Net Equality Network.

Figure 4.3(a) shows the degradation of client serviceability in the network as links are removed. The blue triangles give a lower bound on the number of relays connected to the Internet after the indicated number links have been removed. Circles in red provide a complimentary upper bound on the number of serviceable clients. These values correspond to partial (lower bound) and complete (upper bound) block assignments for each vertex. Two major features dominate Figure 4.3(a), the first is a severe drop in client serviceability after allowing 12 links to be removed. The second is another drop when 19 links can be removed. These points coincide with events where large subgraphs are finally disconnected from both gateways. This behavior provides valuable insight into the fault tolerance and attack vulnerability of the Net Equality graph. Communication difficulties between a handful of relays will not usually cause a major disruption, but degradation is not smooth, in general.

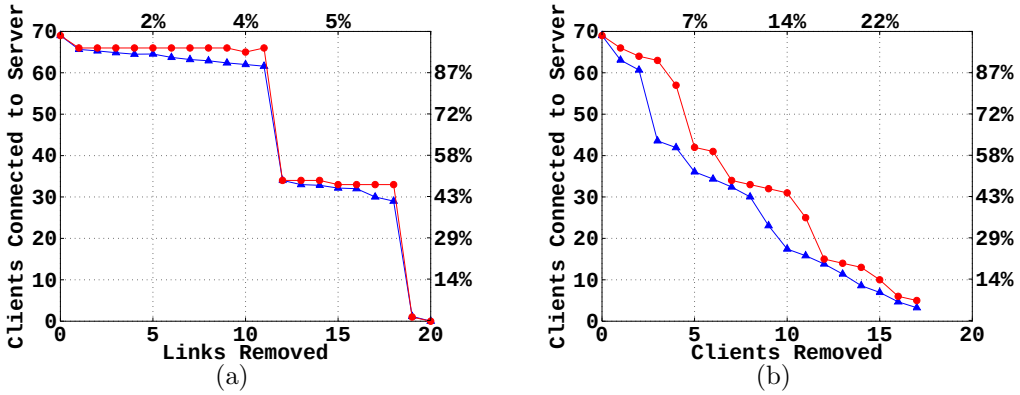


Figure 4.3. Net Equality Wireless Mesh connectivity with (a) link removal and (b) client removal.

Figure 4.3(b) offers a view of client removal. Again, triangles in blue indicate a lower bound on the number of remaining clients that are serviceable when the given number of clients have been removed. Circles in red offer a complimentary upper bound. In contrast to the link removal problem, client removal exhibits a much more uniform decline in serviceability. However, while the rate of deterioration is roughly uniform, it is much more severe for the same number clients removed as links. For example, removing 8 clients can disconnect as many as 39 clients from all servers, while removing 8 links will disconnect no more than 10 clients.

4.3.1.1 Philippine Power Grid

We also analyzed a network representing the Power Grid of the Philippines from 2006 [6]. The Grid comprised 44 power generators and 360 substations connected by 796 transmission lines. We modeled the Grid as a client-server graph with power generators corresponding to servers, substations corresponding to clients, and transmission lines corresponding to edges. We studied, independently, the effect of removing up to 50 transmission lines (links) or 50 substations (clients).

Figure 4.4(a) shows how power service degrades as the number of links removed increases. The lower bound departs somewhat from the upper, but does serve to establish that removing as many as 50 of transmission lines will not disconnect more than 50% of the substations. On the other hand, the plot also indicates that it *is* possible to disconnect roughly 20% of the substations after removing some 50 transmission lines.

Figure 4.4(b) shows client serviceability degradation with client removal. The most noticeable feature is the greater impact client removal has on client serviceability

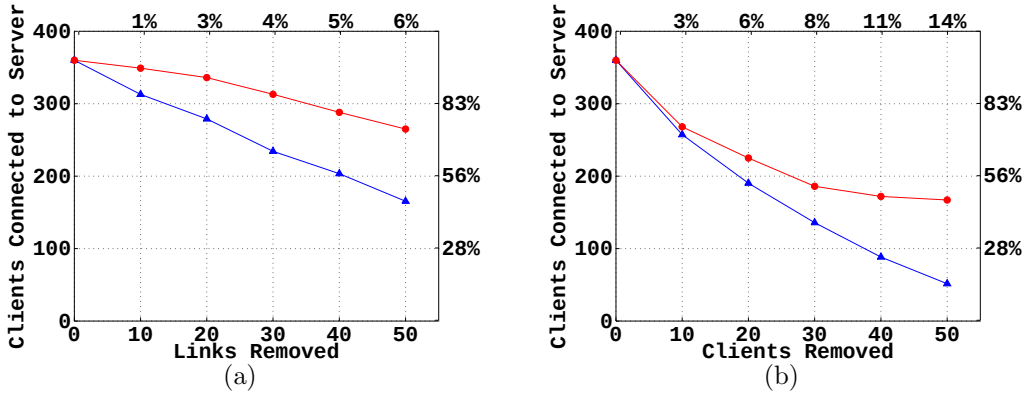
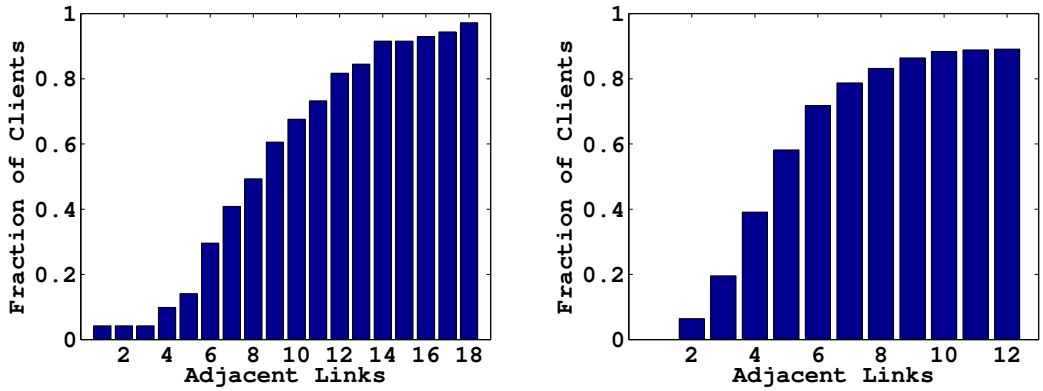


Figure 4.4. Connectivity in the Power Grid of the Philippines with (a) link removal and (b) client removal.

compared with link removal. The bounds also depart from each other more gradually than in the case of link removal. The figure shows that almost 50% of the substations can be disconnected by strategically removing just 20 other substations. In other words, removing fewer than 5% of the substations can leave half the remaining grid without service.

4.3.2 Topological Influence on Serviceability

Removing one client is always at least a damaging to serviceability as removing one link, so it makes sense that client removal is more damaging than link removal in both the Wireless Mesh and Power Grid networks. But the Wireless Mesh Network exhibits profound robustness even after a relatively large quantity of links have been removed while the Power Grid does not. Upon examination we find that clients in the former have a much more rich degree distribution than clients in the latter. Even though the Wireless Mesh Network has just 20% of the clients as the Power grid,



(a) (b)

Figure 4.5. Cumulative vertex degree distribution in (a) the Net Equality Wireless Mesh Network and (b) the Power Grid of the Philippines

more than half the clients in the Wireless Mesh network share a link with at least eight other nodes. In contrast, more than half the nodes in the Power Grid share four or fewer links with adjacent nodes. We hypothesize that it is this dramatic difference in degree distribution between the two networks that develops such a significant difference in robustness.

CHAPTER 5

CLIENT CONNECTIVITY IN LARGE GRAPHS

For large graphs, even the branch-and-cut procedure *MinConnBC* will fail because the semidefinite programming solver will fail to converge when attempting to solve *MinConnSDP* or it will return an extremely weak bound. Such failures typically occur when the size of the graph being analyzed becomes too large. In this section, we provide an additional solution to bound $MinConn(G, \nu, 0)$ that uses *MinConnSDP* as a subroutine in a divide-and-conquer algorithm. The subproblems are small enough that the semidefinite program will have no problem returning a high quality solution. Our approach is the first capable of providing both upper and lower bounds on connectivity after vertex removal in graphs with thousands of vertices. The limitation of this extension is that our analysis does not apply to edge cuts and it will not work well for every graph. Success is very topologically dependent.

5.1 A Divide-and-Conquer Solution for Client Removal

Let G be a client-server graph. Our strategy is to solve $MinConn(G, \nu, 0)$ by breaking G into subgraphs $G_1, \ldots, G_n$ by groups of clients that act as gateways for other clients. The connectivity properties of subgraph G_i can be found by solving a series of subproblems of the form $MinConnBC(G_i, \nu_i, 0)$. Once the connectivity

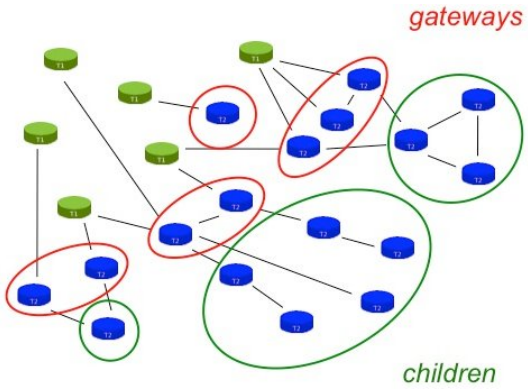


Figure 5.1. Client decomposition into gateways and children.

properties of each subgraph G_i have been quantified, the solution to $MinConn(G, \nu, 0)$ is formed by judiciously choosing those subproblems that render the lowest client-server connectivity.

Our objective is to count the minimum number of clients that remain connected to at least one server after the removal of some fixed number of clients. Therefore, we seek to count the minimum number of clients that retain a service path after the removal of some fixed number of other clients. Generally, the serviceability of a client can depend on the serviceability of every other client. However, in real world networks, this is rarely the case. We begin by describing how to isolate the serviceability of a subgraph in an arbitrary client-server graph.

For arbitrary $G = (S \cup C, E)$ define the set of *gateways*, $Y \subseteq C$, as those clients that are adjacent to at least one server in S. The remaining clients, labeled *children*, are placed in the child set L. Figure 5.1 shows such a decomposition on a hypothetical instance of the AS graph. A subset of children, L', are said to be *siblings* when they form a single connected component after all gateways are removed. For each set

of siblings L', there is a corresponding subset of gateways, Y' that are adjacent to the children in that set. We call such a subset Y' the gateways of siblings L' and conversely, we call siblings L' the children of gateways Y'. Together, the set $Y' \cup L'$ is called a *family*. The serviceability of the children in a family depends entirely on their connectivity to the gateways of that family. We begin by identifying all families in G, then independently measure how their serviceability is affected by client removal, and finally combine the results for each family.

5.1.1 Decomposition

The individual families of G are enumerated by first splitting clients into gateways and children, then finding groups of children that share the same gateways. These groups will form the basis of our delineation of a family. Algorithm 1 is responsible for performing this decomposition. In this algorithm, subroutine NEIGHBORS(G, c) returns the set of vertices in G that share an edge in E with vertex c. Subroutine COMPONENTS(G) returns a set of sets of vertices, with each set of vertices corresponding to a connected component of G. Subroutine CLOSURE$(\mathcal{F})$ returns the transitive closure of the union operation on each set $F \in \mathcal{F}$ — i.e., subroutine CLOSURE$(\mathcal{F})$ combines overlapping sets in $\mathcal{F}$ until all families in $\mathcal{F}$ are pair-wise disjoint.

Lines 1–4 are devoted to delineating two sets Y and L, which contain all gateways and children respectively. In line 6, families of G are initially formed as the connected components of the subgraph of G induced by the set of all children L. In lines 8–14, each family is enlarged to include the gateways for each set of siblings. Finally, in line

Algorithm 1 FAMILIES(G)

```
1: Y ← ∅                                              (gateways)
2: for s ∈ S do
3:    Y ← Y ∪ NEIGHBORS(G, s)
4: end for
5: L ← C \ W                                          (children)
6: B ← COMPONENTS(G(L))                               (siblings)
7: F ← ∅                                              (families)
8: for B ∈ B do
9:    F ← ∅
10:   for b ∈ B do
11:      F ← F ∪ NEIGHBORS(G(L ∪ Y), b) ∪ {b}
12:   end for
13:   F ← F ∪ {F}
14: end for
15: return CLOSURE(F)
```

15, we combine families that share common gateways so that each family is disjoint from every other family.

LEMMA 2: If F is a family in G and c is a child in F, then c is serviceable iff there is a path from c to some gateway in F.

PROOF: If there is a path from c to some gateway, then c must also have a service path since all gateways are adjacent to at least one server. Conversely, suppose that c is serviceable. Every service path from c must contain some gateway because gateways are the only clients that are adjacent to servers.

□

5.1.2 Family Serviceability

According to the above lemma, we can decouple the serviceability of clients that are placed in different families. The serviceability of each family can be found by solving a series of subproblems on the subgraph induced by this family. Gateways within a family act as servers in a new subproblem. Consider an arbitrary family F with gateway set Y and child set L. In general, we would like to solve the following intermediate problem, which gives the quantity of serviceable clients in F after removing a fixed number of clients from F.

PROBLEM 6: *PartialMinConn*(G, F, ν)

- **Given:** client-server graph $G = (S \cup C, E)$, family $F = Y \cup D$ and $\nu \in \mathbb{R}$.
- **Find:** minimum number of clients in F that are serviceable in $G \ominus A$, over all $A \subseteq F$ such that $|A| \leq \nu$.

The solution to Problem *PartialMinConn* can be bounded from below by solving a series of subproblems using *MinConnSDP*. We begin by providing an intuition for how these subproblems can be constructed and subsequently show how to derive the bound. By removing Y, all clients in F will be disconnected from all servers. So when $\nu = |Y|$, *PartialMinConn*$(G, F, \nu) = |F|$. When no gateways are removed, we have a client-server subproblem on $G(F)$ with gateways being servers. But gateways are also clients, so it is possible that they will themselves be removed. In general, we must consider removing ν clients from $Y \cup L$. This leads to a quantity of subproblems that is combinatorial in the size of the gateway set. There is one subproblem for each possible choice of unremoved gateways. For each $Y_i \subseteq Y$ being the set of

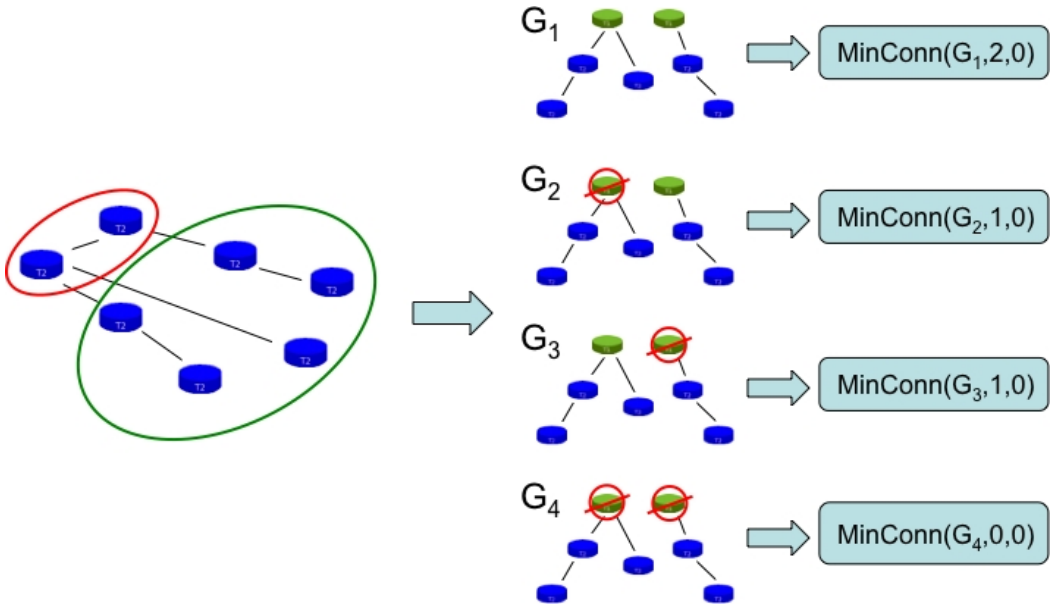


Figure 5.2. Client decomposition into gateways and children.

remaining gateways for a given subproblem, we construct a new client-server graph $G_i = G(Y_i \cup L)$ and solve $MinConnSDP(G_i, \nu - |Y_i|, 0)$. The following procedure formalizes the process.

PROCEDURE 2: $PartialProc(G, F, \nu)$

- For each $Y_i \in Y$, let $G_i = G(Y_i, L)$.
- For each $\nu = 0$ to $|Y| - |Y_i|$, find $MinConnSDP(G_i, \nu, 0)$
- Return minimum value over all subproblems.

Procedure *PartialProc* enumerates all combinations of removed gateways in Y, forms an associated client-server subgraph for each combination, and finally solves a subproblem on that graph using *MinConnSDP* that is fashioned so that no more than ν

clients are removed from F in total. The subproblem that returns the lowest value will provide a lower bound on the optimal solution to Problem $PartialMinConn(G, F, \nu)$ because we have exhaustively enumerated all possible subproblems on family F and, according to Theorem 2, the solution to *MinConnSDP* will always bound the solution to *MinConn* from below. Figure 5.2 provides a simple example of the subproblems generated by a single family when the maximum cut size is $\nu = 2$. Four subgraphs are initially formed: G_1, G_2, G_3 and G_4. Each subgraph corresponds to a choice of some set of removed gateways. The first leaves gateways intact and cuts 2 clients from among the children, the second and third remove one gateway and one child, and the fourth simply removes both gateways.

5.1.3 Aggregation

Our goal in this section is to consolidate local knowledge from the solution to *PartialMinConn* for each family, into a global measurement for *MinConn* on G. To do so, we formulate a knapsack-like problem by issuing a pair of scalars to each family called the maximum weight, W, and unit value, U. We then present a procedure that bounds $MinConn(G, \nu)$ from below by solving this problem for appropriate choices of W and U. Our first task is to derive the maximum weight and unit value for each family. Consider any family $F_i = Y_i \cup L_i$, by applying Procedure *PartialProc* we can evaluate how serviceability decreases with client removal in F_i. For each $\nu = 1$ to $|Y_i|$, let $S_{i\nu} = PartialProc(G, F_i, \nu)$. Define $W_i \equiv |Y_i|$ and

$$U_i \equiv \min_{\nu}\{PartialProc(G, F_i, \nu)/\nu\} \leq PartialMinConn(G, F_i, \nu)/\nu.$$

By construction, we have $\nu U_i \leq \mathit{PartialMinConn}(G, F_i, \nu)$. Algorithm 2 details the construction of the unit value for an arbitrary family F. The subroutine POWER(Y) returns the power set of set Y, or a set of all possible subsets of Y.

Algorithm 2 MULTIPLIER(Y, L)

```
 1: if L ≡ ∅ or |Y| ≡ 1 then
 2:    return |Y ∪ L| / |Y|
 3: end if
 4: x ← |Y|, y ← |L|, U ← y/x          (best initial guess)
 5: P ← POWER(Y)
 6: for P ∈ P do
 7:    for i = 1 to |Y| − |P| do
 8:       r ← MinnConnSDP(G(Y ∪ L) ⊖ P, i, 0)
 9:       y* ← y − r                   (clients out of service)
10:       x* ← i + |P|                       (clients removed)
11:       if x* ∗ U > y* then
12:          U ← y*/x*                 (choose highest ratio)
13:       end if
14:    end for
15: end for
16: return U
```

Algorithm 2 begins by testing for trivial cases where we can immediately identify the unit value multiplier. For all other cases (line 4), we begin by setting the multiplier to the most obvious ratio where all gateways are removed. It is possible that a better solution exists, and by iterating on all possible subproblems of this family, we progressively refine this choice of unit value multiplier. Line 8 finds the solution to each subproblem by running it through $\mathit{MinConnSDP}(G, \nu, 0)$. Note that in Chapter 4 we worked with relatively large graphs that did not immediately yield a strong solution to the semidefinite program, and therefore required Procedure *MinConnBC* for strengthening the result. For the subproblems in Algorithm 2,

however, we find that the problems are small enough that further analysis is not necessary. The solution to $MinConnSDP(G, \nu, 0)$ gives a *lower* bound on the number of clients that remain serviceable after removing some ν clients. For y^*, however, we require an *upper* bound on the total number of clients that are removed from service after removing ν clients. Line 9 reverses the output from *MinConnSDP* from number of clients connected to number disconnected. The remaining lines replace the old multiplier value only if the new value is greater.

Having maximum weight and unit value for each family, we now show how to use these values to bound the solution to $MinConn(G, \nu, 0)$. Consider the optimal solution to $MinConn(G, \nu, 0)$, and label A the set of clients removed by this solution. Set A will contain some subset of clients from each family F_i. Let $A_i = F_i \cap A$, or the set of clients removed from family F_i. Since $\nu U_i \leq PartialMinConn(G, F_i, \nu)$ for any ν, it follows that $|A_i|U_i \leq PartialMinConn(G, F_i, |A_i|)$. Aggregating over all families it follows that

$$MinConn(G, \nu, 0) \geq \sum_i |A_i|U_i.$$

This means that we can bound the solution to $MinConn(G, \nu, 0)$ from below by solving the knapsack problem that takes some integer portion of W_i from each family so that the sum does not exceed ν and so that the aggregate value is maximized. Procedure *MinConnDC* formalizes this process.

PROCEDURE 3: $MinConnDC(G, \nu, 0)$

- Form an exhaustive and disjoint decomposition of G into families $\mathcal{F} = F_1, F_2, \ldots$ according to Algorithm 1

- For each F_i, identify maximum weight W_i and unit value U_i using Algorithm 2.
- Derive upper bound by greedily choosing the gateway set with maximum weight no greater than the current available weight and that corresponds to the family with highest unit value. Continue while the aggregate weight is less than ν.
- Derive lower bound by solving linear program $SelectLP(W, U, \nu)$, which is a standard knapsack linear program except for an additional constraint that keeps the value of a family above 0.

PROGRAM 4: *SelectLP*(W,U,w)

- **Given:** weight and value multiplier vectors W and V, and maximum aggregate weight w.
- **Find:** maximum $\sum_i X(i)\ U_i$ such that,

  (1) $\sum_i X(i) \leq w.$

  (2) $X(i)\ U_i - W_i \leq 0, \forall i.$

  (3) $X(i) \in \{0, W_i\}, \forall i.$

5.1.4 Managing Subproblem Size

Algorithm 2 iterates over the power set of Y. This set becomes quite large for even small sets, so we aim to keep the number of gateways at 5 or fewer. However, we continue to require that families remain pair-wise disjoint so we must find a way

to *split* families with large gateway sets. For the purpose of creating a lower bound on $MinConn(G, \nu, 0)$, it's clear that removing edges will only lower the value of the solution and will not jeopardize the bound. So, we judiciously choose edges to remove so that there are 5 or fewer gateways corresponding to each family in the new graph. Since we are dealing with relatively small subgraphs, a simple spectral approach will suffice.

Ding et. al [59] showed how to use the eigenvector corresponding to the second smallest eigenvalue of the Laplacian Matrix of a graph to find nearly-disconnected components of that graph. Consider the graph $H = G(F)$ for any family F. The spectrum of the Laplacian of H has one eigenvalue equal to 0 for each connected component in H. In this case, zero-valued eigenvalues are the smallest because the Laplacian is always real and symmetric so its eigenvalues are always non-negative. Any eigenvector belonging to a 0 eigenvalue will have the same value for all indices that correspond to vertices in the same connected component. Matrix perturbation theory predicts that graphs that have nearly disconnected components will show similar eigenvector values between indices corresponding to vertices in each component when considering the eigenvector corresponding to the second smallest eigenvalue. The following procedure outlines the steps that we took to to break families with large gateway sets into groups of families with no more than 5 gateways.

Let $F = (Y, L)$ be a family in client-server graph G and define subroutine EIGENVECTOR(H) to be a function that returns the eigenvector corresponding to the second smallest eigenvalue of the Laplacian of H. If $|Y| > 5$, then it can be disassembled with the following steps.

- Compute $x \leftarrow \text{EIGENVECTOR}(G(F))$
- Let $k = \lceil |Y|/5 \rceil$, and break Y into k subsets labeled $Y_1, \ldots, Y_k$ where $|Y_i| = 5$ for $i < k$ and $|Y_k| = |Y| - 5(k-1)$.
- Assign clients of Y to one of the subsets so that clients in the same subset have similar entries in eigenvector x.
- Create k new families $F_1, \ldots, F_k$ so that family F_i has set Y_i as its gateways, and create child sets L_i for each family F_i with L_i initially empty.
- Assign each child in L to the family F_i whose average eigenvector entry matches closest to its own eigenvector entry.

5.2 Evaluation on the Internet AS Graph

Our first evaluation is on a snapshot of the Internet at the Autonomous Systems (AS) level taken by the CAIDA project in April 2005 [58]. Their taxonomy identifies *inferred* Tier-1 and Tier-2 ASes. We create a client-server graph with Tier-1 and Tier-2 ASes labeled as servers and clients, respectively, and with an edge set derived from the corresponding list of peering relationships. We assume that the backbone of the Internet (the Tier-1 ASes) remains fully intact, but that Tier-2 ASes are vulnerable to removal due to attack or administrative-level disenfranchisement. In all, there were 5,396 clients and 44 servers. Figure 5.3(a) shows the degree distribution over all vertices while Figure 5.3(b) shows a breakdown of those distributions between clients and servers. The client degree distribution dominates the server distribution since there are just 44 servers. These plots indicate that clients have a strongly

power-law degree distribution with a small number having very high degree and the rest having low degree (more than 90% have fewer than 10 incident edges). Servers, on the other hand, show the opposite phenomenon. More than 60% have at least 100 incident edges and fewer than 5% have less than 20 incident edges.

Let G denote the client-server graph that models the AS graph. We use *MinConnDC* to compute upper and lower bounds on $MinConn(G, n, 0)$ for various values of n. Figure 5.3(c) shows the results where the number of clients removed, n, varies from 0 to 2,500 in increments of 50. The blue curve in triangles is the lower bound delivered as the solution to our knapsack problem. The red curve in circles is an upper bound developed by actually removing from G those client vertices identified by the knapsack solution. In contrast to the performance of $MinConnBC(G, n, 0)$ from Chapter 4, the Knapsack solution has somewhat weak performance for small values of n, but tightens considerably for larger values. The initial weaker performance can be attributed to the error introduced by splitting families with large gateway sets. Algorithm 1 returned 2,759 families total, and among those there were 9 that had more than 5 gateways vertices. Because of the large number of subproblems associated with large gateway sets, we decomposed each of these according to the algorithm in Section 5.1.4. The largest set was found in one family that had 292 gateways. The rest had fewer than 30 gateways. By splitting the largest gateway set almost 60 times we introduced error. The lower bound makes it seem *easier* to disconnect service paths to a larger number of clients than is actually possible. However, once n is large enough that this large gateway set can actually be removed, the upper bound settles close to the lower. We believe that this type of tradeoff will be typical when applying

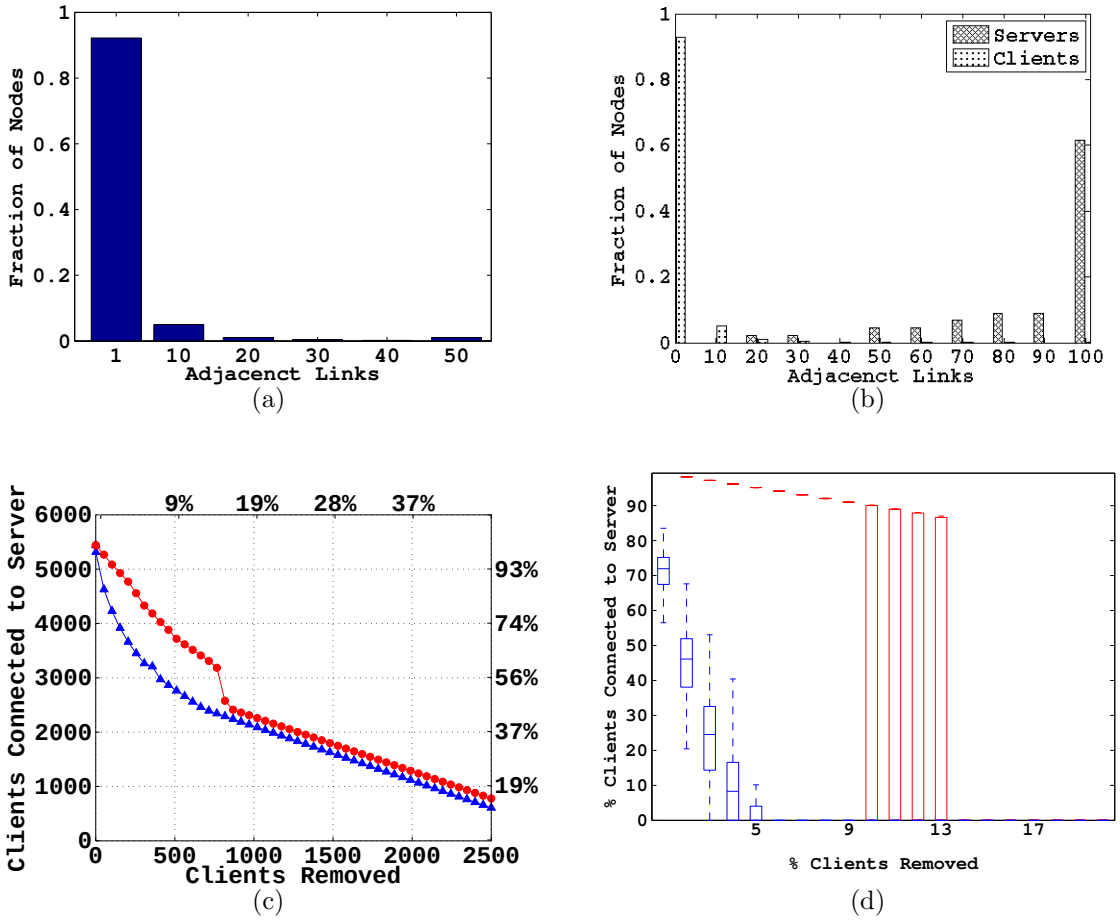


Figure 5.3. AS graph: (a) degree distribution over all vertices, (b) degree distribution of both clients and servers, (c) client connectivity after client removal, and (d) result of synthetic random graph experiment.

MinConnDC to any client-server graph with such hierarchical structure. Accuracy will largely depend on the extent to which the graph can be decomposed into families with the small gateway sets.

In our analysis we identified a small handful of families responsible for almost all of the complexity in the graph. They had large gateways sets and large sets of children. The remaining families had small gateway sets and small sets of children. They appear as commodities to the knapsack algorithm. Overall, the AS graph exhibits strong serviceability in the face of significant deterioration of Tier-2 clients. Even when 900 ($> 16\%$) Tier-2 clients are removed, better than half the Internet remains in service. This result appears in sharp contrast to research outlined in Section 2.2.2 where both Tier 1 and Tier 2 nodes can be removed. There, far fewer than 50% of all nodes remain in the connected component even when only 3% of the nodes have been removed [113, 148].

5.2.1 Random Graph Model

The degree distributions reflected in Figure 5.3(b) suggest that the AS graph can be modeled as a BA random graph (see Section 3.1.3 for a definition) with servers chosen from among the vertices with highest degree. To test this hypothesis, we randomly generated 100 BA graphs with $n = 5{,}440$ (the size of the AS graph) and m chosen so that each graph had approximately 14,428 edges (the quantity of edges in the AS graph). We randomly selected 44 servers from among the vertices of each random graph with the probability of selecting any given vertex being proportional to its degree. In this fashion, we ensured that the highest degree vertices were most likely to be labeled as servers. The remaining vertices were labeled clients. Figure 5.3(d)

shows box and whisker plots for the result of client removal in these graphs. The boxes show points falling between the 25^{th} and 75^{th} percentiles, and the whiskers show the extent of the remaining data.

The upper bound boxes in Figure 5.3(d) exhibit almost no variability for up to 9% of clients removed. For higher quantities, the number of clients that remain connected to a server varies widely from 0% to 90%. In contrast, the corresponding lower bound exhibits roughly uniform variability for up to 5% of clients removed. For larger quantities, none of the random graphs are guaranteed to maintain any server connectivity. Overall, the agreement between upper and lower bounds is poor. The plots show that, according to the lower bound, it's possible that no clients will remain connected with more than 5% of clients removed, but in terms of the upper bound, we can never find an actual cut set that lowers client connectivity to less than 90% for up to 9% of clients removed.

It's difficult to determine why the upper bounds for the random graphs become volatile across random trials for more than 9% of clients removed. A similar sudden drop is exhibited in the AS graph after approximately 15% of clients have been removed. This could mean that the dramatic drop is due to the same artifact found in the AS graph that arises from large gateway sets being divided. In terms of resilience, the bounds presented for random graphs are generally too weak to draw any strong conclusions about robustness to client removal relative to the AS graph. However, after 13% of clients are removed from any of the random graphs tested, no clients will remain connected to servers. But in the AS graph, at least half the clients will maintain server connectivity even when 15% of clients are removed. This indicates

that the AS graph is more resilient than the random graphs to large quantities of removed clients.

5.3 Evaluation on Airport Connectivity

We next address airport connectivity in the states of Iowa and Michigan. Airports are chosen as only an example of critical infrastructure, and as in our above cases, there are limitations to our analysis. We make the simplifying assumption that each airport is equally resourceful. We also do not claim this is the most effective attack on infrastructure, but it does serve to bound the severity of a particular kind of coordinated attack on a very import piece of national infrastructure. First, we formed a client-server graph with clients corresponding to key route intersection points, called *waypoints*, in the given state and whose value was approximately equal to the portion of the state's population residing near the waypoint (see Section 1.3 for details). Our objective was to bound the quantity of people who maintain highway access to *any* airport after the removal of some fixed number of waypoints. We imagine that the waypoints are removed by either natural disaster of coordinated attack.

We constructed a solution to the *WeightedMinConn* Problem for this scenario by modifying *MinConnDC* as follows: *i)* client values were incorporated by first modifying Algorithm MULTIPLIER to return the highest ratio of client *values* to gateways removed; *ii)* the objective function of Program *MinConnSDP* was altered to minimize the aggregate value of connected clients.

5.3.1 Iowa

To apply our *WeightedMinConn* solution, we constructed a client-server graph corresponding to the state of Iowa. The boundary of the state was delineated by the coordinates (-104.3701, 42.6738) and (-96.2842, 46.1566). In total there were 1,947 waypoints and 272 airports. The entire network was connected by 4,163 edges representing route segments. Algorithm FAMILIES returned 493 families total with just three families having more than 5 gateways. They had gateway sizes 8, 19, and 501. Breaking the family with 501 gateways into 101 subfamilies was the main source of error. It causes the lower bound to retreat substantially from the upper for small numbers of removed waypoints.

Figure 5.4(c) shows that despite the somewhat loose agreement between upper and lower bounds, we can still draw interesting conclusions about how airport access degrades with waypoint removal. For example, when just 200 waypoints (approximately 10%) are blocked, it is possible to disconnect at least 500K (15%) people from every airport, but no more than 1.7M (50%) can possibly be disconnected in the worst case. There is also a sudden drop in the upper bound after approximately 575 waypoints are removed. This is the point when the largest family (the one that began with 501 gateways) is finally completely disconnected. It's a critical point where the number of people connected drops by about 500K.

5.3.1.1 Random Graph Model

We developed a random graph model for the state of Iowa by attempting to match the qualitative properties of the graph. Figure 5.4(b) shows the distribution of both client and server vertex degrees. This distribution is quite different than that seen for

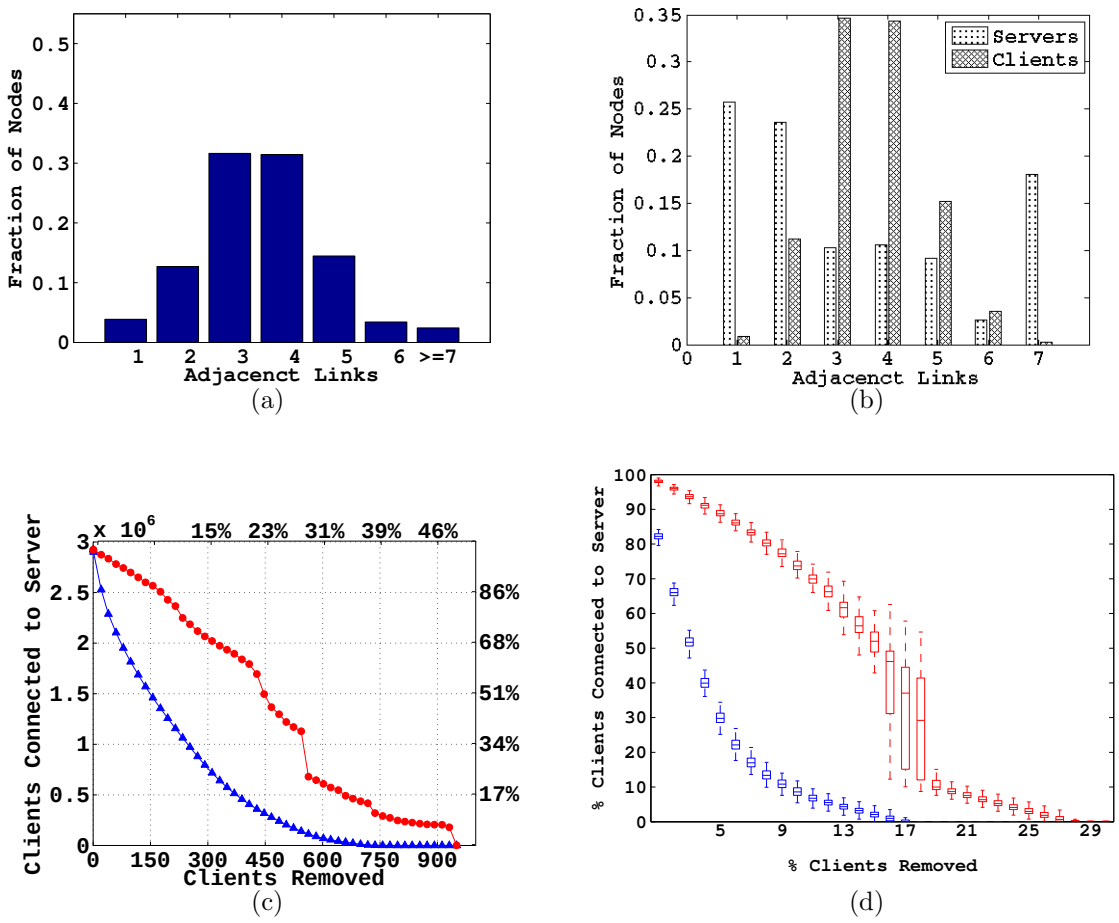


Figure 5.4. Airport connectivity and graph properties in Iowa.

the AS graph. Here, the clients exhibit an approximately normal degree distribution while the degrees of servers appears almost uniformly random. Accordingly, we randomly generated 100 ER graphs (see Section 3.1.3 for a definition) with $n = 2{,}219$ and p chosen so that the expected number of edges was 4,163. The ER model seemed like a good choice since the expected degree distribution of an ER random graph is the qualitatively similar Poisson distribution. For each graph, we chose 272 vertices to be servers uniformly at random. This choice was made to reflect the roughly uniform distribution of server degree in the Iowa graph.

Figure 5.4(d) shows box and whisker plots for the result of client removal in each of the 100 random graphs. The variability between trials for both the upper and lower bounds is much more consistent than in the random graph model of the AS graph. After approximately 15% of clients are removed, we again observe a sudden drop in the quantity of clients connected to servers for the upper bound. This occurs for smaller quantities of removed clients in the random graphs than in the Iowa graph. For Iowa, we see the largest drop in the upper bound when about 27% of clients are removed. The agreement between upper and lower bounds is also fairly weak for the random graph model. When 10% of clients have been removed, the upper bound predicts that at least 70% of clients will remain connected to servers. On the other hand, the lower bound shows that it's possible that only about 10% of clients will maintain server connectivity. The bounds generated for the random graphs are too weak for small quantities of removed clients for us to compare their resilience to the Iowa graph. However, the bounds do indicate that after 21% of clients have been removed from any of the random graphs, no more than 10% of remaining clients will

be connected to some server. In contrast, after removing the same number of clients in the Iowa graph, at least 15% of remaining clients will still be connected. This implies that the Iowa graph is more robust to client removal, particularly for large quantities of removed clients.

5.3.2 Michigan

We performed similar analysis for the state of Michigan where the corresponding client-server graph contained 2,207 waypoints and 285 airports. A total of 5,205 route segments connected these points. We chose to focus on the lower peninsula of Michigan (the "mitten") bounded by the coordinates (-87.4072, 41.5327) and (-82.2656, 45.7593). After applying Algorithm FAMILIES there were 806 families total with 15 having more than 5 gateway vertices. The three largest gateway sets were 129, 112, and 53. All other families had fewer than 25 gateways. Figure 5.5(c) shows how airport connectivity degraded with waypoint removal. Again, the families with large gateway sets were the major source of error in the lower bound. They forced a large gap between upper and lower bounds for up to 300 removed waypoints. Subsequently, the bounds tightened significantly. From the plot we can see that no more than 4.3 million people (43%) will maintain airport connectivity after 300 waypoints have been removed, and it's possible that as few as 2.8 million (28%) will maintain connectivity. In other words, at least half the residents of Michigan will lose airport access if fewer than 14% of the waypoints are removed.

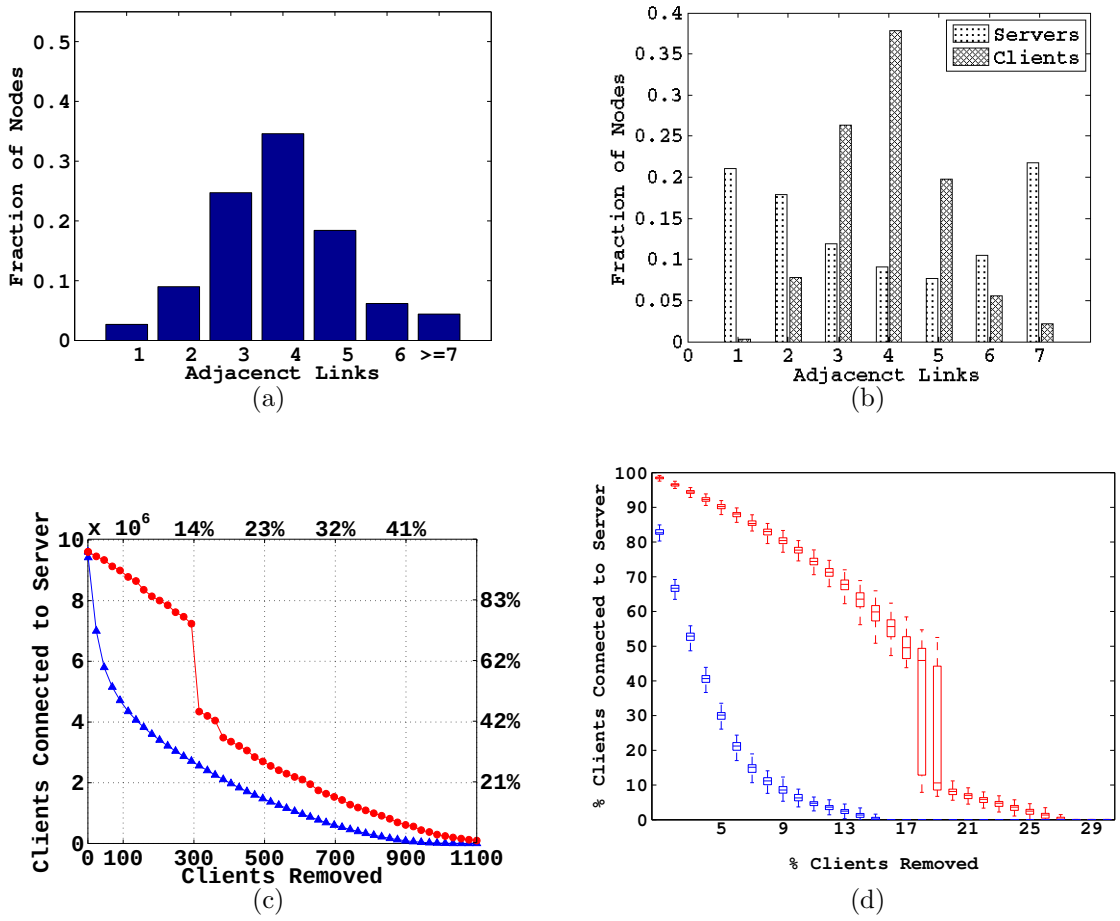


Figure 5.5. Airport connectivity and graph properties in Michigan.

5.3.2.1 Random Graph Model

Figure 5.5(b) shows that the degree distribution of clients and servers in the Michigan graph is very similar to the corresponding distributions in the Iowa graph. For this reason, we developed a random graph model that is fundamentally identical to the random graph model used for the Iowa graph. Specifically, we randomly generated 100 ER graphs each with 2,492 vertices and with p chosen so that each graph had 5,205 edges in expectation. Servers were again chosen uniformly at random from among the existing vertices.

Figure 5.5(d) shows box and whisker plots for the result of client removal in each of the 100 random graphs. The qualitative results are very similar to the results presented for client removal in the Iowa random graph model. Variation between trials in both the upper and lower bounds is largely uniform except for a drop that appears in the upper bound after approximately 17% of clients have been removed. A similar drop appears in the upper bound corresponding to the Michigan graph after about 14% of clients have been removed.

Overall, however, bounds for the Michigan graph are quite different from the bounds found for the corresponding random graphs. First, the agreement between upper and lower bounds in the random graphs is far worse than the agreement in the Michigan graph. Second, while the agreement between bounds for the random graphs is too weak to draw any strong conclusions for small numbers of clients removed, we see that after 20% of clients are removed, the number of clients that remain connected to a server drops below 10%. In contrast, at least 20% of clients are still connected in the Michigan graph after removing any 20% of clients.

CHAPTER 6

VULNERABILITY OF DTNS

In this chapter, we evaluate the impact of node attacks on end-to-end throughout in DTNs and find that such attacks are relatively benign. We hypothesize that this robustness derives from DTN mobility and the high degree of packet replication utilized by DTN routing protocols. This renders DTNs much less fragile than MANETs. This is not to say that a DTN's absolute performance is better than a MANET's — rather that packet delivery rate degrades more gracefully in a DTN under attack. Let $\mathcal{D} = (N, K, C)$ be a DTN on node set N, node positions K, and connection events C. We seek to address the following problem.

PROBLEM 7: $MinThru(\mathcal{D}, M, \nu)$

- **Given:** DTN $\mathcal{D}(N, K, C)$, demand matrix M, and $\nu \in \mathbb{R}$
- **Find:** minimum $Thru(\mathcal{D} \ominus R)$, over all R where $R \subseteq N$ and $|R \cap N| = \nu$.

We are interested in identifying the most damaging attack in terms of throughput, which corresponds to finding the optimal solution to Problem $MinThru(\mathcal{D}, M, \nu)$. Accordingly, an *attacker* in this chapter is assumed to have knowledge of the optimal solution to Problem $MinThru(\mathcal{D}, M, \nu)$ and will attack DTN $\mathcal{D}$ by removing those

nodes that deliver this optimal solution. We prove in Chapter 8 that identifying the most damaging attack in a DTN is NP-hard even when modeling the DTN as a static graph and considering any of a broad class of metrics. This result suggests that we should not expect to be able to bound the impact of the optimal attack, but should instead be satisfied with a good approximation. We adopt an attack heuristic that seeks to eliminate the most temporally connected pairs of nodes in a DTN. With this strategy and using an effective DTN routing protocol, the DieselNet network achieves 50% of demand when 20% of the network nodes are removed and 30% of demand with 50% of nodes are removed. Haggle demonstrates similar but slightly stronger robustness. While our simulation results are limited to a particular protocol and attack heuristic, we believe many of our conclusions hold in general for the numerous DTN routing protocols that have been proposed.

6.1 Network Details

We make use of public traces from two experimental DTNs: the DieselNet [39] testbed and the Haggle Project [88], with 60 days of traces available from the former and 10 traces available from the latter. There are other DTNs that are being constructed or have been operational for a period of time [92, 120, 143]; unfortunately, none make their data publicly available. Our scope is restricted to pure DTNs.

The nodes in the networks we studied were only intermittently connected. Figures 6.1(a) & 6.1(b) show the average quantity of unique node contacts over all trace days. Only 75% of node pairs in DieselNet met consistently while 90% of node pairs met consistently in Haggle. Figure 6.1(c) shows the average daily bandwidth achieved

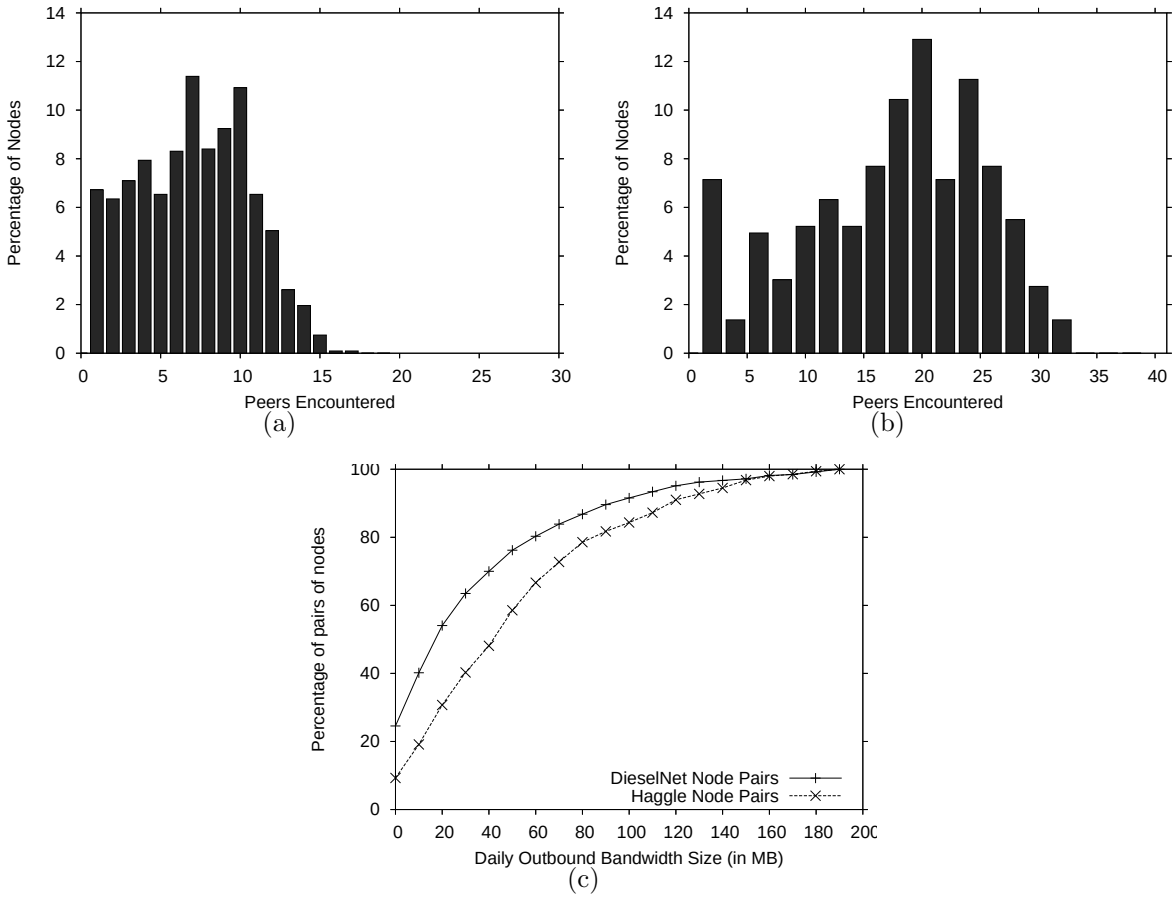


Figure 6.1. Plots (a) & (b) show the average percentage of unique peers contacted over all trace days in DieselNet and Haggle respectively. Plot (c) shows a CDF of bandwidth between node pairs in both DieselNet and Haggle.

for all pairs of nodes. In DieselNet 80% of node pairs exchange at least 60 MBs of data each day, while in Haggle, 80% of nodes exchange at least 90 MB each day. We believe that Haggle enjoys higher average aggregate throughput between pairs of nodes because the network is roughly twice as large as the DieselNet DTN.

6.2 Routing Strategy

For routing we used the RAPID routing protocol [19]. RAPID is a history-based routing protocol that maintains information at each node regarding past contact events with other nodes. When a pair of nodes establish contact they combine meta information concerning past contact events. Each half of the contact pair then prioritizes its own packets according to how likely it is that the other node will be able to deliver those packets according to a certain service metric.

6.2.1 Graph Model

A DTN is dynamic both in terms of connectivity and payload. Therefore, the ideal model would also be capable of capturing these aspects of a network's dynamics. However, as we show formally in Chapter 8, even studying attack vulnerability in much simpler models is NP-hard. In this chapter we resort to a graph representation of pair-wise connectivity in a DTN for our analysis. This problem is still NP-hard, but it proves easier to analyze. We use this graph representation to evaluate a greedy attack heuristic. For DTN $\mathcal{D} = (N, K, C)$ define the *flat graph* corresponding to $\mathcal{D}$, as $G = (V, E)$. Graph G is formed in the following manner: each node $u \in N$ is assigned a unique vertex $\mathcal{V}(u) \in V$. For any two nodes $u, v \in N$: if $\exists b, t$ such that

$(u, v, b, t) \in C$, then undirected edge $(\mathcal{V}(u), \mathcal{V}(v))$ is placed in E. In lieu of solving Problem *MinThru* directly, we instead look for the set of vertices whose removal in the flat graph most lowers metric $Pairs(G)$, or the number of connected paris of vertices in that graph. This makes sense because connectivity in the flat graph is a distillation of connectivity in $\mathcal{D}$ and rich connectivity is critical to maximizing throughput. More formally we analyze,

PROBLEM 8: $MinPairs(G, \nu)$

- **Given:** Graph $G = (V, E)$ and $\nu \in \mathbb{N}$.
- **Find:** minimum $Pairs(G \ominus R)$ over all R, such that $|R| \leq \nu$.

In Chapter 8 we show Problem $MinPairs(G, \nu)$ is NP-hard. We also show that any similar problem on a different metric is NP-hard provided that the metric achieves a single global minimum when G contains no edges. This suggests that measuring a graph's susceptibility to attack is difficult given most useful metrics. This does not preclude the possibility of finding a good approximation, but we show experimentally that the greedy attack is very close to the optimal attack for small ν, or the number of attacked nodes.

6.3 Experimental Findings

A brute force attack will always deliver the set of k vertices that most lower $Pairs(G)$, but the complexity of this approach grows exponentially with k. Fortunately, the greedy attack appears nearly as effective. In Figures 6.2(a) and 6.3(a) we show the median impact of the Brute and Greedy attacks on the flat graphs corresponding

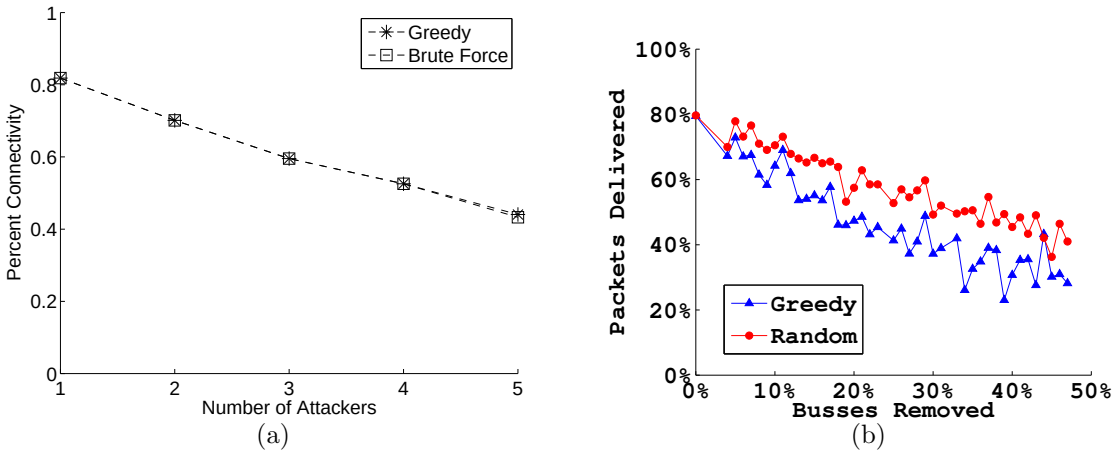


Figure 6.2. Robustness of the DieselNet DTN as measured in the (a) flat graph by comparing brute force node removal and greedy (b) trace driven simulation for both greedy and random node removal.

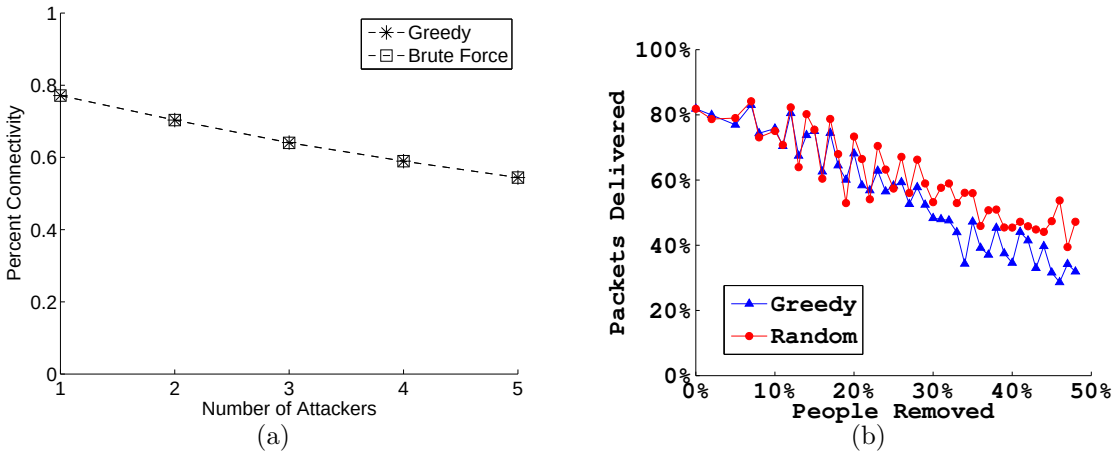


Figure 6.3. Robustness of the Haggle DTN as measured in the (a) flat graph by comparing brute force node removal and greedy (b) trace driven simulation for both greedy and random node removal.

to seven traces each for DieselNet and Haggle. The figures indicate that the Brute and Greedy attacks nearly coincide for both types of traces up to $k = 5$. Hence forth, we operate under the assumption that the greedy attack will continue to model the brute force attack well for higher values of k. We next apply the greedy attack to nodes in both the DieselNet and Haggle traces by choosing to remove those nodes with the greatest number of temporal connections.

Our analysis involved actual greedy removal of the most frequently contacted nodes (buses or people) in a given trace. We subsequently tested the packet delivery rate for each trace using a set of randomly generated packets. For DieselNet, we generated 100 packets at the beginning of every 6 hour interval throughout the day. For each packet, source and destination buses were selected uniformly at random from the set of buses still on the road at the time. Each packet was also given a TTL of 6 hours. For Haggle, trace duration varied widely so we divided each trace into 5 epochs each of equal duration. At the beginning of each epoch, we generated 100 packets with source and destination hosts selected uniformly at random from the people still present in the trace. The TTL for these packets was set to the previously determined epoch length.

In general, our experimental results indicate that greedy removal is relatively benign in terms of packet delivery rate under the RAPID protocol, but it does severely hinder packet delivery as the number of nodes removed approaches 50%. Figure 6.2(b) shows the effect on packet delivery rate averaged over all traces with random and greedy node selection in DieselNet. Figure 6.3(b) shows the results of the same experiment when conducted on the Haggle traces. For each plot, we decreased the

reported fraction of delivered packets proportional to the fraction of nodes removed since any packet with a source or destination node that has been removed will never be delivered. The figures show that when no nodes have been removed from the network, the number of delivered packets is slightly higher in Haggle than in DieselNet. We believe, from the results in Figures 6.1(a) and 6.1(b), this is due to a better "mixing" of participants at Infocom compared to a scheduled bus network. Overall, the Haggle results agree with the DieselNet results: DTNs are robust to attack in terms of the degradation to packet delivery rate. In particular, even when nearly 50% of the nodes are removed from either network, the packet delivery rate remains close to 30% in DieselNet and 40% in Haggle.

CHAPTER 7

GREEDY DTN AUGMENTATION

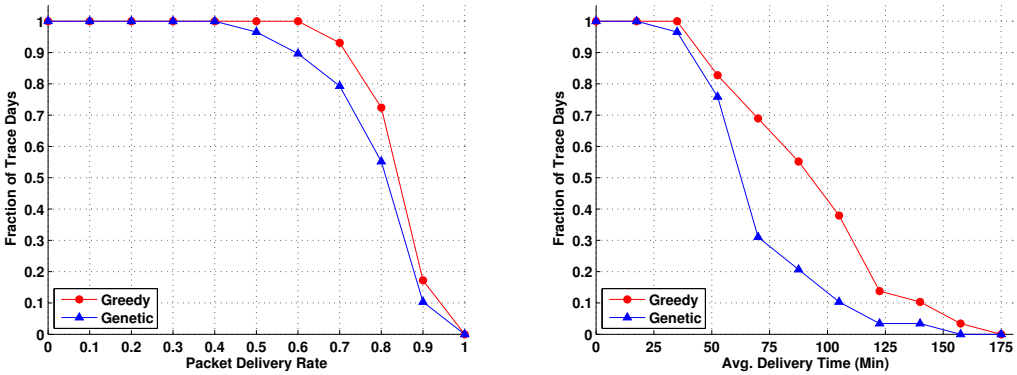
In the last chapter, we showed that DTNs are relatively robust to attack in terms of network throughput. However, even in an unperturbed DTN, bandwidth is scarce and delay is high (often on the order of minutes not seconds). In order to improve throughput and decrease delay, we consider DTN augmentation in this chapter. In contrast to static networks, DTNs are easily augmented because of the mobility of existing nodes. By placing a stationary relay in a geographic location that is often visited by mobile nodes, one can effectively add a new node and temporal links to the network. Relays hold packets left by one mobile node that are later picked up by another. Because relays need not be mobile or contiguous, they are much cheaper to deploy and can cover large areas. When solar-powered, they are also independent of the power grid [21]. Most previous work on relays for supporting mobile networks has focused on the design of the relay, including energy efficiency [21], asymptotic routing performance for large numbers of boxes [89], and performance comparison against other infrastructure [23]. In this Chapter we focus on finding the relay placement that offers the lowest delay for a given demand. This problem is very hard because even evaluating the quality of a placement requires knowledge of the perfect routing strategy. We fix the routing protocol and formulate a new problem statement relative

to this protocol whose solution is labeled *D-optimal*. We then introduce a dynamic program that provides a placement that bounds the D-optimal delay from below. This algorithm can be used to find good placements in arbitrary DTNs, but we use it to formally evaluate a greedy placement strategy in DieselNet. Specifically, we show that a greedy placement strategy will come within 40% of D-optimality 40% of the time.

7.1 Greedy Placement Compared to Prior Work

We begin with an illustration of the efficacy of a greedy relay placement in DieselNet. Lochert et al. [112] introduced a genetic programming algorithm for relay placement that seeks to minimize end-to-end delay in simulation. Their approach was to produce an initial population of bit-vectors with index i of any given vector corresponding to the presence or absence of a relay at location i. We reproduced this experiment with trace driven simulation for 50 relays in the DieselNet DTN running the RAPID protocol (see Section 7.4.1) and compared the results to a greedy placement according to frequency of location visitation.

Figure 7.1 shows the result of our experiment. Each simulation generated just 100 packets in order to keep run times low. We feel that this is a reasonable measure because we are most interested in delay that arises from the absence of a route from source to destination rather than from heavy network congestion. For the genetic program we began with a population of 10 bit-vectors and evolved for 5 rounds. These results were not significantly different than the results for a population of 3 vectors evolved for 3 generations. The greedy placement chose those 50 locations



(a) (b)

Figure 7.1. Comparison of genetic programming and greedy placement strategies in DieselNet in terms of (a) packet delivery rate and (b) packet delivery delay. The red curve with circles represents the results from our greedy placement, while the blue curve with triangles represents the results from a placement using genetic programming.

most frequently visited by buses. In all simulations, the TTL for every packet was set to 6 hours. The results reported are for a period of 29 days of traces in DieselNet. It's not possible to fix either delivery rate or delay with either placement strategy, so it's difficult to derive a direct comparison. On one hand, Figure 7.1(a) shows the greedy strategy consistently outperforming the genetic programming strategy in terms of the fraction of delivered packets. On the other hand, Figure 7.1(b) shows that the genetic programming approach always delivers packets with less delay. Overall, however, the results are close for both strategies even though the greedy solution requires no simulation while the genetic programming solution requires many tens of simulations. Our goal in this Chapter is show that a greedy placement strategy, as well as others, is nearly D-optimal in DieselNet.

7.2 Problem Description

Recall from Chapter 1 that a DTN is defined by $\mathcal{D} = (N, S, C)$, which is composed of a node set N, node positions K, and connection events C. Our focus is on the problem of relay placement in DTNs: *given x relays, where should they be geographically placed to most improve the performance of a particular DTN?* We call this problem the *Relay Placement Problem*. In Chapter 1 we formally defined this problem by the following.

PROBLEM 9: $Placement(\mathcal{D}, L, M, m, P^*)$

- **Given:** DTN $\mathcal{D} = (N, K, C)$, a set of potential relay locations L, a demand matrix M, a number of relays $m \in \mathbb{N}$, and a minimum throughput P^*.
- **Find:** minimum $Delay(\mathcal{D} \oplus R, M, P^*)$, over all supplementary sets of stationary nodes $R \subseteq L$, $|R| \leq m$.

7.2.1 Decoupling Routing from Placement

Problem *Placement* is complicated by its entanglement with optimal routing. Zhao et al [147] have shown that this joint problem of optimizing routing and placement is NP-Hard, even if the schedule of meetings between mobile nodes and relays is known *a priori*. Balasubramanian et al. [19] have shown that optimizing routing alone is NP-Hard and that the problem even resists close approximation. Indeed, according to these results, it is NP-hard just to evaluate the quality of a given solution to Problem *Placement*, which takes the Relay Placement Problem out of NP provided that $P \neq NP$. This entanglement motivates the development of a new problem, the

Decoupled Placement Problem, which is optimal with respect to some 1-hop demand matrix M^1.

PROBLEM 10: *D-Placement*$(\mathcal{D}, L, M^1, m, P^*)$

- **Given:** DTN $\mathcal{D} = (N, K, C)$, a set of potential relay locations L, a one-hop demand matrix M^1, a number of relays $m \in \mathbb{N}$, and a minimum throughput P^*.
- **Find:** A set of locations $L' \subseteq L, |L'| = m$, minimizing delay and subject to the throughput being greater than or equal to P^*.

We label the optimal solution *D-optimal*. The associated delay and placement are called D-optimal as well. Problem *D-Placement* concerns one-hop demand as opposed to the end-to-end demand that is satisfied in Problem *Placement*. In order to clearly differentiate the two we define the *relay flow* to be the total amount of one-hop demand passing through placed relays. Accordingly, we refer to the throughput parameter P^* from Problem *D-placement* as the *target relay flow*.

Chapter 8 shows that even finding the D-optimal solution is NP-hard. So instead we solve a relaxed version of the problem: *R-Placement*. Specifically, we relax Problem *D-Placement* by assuming that *i)* if demand is allocated to one relay, there will be enough additional demand to allocate demand to other relays, *ii)* no two buses appear at the same relay location at the same time. We then present a dynamic program that solves Relaxation *R-Placement* to within arbitrary accuracy. Because this algorithm provides a lower bound for the D-optimal solution to the relay placement problem, we refer to it as the *bounding solution* and the placement it delivers as the *bounding placement*.

7.2.2 Relaxing *D-Placement*

Chapter 8 proves that *D-Placement* is itself NP-hard. Algorithmically, the problem is similar to several canonical problems. The placement aspect is reminiscent of the Knapsack, Facility Location, or Vertex Cover Problems. Indeed, we reduce a variant of Vertex Cover to the Decoupled Placement Problem in Chapter 8. On the other hand, our desire to minimize delay recalls the Job Shop Scheduling Problem. At this point, it's not clear whether *D-Placement* can be effectively approximated. Rather than attempt to approximate the Decoupled Placement Problem directly, we elect to relax the problem by making the following assumptions.

1. **Ample Demand**. If two relays are each individually capable of satisfying a unit of demand, then there is enough demand to accommodate both.

2. **No Co-location**. buses arrive at relays at different times, so that each bus has full access to the bandwidth available through that relay.

Let E denote the *egress matrix* derived from some DTN $\mathcal{D}$ and one-hop demand matrix M^1. For every location x in some finite set $\mathcal{X}$ and time interval t, matrix entry $E(x,t)$ is the greatest aggregate flow achievable through x during interval t. Given the assumptions above, we can construct matrix E by considering each location independently from the others. Our approach is to *front load* location x by assigning flow to the earliest time intervals possible. This ensures that we have maximum throughput for minimum delay. Algorithm FRONT_LOAD (see Section 7.6) performs this calculation for each location and time in order to form matrix E. The idea is to update the *store* of flow at each location, and then to pass that flow as quickly as

possible (since this will ensure the lowest delay). FRONT_LOAD takes the following arguments as input: $M^1(b_1, b_2; t)$ is the maximum flow between buses b_1 and b_2 during time interval t, $A(i, t)$ is the set of buses adjacent to relay i during interval t, and $B(x, t)$ is the number of bytes of capacity at location x during interval t.

For any location x and time interval t, $E(x, t)$ provides an upper bound on the amount of one-hop demand that can be satisfied by location x during time interval t. It follows that the solution to the Decoupled Placement Problem can be bounded from below by the solution to the following problem.

PROBLEM 11: *R-Placement*(E, m, P^*)

- **Given:** egress matrix $E(x, t)$, number of relays $m \in \mathbb{N}$, and minimum throughput P^*.
- **Find:** m row indices of $E(x, t)$ and m termination times, which collectively achieve P^* and minimize Y.

7.3 Solving *R-Placement*

We approximate the solution to Problem *R-Placement*(E, m, P^*) with dynamic program ALLOCATE (see Section 7.6), which seeks to find the set of optimal locations and the optimal allocation of flow through those locations. It is essentially a routing scheme for one-hop demand. We now provide the details of constructing Algorithm ALLOCATE and bounding its error with respect to the optimal solution to Problem *R-Placement*. To begin, assume that time progresses discretely in steps $t = 1, 2 \ldots$ where it is understood that each step corresponds to some fixed interval of time I

independent of t. Define discrete cumulative flow in units of U bytes as $f_1, f_2 \ldots$ with $f_i = iU$. Let $Y(x, f_i)$ give the minimum total delay for relay location x and flow f_i. We have,

$$Y(x, f_i) = \sum_{t \leq t^*} tE(x, t),$$

where

$$t^* = \arg\min_{t'} \sum_{t \leq t'} E(x, t) \geq f_i.$$

Algorithm ALLOCATE constructs a table of total delay values, δ. Each entry, $\delta(m, L, f_i)$, of the table gives the smallest total delay achievable by sending as many as f_i units of flow in total through at most m relays chosen from the set L. In contrast to table δ, let $\Delta(k, L, f)$ be a *continuous* function of a continuous flow variable f, which gives the smallest total delay achievable by sending as many as f bytes of flow through at most k relays chosen from the set L. The largest gap between the total delay reported for $\delta(m, L, f_i)$ and that reported by $\Delta(m, L, f)$ with $f_{i-1} \leq f \leq f_i$ represents the approximation error that Algorithm ALLOCATE introduces in solving Problem *R-Placement*

In developing Algorithm ALLOCATE, we wish to devise an algorithm for constructing the table δ. Our idea is to construct δ iteratively beginning with $|L| = n$ tables, $\{\delta_1 \ldots \delta_n\}$, each with one row and exactly one relay location λ, where entry $\delta(1, \{\lambda\}, f_i) = Y(\lambda, f_i)$ or the total delay through location λ for flow of f_i. Each round of iterations pairs off existing tables. If there is an odd table out, then it simply passes to the next round. Combining two tables is essentially merging the information about two disjoint sets of locations. Suppose that we have tables δ_i and δ_j and we wish to combine them to form table δ_k. We will iterate over all

possible distributions of relays and flows between the two tables. For the step where we combine $\delta_i(m_i, L_i, f_i)$ and $\delta_j(m_j, L_j, f_j)$ the value of $\delta_k(m_i + m_j, L_i \cup L_j, f_i + f_j)$ should be set to $\delta_i(m_i, L_i, f_i) + \delta_j(m_j, L_j, f_j)$ iff that value exceeds its current value. In this manner each entry of the new table can be filled in. Iteration ceases when only one (the solution) table remains.

THEOREM 3: For any choice of k and L, $\delta(k, L, f_i) \leq TU + \Delta(k, L, f)$ when $f_i \leq f$, T is the total number of time intervals, and U is granularity of flow values.

PROOF:

For a given f_i, the functions $\Delta(k, L, f_i)$ and $\delta(k, L, f_i)$ can be broken down as

$$\delta(k, L, f_i) = \min_{\sum F_j = f_i} \sum_j \delta(1, L, F_j)$$

and

$$\Delta(k, L, f_i) = \min_{\sum x_j = f_i} \sum_j \Delta(1, L, x_j)$$

where F_j is the discrete flow and x_j the continuous flow assigned to relay j. Because the function δ can assume only discrete values of flow, it's possible that the true minimum total delay is less than what is rendered. However, since the function Δ is non-increasing with x and $\delta(1, L, f_i)$ is accurate at discrete flows, f_i, we have

$$
\begin{aligned}
\Delta(k, L, f_{i-1}) &= \min_{\sum x_j = f_{i-1}} \sum_j \Delta(1, L, x_j) \\
&= \min_{\sum F_j = f_{i-1}} \sum_j \delta(1, L, F_j) \\
&\leq \min_{\sum F_j = f_i} \sum_j \delta(1, L, F_j) \\
&= \delta(k, L, f_i) \\
&= \min_{\sum x_j = f_i} \sum_j \Delta(1, L, x_j) \\
&= \Delta(k, L, f_i),
\end{aligned}
\tag{7.1}
$$

or $\Delta(k, L, f_{i-1}) \leq \delta(k, L, f_i) \leq \Delta(k, L, f_i)$. Going from $\Delta(k, L, f_{i-1})$ to $\Delta(k, L, f_i)$, we will pass no more than one additional unit (U bytes) of flow through all boxes. This additional flow cannot increase the total delay by more than a factor of T. Hence,

$$\Delta(k, L, f_i) - \Delta(k, L, f_{i-1}) \leq TU.$$

□

The preceding analysis shows that we can make this gap as small as we like by continuing to refine the set F. On the other hand, this makes ALLOCATE less efficient computationally.

7.4 Experimental Goals and Procedure

Our hypothesis is that choosing relay locations in greedy fashion, according to the frequency that they are visited, will be a good placement strategy. To validate this hypothesis, we evaluated a total of 58 days of bus data from DieselNet [107].

These traces include both bus-to-bus contact events and single-bus GPS readings throughout each day. This analysis requires multiple steps, which are outlined in the procedure below.

PROCEDURE 4:

1. **End-to-end Demand** Create a single random demand matrix M between each bus, which passes a total of 2000 packets each 1K in size.

2. **One-hop Demand** For each trace day, run a simulation, with unlimited bandwidth and no relays that produces one-hop demand matrix M^1, and which attempts to meet demand M according to the contacts between buses recorded for that day (Figures 7.2(b and c)).

3. **Bounding Placement and Delay** For each trace day, a fixed quantity of relays, and one-hop demand M^1, find the minimum delay and bounding placement provided by Algorithm ALLOCATE (Figure 7.2(d)).

4. **Evaluate Performance** Rerun the simulator for each trace day and each placement strategy, but this time with bandwidth limited and with maximum delay set to the minimum average delay from bounding solution (via ALLOCATE).

With this procedure, we will compare the greedy solution to the optimal solution to *R-Placement*(E, m, P^*) by showing that a greedy strategy can deliver 40% of the target relay flow P^* with D-optimal delay on 40% of the trace days. Better results are achieved when constraints on maximum delay are relaxed.

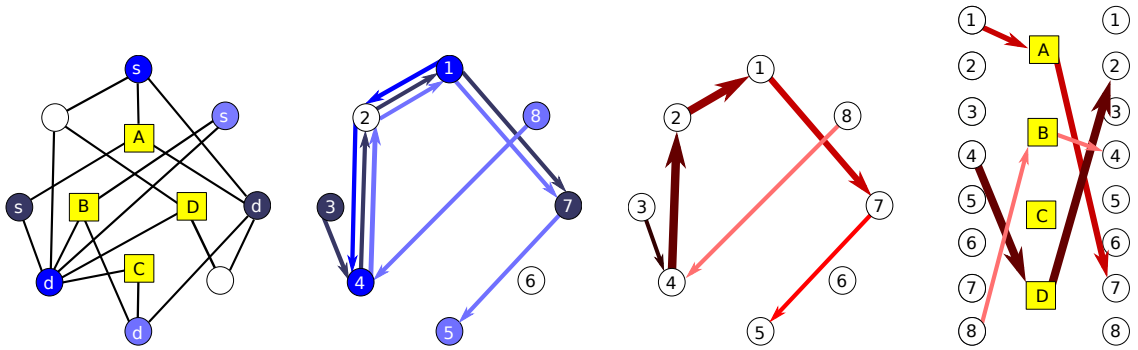


Figure 7.2. (a) DTN $\mathcal{D}$: at fixed time, with demand M for commodities indicated by color and labeled s for source and d for destination. (b) Stage 1: uncapacitated demand satisfaction without relays (c) M^1, one-hop demand, formed from uncapacitated solution. (d) Stage 2: relays partially accommodating demand M^1 while respecting capacity.

7.4.1 Simulation

We used the RAPID routing protocol [19] for all simulations. RAPID is a history-based routing protocol that maintains information at each node regarding past contact events with other nodes, and in this case, other relays. When a pair of nodes or relays establish contact they combine meta information concerning past contact events. Each half of the contact pair then prioritizes its own packets according to how likely it is that the other node will be able to deliver them according to a certain service metric. In our simulations we elected to have RAPID attempt to minimize average delay. For step 1 of Procedure 4, we injected 2000 packets, each 1K in size, into the network at the beginning of the day at source buses chosen uniformly and independently at random. The corresponding destination buses were also chosen uniformly and

independently at random. All mobility traces came from the DieselNet [107] bus network (see Section 1.3).

The wireless radios used in the DieselNet testbed have an approximately 140 meter range, so we created potential relay locations by tessellating the area of operation with 140 meter squares. Only those squares intercepted by at least two buses (indicated by registration from GPS units on-board each bus) were kept as potential relay locations. We were able to use real transmission values for connection events between buses, but naturally had no transfer data for GPS events. In order to estimate transmission values between buses and relays, we used the mean transfer value over all connection events for the given day.

7.5 Evaluation

Our evaluation proceeds according to Procedure 4, where each step of the procedure is followed for all 58 days of traces. After completing steps 1 and 2 we are left with the one-hop demand matrix M^1. For every trace day, we arbitrarily fix the target relay flow $P^* = 15MB$, which is to say that 15MB of demand is satisfied from matrix M^1. In step 3, Algorithm ALLOCATE identifies good candidates for relay placement, which we call the *bounding placement*. Figure 7.3 shows how average delay decreases with the quantity of relays allocated. The figure indicates that there is a significant reduction in average delay when moving from 10 to 20 relays. However, moving from 40 to 50 relays produces no discernible improvement. This suggests that for $P^* = 15$MB, 50 relays saturates the network. Hence forth, we fix the number of relays at 50. The bounding placement heavily favors regions where buses congregate,

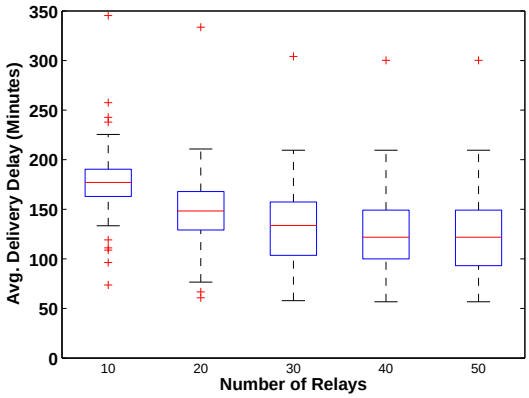
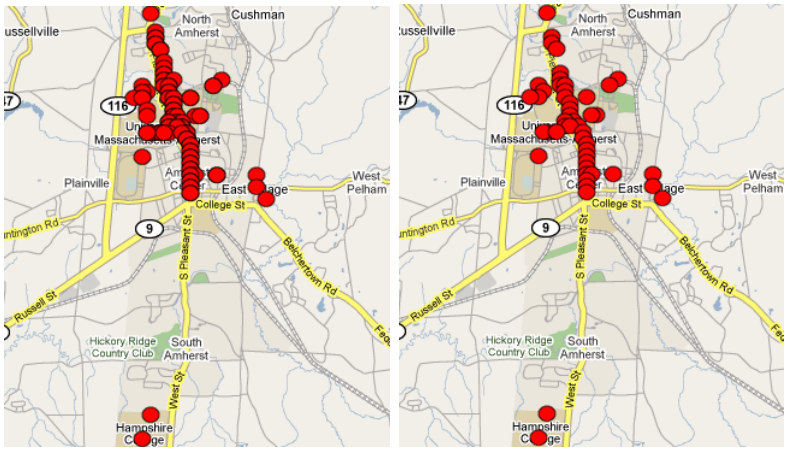


Figure 7.3. Average delay for 15MB throughput given various number of relays.

which supports our decision to analyze frequency-based placement. The next section demonstrates this similarity.

7.5.1 Visualizing Placement

Recall that the bounding placement is the placement returned by Algorithm ALLOCATE while the frequency placement is a greedy placement according to the frequency of bus visitation at a given location. The frequency and bounding placement strategies operate with vastly different sets of information and depth of analysis. While the frequency solution relies entirely on GPS data, the bounding solution additionally utilizes complete knowledge of one-hop demand. In light of these differences, the similarity of the two placements is striking. Figures 7.4(a) and 7.4(b) show the 50 most commonly chosen relay locations for bounding and frequency solutions over the first 30 days of traces. Comparing the placements to the bus routes

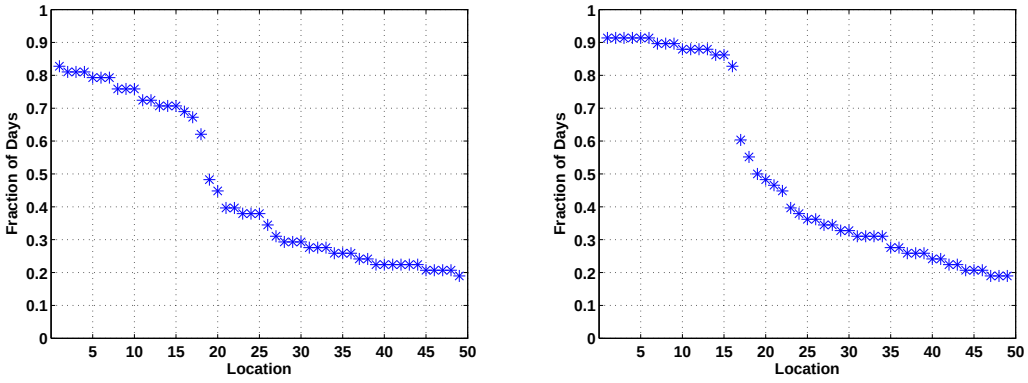


(a) (b)

Figure 7.4. The 50 most commonly chosen locations over the first 30 trace days for (a) bounding placement (b) frequency placement.

depicted in Figure 1.2, we see that the relays span a length of road running through Amherst Center and various adjacent side streets.

Figures 7.5(a) and 7.5(b) show the fraction of trace days that choose each of the 50 most popular locations for the bounding and frequency placements respectively. That is to say, the figures show the popularity of the 50 most commonly chosen locations across all trace days. There are a small number of locations (roughly 20 according to both strategies) that are chosen on the majority of days. These locations constitute hubs, whose existence provide support for the rationality of a frequency-based placement scheme. The relatively high frequency of the other 30 locations makes a case for broadening the scope of our frequency analysis. To test this idea, we create a new placement strategy that trains on the first 30 days by



(a) (b)

Figure 7.5. Redundancy of top 50 location choices for the first 30 days for (a) bounding placement (b) frequency placement.

simply observing the popularity of each location on each day, then tests on the latter 28 days. We call this the *trained placement*.

7.5.2 Frequency-based placement is nearly D-optimal in DieselNet

In this section we apply our bounding techniques to validate frequency-based placement strategies in trace driven simulation. Our ultimate goal is to show that these strategies perform well relative to the D-optimal solution. Recall that the relay flow is the total amount of one-hop demand passing through placed relays, and the target relay flow is equal to the value of parameter P^* in Problem *D-placement*. This section shows that the trained placement strategy achieves 40% of the target relay flow with D-optimal delay on 40% of the trace days.

The bounding solution provided by Algorithm ALLOCATE delivers a *bounding delay*, $\overline{D}$, which is the average delay associated with satisfying the target relay flow.

Because ALLOCATE is a solution to *R-Placement*, which is itself a relaxation of *D-Placement*, we can conclude that for the same target relay flow, the D-optimal delay is no less than $\overline{D}$. This makes the bounding delay an important parameter for testing all placement strategies; by restricting relay flow delay to $\overline{D}$ and measuring what fraction of target relay flow is actually delivered, we can compare each placement strategy to the D-optimal strategy. Our approach is to test each placement strategy in trace driven simulations as described in Section 7.4.1 where we additionally fix the maximum relay flow delay to the bounding delay. We test the following placement strategies.

- **Frequency:** relays placed at most frequently visited locations each day and relay flow delay limited to $\overline{D}$
- **Frequency**$\times$**2:** relays placed at most frequently visited locations each day and relay flow delay limited $2\overline{D}$
- **Trained:** relays placed at most frequently visited locations over first 30 days and relay flow delay limited $\overline{D}$

Figure 7.6 shows the fraction of target relay flow that each strategy was able to pass within the bounding delay (twice this delay for frequency$\times$2). The trained solution is consistently superior to the other two strategies. It delivers up to 40% of the target relay flow with D-optimal delay. The frequency solution was consistently worse, but frequency$\times$2 had the best worst-case performance with fully 90% of trace days achieving at least 15% of target relay flow with D-optimal delay.

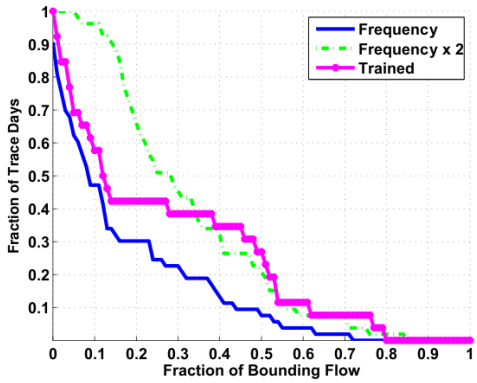
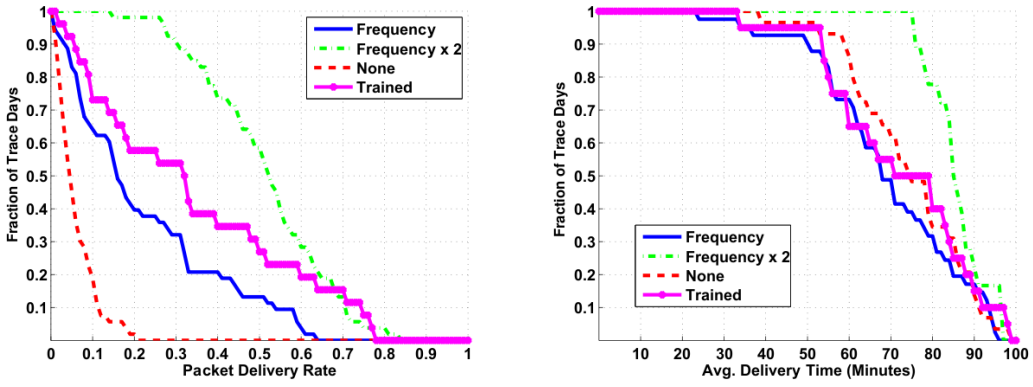


Figure 7.6. Aggregate relay flow by day where maximum average delay is limited to the bounding delay on that day and plotted as a CCDF over all days (28 for Trained, 58 for others).

7.5.3 Impact on End-to-End Performance

In the previous section we saw that frequency-based placements can be nearly D-optimal in DieselNet on a significant portion of the trace days. But measuring the proximity to the D-optimal solution can only show how well a strategy satisfies one-hop demand. In this section we apply our placement strategies to the full end-to-end routing problem.

Figures 7.7(a) and 7.7(b) show the packet delivery rate and average delay, respectively, for each of our placement strategies. For contrast, we added the results from the none placement, which also takes the bounding delay as its maximum delay, but uses no relays. Not surprisingly, none placement does not perform well, it has the lowest delivery rate and the second worst average delay. Improving over the results shown in Figure 7.6, frequency×2 performs the best in terms of delivery rate except that it has no days delivering more than 85% of their packets. Because of its



(a) (b)

Figure 7.7. End-to-end performance in Simulations Frequency, Frequency×2, None, and Trained for (a) packet delivery rate and (b) average delay.

construction, frequency×2 has far higher average delay than all the other strategies. The trained strategy consistently delivers more packets than frequency, which was also predicted by Figure 7.6. Figure 7.7(b) reveals that there is no penalty in terms of average delay for the trained placement.

7.6 Algorithms

Algorithm 3 FRONT_LOAD(M^1, B, A)

$\forall t_o, x,\ E(x, t_o) \leftarrow 0$ Initialize Egress for each out-time

$\forall t, b_i, b_o,\ d(t, b_i, b_o) \leftarrow M^1(b_i, b_o; t)$ Initialize Demand between input and output buses

for $x \in \{1, \ldots, L\}$ **do**

  $\forall b_i,\ I(b_i) \leftarrow 0$ Initialize Ingress for the given input bus

  for $t \in \{1, \ldots, T\}$ **do**

    $\forall b_i \in A(x, t),\ I(b_i) \leftarrow I(b_i) + B(x, t)$ Add bytes to each relay, for each adjacent bus at time t

    for $b_o \in A(x, t);\ t_i \in \{1, \ldots, t\};\ b_i \in A(x, t_i)$ **do**

      $\delta \leftarrow \max\{B(x, t), \min\{I(b_i), d(t_i, b_i, b_o)\}\}$ Pass as many bytes as Egress without exceeding B

      $E(x, t) \leftarrow E(x, t) + \delta$

      $I(b_i) \leftarrow I(b_i) - \delta$

      $d(t_i, b_i, b_o) \leftarrow d(t_i, b_i, b_o) - \delta$

    end for

  end for

end for

Algorithm 4 ALLOCATE(m, L, F)

```
TABLES ← ∅
for λ ∈ L do
  for f_i ∈ F do
    δ(1, {λ}, f_i) ← Y(λ, f_i)
  end for
  TABLES ← TABLES ∪ {δ(1, {λ}, :)}
end for
while |TABLES| > 1 do
  PAIRS ← MATCH(TABLES)
  while |PAIRS| > 0 do
    PAIR ← POP(PAIRS)
    δ_0(m_0, L_0, :) ← PAIR[0]; δ_1(m_1, L_1, :) ← PAIR[1]
    for m* ∈ {1, ..., m_0 + m_1} do
      δ(m*, L_0 ∪ L_1) ← ∞
    end for

    for f_j ∈ {f_1, ..., f_i} do
      for f_k ∈ {f_i − f_j, ..., f_i} do
        for m_j ∈ {0, ..., m_0} do
          m* ← min{m' − m_j, m_1}
          for m_k ∈ {0, ..., m*} do
            m_j ← max{1, m_j + m_k}
            δ_L ← δ(m_1, L1, f_j); δ_R ← δ(m_2, L2, f_k)
            δ(m', L, f_i) ← min{δ(m', L, f_i), δ_L + δ_R}
          end for
        end for
      end for
    end for
    TABLES ← {δ(:, L_0 ∪ L_1, :)}
  end while
end while
return TABLES[0]
```

CHAPTER 8

COMPUTATIONAL COMPLEXITY

8.1 Problem *MinConn*

Wolcowicz and Zhao [144] studied a problem fundamentally similar to *MinConn*: Graph Partitioning. They sought to discover the fewest number of edges whose removal could partition a graph into disconnected blocks with sizes $n_1, \ldots, n_b$. Their idea was to first assign an indicator vector to each block $x^1, \ldots, x^b$, where $x^i_j = 1$ iff vertex j is assigned to block i. The ideal optimization algorithm would assign the x^i so that $|x^i| = n_i$ and the number of edges between vertices in different blocks was minimized. Counting the number of edges between blocks can be accomplished follows: label edges passing between blocks as *external*. These edges compose the *edge cut*; its cardinality is called the *edge cut size*. It can be verified that $L \bullet x^i(x^i)^T$ gives the number of external edges incident to vertices in block i [28]. It follows that $\frac{1}{2}\sum_i L \bullet x^i(x^i)^T$ gives the edge cut size. A simpler version of this problem can be realized for both edge and vertex partitions on two blocks as follows.

PROBLEM 12: $EdgePartition(H, n_1, n_2)$

- **Given:** Graph $H = (V, E)$ and desired blocks sizes n_1 and n_2, with $n_1 + n_2 = |V|$.

- **Find:** minimum quantity of edges whose removal creates two blocks with sizes n_1 and n_2.

PROBLEM 13: $VertexPartition(H, n_1, n_2)$

- **Given:** Graph $H = (V, E)$ and desired blocks sizes n_1 and n_2, with $n_1 + n_2 = |V|$.
- **Find:** minimum quantity of vertices whose removal creates two blocks with sizes n_1 and n_2.

Bui and Jones [117] showed that weaker versions of these problems are NP-hard. The version they analyzed looked for the smallest cut that separates a graph into two blocks, each no smaller than $\alpha|V|$, with $\alpha \leq \frac{1}{2}$. To show that problems $MinConn(\Theta, \nu, 0)$ and $MinConn(\Theta, 0, \epsilon)$ are also NP-hard, it will suffice to reduce them to the latter.

THEOREM 4: $EdgeSeparator(H, n_1, n_2) \leq MinConn(G, 0, \epsilon)$.

PROOF: Consider the decision problem $DEdgeSeparator(H, n_1, n_2, k)$, which returns 1 when there exists an edge separator of size k or less that achieves balance (n_1, n_2). Assume without loss of generality that $n_1 \leq n_2$. We will show that any instance of *DEdgeSeparator* can be efficiently transformed into an instance of the decision problem $DMinConn(G, 0, \epsilon, c)$. $DMinConn(G, 0, \epsilon, c)$ returns 1 when at least c clients are disconnected from all servers after removing any ϵ edges and returns 0 otherwise.

For any instance $(H, n_1, n_2, k) \in$ *DEdgeSeparator*, with $H = (V_H, E_H)$, fashion a corresponding instance of *DMinConn*: $(G, 0, \epsilon, c)$ as follows. For $G = (S_G \cup C_G, E_G)$, let $C_G = V_H$ and initialize $E_G = E_H$. Let $S_G = \{s\}$ be a single vertex, and add to E_G one edge between s and each client in C_G. Finally, set $c = n_1$ and $\epsilon = n_1 + k$. Ignore, for the moment, s and all adjacent edges, call this graph G^*. What remains in G^* is a graph isomorphic to H. Therefore, any optimal cut corresponding to a solution to *DEdgeSeparator* in H will also exist in G^*. That is, the smallest cut in H that creates blocks of size n_1 and n_2 in H will also create blocks of size n_1 and n_2 in G^*. So

$$(H, n_1, n_2, k) \in DEdgeSeparator \Leftrightarrow (G^*, 0, k, n_1) \in DMinConn.$$

But any block of size n_1 in G must additionally cut the n_1 edges adjacent to s before being completely disconnected from the other block. The result follows by combining the size, k, of the block cut in G^* with the size, n_1, of this server cut so that $\epsilon = n_1 + k$.

□

COROLLARY 4: $VertexSeparator(H, n_1, n_2) \leq MinConn(G, \nu, 0)$.

PROOF: The proof of Theorem 4 also applies here if we change edge cuts to vertex cuts and replace each undirected edge of the form $(s, c_i) \in E_G$ with the path $(s, x_i) \sim (x_i, c_i)$ where each x_i is an additional client vertex.

□

8.2 Problem *MinPairs*

Problem *MinPairs*(G, e) is *nearly* the dual of the undirected and unweighted version of the β-edge-disruptor problem. It seeks a subset of edges that disconnect a fraction of at least (1-β) strongly connected pairs of vertices. The authors showed that this problem is NP-hard by reducing to it the Balanced Cut (or Edge-Separator) Problem.

PROBLEM 14: β-*Disruptor*(H, β)

- **Given:** Graph $H = (V, E)$ and desired fraction of *disconnected* pairs $\beta \in [0, 1]$. (excluding reflexive pairs of the form (v, v) with $v \in V$).
- **Find:** minimum quantity of edges whose removal leaves at least $\beta|V|$ disconnected pairs of vertices.

THEOREM 5: β-*Disruptor*$(H, \beta) \leq$ *MinPairs*(G, ϵ)

PROOF: Let (H, n), with $H = (V, E)$, be any instance of the β-*Disruptor* problem. We solve this problem with at most $|E|$ invocations of Problem *MinPairs*(G, ϵ) as follows. Let $G = H$ and for each $\epsilon' \in \{1, 2 \ldots |E|\}$ let $x =$ *MinPairs*(G, ϵ'). If $x - \sqrt{x} \geq \beta|V|$ return ϵ.

□

The proof can be easily extended to show that *MinPairs*(G, ν) is also NP-hard. But we provide a different proof that generalizes the problem to address a wide variety of metrics. Recall the Vertex Cover problem,

PROBLEM 15: $VC(G, k)$

- Given: Graph $G = (V, E)$ and integer $k \leq |V|$.
- Find:
  - *1* if there exists a set $S \subseteq V, |S| \leq k$, such that every edge in $e \in E$ has at least one endpoint in S.
  - *0* otherwise.

The Vertex Cover problem has been shown to be NP-complete [74]. We next introduce a bit of terminology and then a generalized graph vulnerability decision problem.

DEFINITION 1: The set of all graphs is given by $\mathcal{G}$. A function on $\mathcal{G}$ is any function of the form $f : \mathcal{G} \to \mathbb{R}$. We call f *well defined* if it achieves its minimum only when G contains no edges.

PROBLEM 16: $DMinMetric(G, \nu, f, c)$

- **Given:** Graph $G = (V, E)$, $\nu \in \mathbb{N}$, and $f : \mathcal{G} \to \mathbb{R}$.
- **Find:**
  - *1* if there exists a set $S \subseteq V, |S| \leq \nu$, such that $f(G^S) \leq c$.
  - *0* otherwise.

THEOREM 6: Problem MinMetric is NP-hard whenever $f(G)$ is well defined and computable in time polynomial in G and k.

PROOF: Let graph $G = (V, E)$ and integer k constitute an instance of the Vertex Cover Problem. For a given f, construct a corresponding instance of the *DMinMetric* problem by leaving G unchanged and letting $\nu = k$ and $c = f((V, \emptyset))$. Suppose that $\mathrm{VC}(G, k) = 1$ for some set $S \subseteq V, |S| \leq k$. This implies that there are no edges in G^S. Hence, the same choice of S will render $f(G^S) = c$. Therefore, $DMinMetric(G, \nu, f, c) = 1$.

Conversely, suppose that $\mathrm{VC}(G, k) = 0$. It must be the case that $DMinMetric(G, \nu, f, c) = 0$ as well because if it did not then there would be some set $S \subseteq V, |S| \leq \nu$ such that $f(G^S) \leq c$. But since f is well defined we know that G^S contains no edges. This set S could thus be used to form a vertex cover for G of size less than ν. Hence, $\mathrm{VC}(G, \nu) = 1$. It follows by contradiction that $DMinMetric(G, \nu, f, c) = 0$.

□

8.3 Problem *Placement*

Our final discussion attempts to characterize the computational complexity of Problem *D-Placement*. Recall the Maximum Coverage Problem

PROBLEM 17: Maximum Coverage (MC)

- **Given:** Collection of sets $\mathcal{U}$ and integer $k \leq |V|$.
- **Find:** maximum $|\bigcup_{S \in \chi} S|$ over all $\chi \subseteq \mathcal{U}, |\chi| = k$.

THEOREM 7: Problem *D-Placement*$(\mathcal{D}, L, M^1, m, P^*)$ is NP-hard.

PROOF: We reduce the Problem MC, to Problem *D-Placement*$(\mathcal{D}, L, M^1, m, P^*)$. Construct instance $I = (\mathcal{D}, L, M^1, m, P^*)$ of Π^D as follows. Assume that there is a single time step in $\mathcal{D}$ and let $m = k$. For each $i \in \mathcal{U}$, create two nodes n_{i1} and n_{i2} in $\mathcal{D}$ and assign a single unit of one-hop demand between them. Map each set $S \in \chi$ to a relay location $\lambda \in L$, and let both n_{i1} and n_{i2} be adjacent to λ iff $i \in S$. Consider the set of optimal locations K and two-hop routing $\mathcal{R}^2$ from *D-Placement*. Exactly u items in $\mathcal{U}$ can be covered by k sets in χ iff there exists finite average delay $P^* = u$.

□

BIBLIOGRAPHY

[1] `www.caida.org`.

[2] http://sedac.ciesin.columbia.edu.

[3] http://www-cta.ornl.gov/transnet/highways.html.

[4] http://www.socrata.com.

[5] www.haciendacdc.org.

[6] www.transco.ph/gridmap/2006

[7] I.F. Akyildiz, W. Su, Sankarasubramaniam.Y., and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 2001.

[8] R. Albert, H. Jeong, and A. Barabasi. Error and attack tolerance of complex networks. *Nature*, 406:378–382, 2000.

[9] David Alderson, Lun Li, Walter Willinger, and John C. Doyle. Understanding internet topology: principles, models, and validation. *IEEE/ACM Trans. Netw.*, 13(6):1205–1218, 2005.

[10] Noga Alon. On the edge-expansion of graphs. *Combinatorics Probability and Computing 11*, 1993.

[11] Noga Alon. *Spectral techniques in graph algorithms*. Springer-Verlag, 1998.

[12] M. Amin. Security challenges for the electricity infrastructure. *Computer*, 2002.

[13] M. Amin. North america's electricity infrastructure: are we ready for more perfect storms? *IEEE Symposium on Security and Privacy*, 2003.

[14] William A. Arbaugh, Narendar Shankar, and Y.C. Justin Wan. Your 802.11 wireless network has no clothes. *IEEE Wireless Communications*, 2002.

[15] Susanne Aref and Barry Miller. Analysis on 2004 data Submitted into Damage Information and Reporting Tool (DIRT). Technical report, Common Ground Alliance, December 2005.

[16] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Proceedings of the ACM Symposium on Theory of computing*, 2004.

[17] Ashwin Arulselvan, Clayton W. Commander, Lily Elefteriadou, and Panos M. Pardalos. Detecting critical nodes in sparse graphs. *Computers and Operations Research*, 2008.

[18] Aruna Balasubramanian, Brian Neil Levine, and Arun Venkataramani. DTN Routing as a Resource Allocation Problem. In *Proc. ACM SIGCOMM*, August 2007.

[19] Aruna Balasubramanian, Brian Neil Levine, and Arun Venkataramani. Dtn routing as a resource allocation problem. *ACM Special Interest Group on Data Communication*, 2007.

[20] Aruna Balasubramanian, Ratul Mahajan, Arun Venkataramani, Brian Neil Levine, and John Zahorjan. Interactive wifi connectivity for moving vehicles. *ACM Special Interest Group on Data Communication*.

[21] Nilanjan Banerjee, Mark D. Corner, and Brian Neil Levine. An Energy-Efficient Architecture for DTN Throwboxes. In *Proc. IEEE Infocom*, May 2007.

[22] Nilanjan Banerjee, Mark D. Corner, Don Towsley, and Brian N. Levine. Relays, base stations, and meshes: Enhancing mobile networks with infrastructure. *International Conference on Mobile Computing and Networking*, 2008.

[23] Nilanjan Banerjee, Mark D. Corner, Don Towsley, and Brian Neil Levine. Relays, Base Stations, and Meshes: Enhancing Mobile Networks with Infrastructure. *Proceedings of ACM MobiCom*, 2008.

[24] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286:590–512, 1999.

[25] Albert-laszlo Barabasi, Reka Albert, and Hawoong Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. 2000.

[26] Albert-Laszlo Barabasi, Zoltan Dezso, Erzsebet Ravasz, Soon-Hyung Yook, and Zoltan Oltvai. Scale-free and hierarchical structures in complex networks. *American Institute of Physics Conference Proceedings 661: Modeling of Complex Systems*, 2003.

[27] Albert-Laszlo Barabasi, Erzsebet Ravasz, and Tamas Vicsek. Deterministic scale-free networks. *Physica A: Statistical Mechanics and its Applications*, 2001.

[28] E. R. Barnes and A. J. Hoffman. Partitioning, spectra and linear programming. *Progress in Combinatorial Optimization*, 1984.

[29] Steven M. Bellovin. Using Link Cuts to Attack Internet Routing. Work-in-progress Reports. Usenix Security Symposium, April 2003.

[30] Steven M. Bellovin and Emden R. Gansner. Using Link Cuts to Attack Internet Routing. Technical report, `http://www.research.att.com/~smb/papers/reroute.pdf`, 2003.

[31] Cathleen A. Berrick, Steve Morris, and Gary M. Malavenda. Federal efforts to strengthen security should be better coordinated and targeted on the nation's most critical highway infrastructure. Technical report, United States Government Accountability Office, 2009.

[32] Sergei L. Bezrukov. Edge isoperimetric problems on graphs. 2002.

[33] G. Bianconi and A.-L. Barabasi. Competition and multiscaling in evolving networks. *Europhysics Letters*, 54:436–442, 2001.

[34] Bela Bollobas. *Random Graphs*, chapter 2. Cambridge University Press, 2001.

[35] Bela Bollobas. Mathematical results on scale-free random graphs. 2003.

[36] Bela Bollobas and Oliver Riordan. Robustness and vulnerability of scale-free random graphs. *Internet Mathematics*, 1(1):1–35, 2003.

[37] Anthony Bonato. A servey of models of the web graph. 2004.

[38] A. Broido and K. Claffy. Internet topology: connectivity of IP graphs. *Proceedings of SPIE ITCom*, 2001.

[39] John Burgess, Brian Gallagher, David Jensen, and Brian Neil Levine. MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networks. In *Proc. IEEE INFOCOM*, April 2006.

[40] Brendan Burns, Oliver Brock, and Brian Neil Levine. Mv routing and capacity building in disruption tolerant networks. *The IEEE Conference on Computer Communications*, 2005.

[41] Kevin Butler, William Aiello, and Patrick McDaniel. Optimizing BGP Security by Exploiting Path Stability. In *Proc. ACM Computer and Comunnications Security (CCS)*, pages 298–310, October 2006.

[42] Duncan S. Callaway, M. E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. Network robustness and fragility: Percolation on random graphs. *Physical Review Letters*, 2000.

[43] J. M. Carlson and John Doyle. Highly optimized tolerance: a mechanism for power laws in designed systems. *Physics Review E*, 1999.

[44] Satyabrata Chakrabarti and Amitabh Mishra. Qos issues in ad hoc wireless networks. *IEEE Communications Magazine*, 2001.

[45] Haowen Chan, Debabrata Dash, Adrian Perrig, and Hui Zhang. Modeling Adoptability of Secure BGP Protocols. In *Proc. ACM SIGCOMM*, pages 279–290, September 2006.

[46] P. Chan, M. Schlag, and J. Zien. Spectral k-way ratio-cut partitioning and clustering. *ACM/IEEE Design Automation Conference*, 1993.

[47] Hyunseok Chang, Sugih Jamin, and Walter Willinger. Internet connectivity at the as-level: an optimization-driven modeling approach. *ACM Special Interest Group on Data Communication*, 2003.

[48] Hyunseok Chang, Sugih Jamin, and Walter Willinger. What causal forces shape Internet connectivity at the AS-level. Technical report, University of Michigan, 2003.

[49] Ling-Jyh Chen, Chen-Hung Yu, Tony Sun, Yung-Chih Chen, and Hao-hua Chu. A hybrid routing approach for opportunistic networks. *ACM Special Interest Group on Data Communication*, 2006.

[50] F. R. K. Chung. Graphs with small diameter after edge deletion. *Discrete Applied Mathematics*, 1991.

[51] F. R. K. Chung and M. R. Garey. Diameter bounds for altered graphs. *Journal of Graph Theory*, 1984.

[52] Fan Chung. Discrete isoperimetric inequalities. *Proc. Sympos. Appl. Math*, 1991.

[53] Fan Chung, Linyuan Lu, and Van Vu. Eigenvalues of random power law graphs. 2003.

[54] Fan Chung and Kevin Oden. Weighted graph laplacians and isoperimetric inequalities. 2000.

[55] Reuven Cohen, Keren Erez, Daniel ben Avraham, and Shlomo Havlin. Breakdown of the Internet under intentional attack. *Physical Review Letters*, 2001.

[56] James Davis, Andy Fagg, and Brian Neil Levine. Wearable Computers and Packet Transport Mechanisms in Highly Partitioned Ad hoc Networks. In *Proc. IEEE Intl. Symp on Wearable Computers (ISWC)*, pages 141–148, October 2001.

[57] Anthony H. Dekker and Bernard D. Colbert. Network robustness and graph topology. 2004.

[58] Xenofontas Dimitropoulos, Dmitri Krioukov, George Riley, and KC Claffy. Revealing the autonomous system taxonomy: The machine learning approach. *Passive and Active Measurement Conference*, 2006.

[59] Chris H Q Ding, Xiaofeng He, and Hongyuan Zha. A spectral method to separate disconnected and nearly-disconnected Web graph components. *The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[60] Thang N. Dinh, Ying Xuan, My T. Thai, E.-K. Park, and Taieb Znati. On approximation of new opimization methods for assessing network vulnerability. *Proceedings of the IEEE Communications Society*, 2010.

[61] Danny Dolev, Sugih Jamin, Osnat Mokryn, and Yuval Shavitt. Internet resiliency to attacks and failures under BGP policy routing. Technical report, Hebrew University, 2003.

[62] W.E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17:420–425, 1973.

[63] John C. Doyle, David L. Alderson, Lun Li, Steven Low, Matthew Roughan, Stanislav Shalunov, Reiko Tanaka, and Walter Willinger. The "robust yet fragile" nature of the Internet. *Proceedings of the National Academy of Sciences of the United States of America*, 2005.

[64] R. Elsasser, T. Lucking, and B. Monien. New spectral bounds on k-partitioning of graphs. *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, 2001.

[65] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: Vehicular Content Delivery Using WiFi. In *Proceedings of ACM MobiCom*, San Francisco, CA, September 2008.

[66] Alex Fabrikant, Elias Koutsoupias, and Christos H. Papadimitriou. Heuristically optimized trade-offs: a new paradigm for power laws in the Internet. *ACM Symposium on Theory of Computing*, 2002.

[67] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. *Special Interest Group on Data Communications*, 1999.

[68] Stephen Farrell, Vinny Cahill, Dermot Geraghty, Ivor Humphreys, and Paul McDonald. When tcp breaks: Delay- and disruption-tolerant networking. *IEEE Internet Computing*, 2006.

[69] M Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 1973.

[70] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 1973.

[71] Miroslav Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 1974.

[72] Abraham Flaxman, Alan Frieze, and Juan Vera. Adversarial deletion in a scale free random graph process. *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, 2005.

[73] Lixin Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Transactions On Networking*, 2001.

[74] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.

[75] Christos Gkantsidis, Milena Mihail, and Ellen Zegura. Spectral analysis of internet topologies. *Proceedings of IEEE Conference on Computer Communications*, 2003.

[76] S.P. Gorman, Schintler L., R. Kulkarni, and R. Stough. The revenge of distance: Vulnerability analysis of critical information infrastructure. *Journal of Contingencies and Crisis Management*, To appear.

[77] Leo Grady and Eric Schwartz. Isoperimetric graph partitioning for data clustering and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3), 2006.

[78] Tony H. Grubesic, Timothy C. Matisziw, Alan T. Murray, and Diane Snediker. Comparative approaches for assessing network vulnerability. *International Regional Science Review*, 2008.

[79] Stephen Guattery and Gary L. Miller. On the performance of spectral graph partitioning methods. *Symposium on Discrete Algorithms*, 1995.

[80] Jean-Loup Guillaume, Matthieu Latapy, and Clemence Magnien. Comparison of failures and attacks on random and scale-free networks. *International Conference on Principles of Distributed Systems*, 2005.

[81] Lars Hagen and Andrew B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design*, 1992.

[82] C. Helmberg. Semidefinite programming. *European Journal of Operational Research*, 1999.

[83] B. Hendrickson and R. Leland. *The Chaco User's Guide*, 2.0 edition, 1995.

[84] Alvin Hickson. Terrorist threat to u.s. highway system. Technical report, U.S. Department of Homeland Security Transportation Security Administration, 2006.

[85] Petter Holme, Beom Jun Kim, Chang No Yoon, and Seung Kee Han. Attack Vulnerability of Complex Networks. *APS Physics Review E*, 65(5):056109.1–056109.14, May 2002.

[86] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. *Wireless Networks*, 11(1-2):21–38, 2005.

[87] Yih-Chun Hu, Adrian Perrig, and Marvin Sirbu. SPV: secure path vector routing for securing BGP. In *Proc. ACM SIGCOMM*, pages 179–192, 2004.

[88] Pan Hui, Augustin Chaintreau, James Scott, Richard Gass, Jon Crowcroft, and Christophe Diot. Pocket Switched Networks and Human Mobility in Conference Environments. In *Proc. ACM Workshop on Delay-Tolerant Networking*, pages 244–251, Aug. 2005.

[89] Mouhamad Ibrahim, Ahmad Al Hanbali, and Philippe Nain. Delay and Resource Analysis in MANETs in Presence of Throwboxes. In *Proc. Performance 2007*, October 2007.

[90] Sushant Jain, Michael Demmer, Rabin Patra, and Kevin Fall. Using redundancy to cope with failures in a delay tolerant network. *ACM Special Interest Group on Data Communication*, August 2005.

[91] Sushant Jain, Kevin Fall, and Rabin Patra. Routing in a delay tolerant network. *ACM Special Interest Group on Data Communication*, 2004.

[92] P. Juang et al. Energy-Efficient Computing for Wildlife Tracking: design tradeoffs and early experiences with ZebraNet. *SIGOPS Oper. Syst. Rev.*, 36(5):96–107, 2002.

[93] H. Jun, M. H. Ammar, M. D. Corner, and E. Zegura. Hierarchical Power Management in Disruption Tolerant Networks with Traffic-Aware Optimization. In *Proc. ACM SIGCOMM Workshop on Challenged Networks (CHANTS)*, September 2006.

[94] Hyewon Jun, Wenrui Zhao, Mostafa Ammar, Ellen Zegura, and Chungki Lee. Trading latency for energy in densely deployed wireless ad hoc networks using message ferrying. *Elsevier Journal of Ad Hoc Networks*, 2006. To Appear.

[95] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: attacks and countermeasures. *Ad hoc Networks (Elsevier)*, 1(2–3):293–315, September 2003.

[96] G. Karypis and V. Kumar. *Metis: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*, 4.0 edition, 1998.

[97] S. Kent, C. Lynn, J. Mikkelson, and K. Seo. Secure Border Gateway Protocol (S-BGP) — Real World Performance and Deployment Issues. In *Proc. ISOC Symposium on Network and Distributed System Security (NDSS)*, February 2000.

[98] Rohit Khandekar, Satish Rao, and Umesh Vazirani. Graph partitioning using single commodity flows. *Annual ACM Symposium on Theory of Computing*, 2006.

[99] J. Kleinberg. Detecting a network failure. *Internet Mathematics*, 1(1):37–56, 2000.

[100] Jon Kleinberg, Mark Sandler, and Aleksandrs Slivkins. Network failure detection and graph connectivity. *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, 2003.

[101] JEAN-JACQUES LAFFONT, SCOTT MARCUS, PATRICK REY, and JEAN TIROLE. Internet peering. *The American Economic Review*, 2001.

[102] Ying-Cheng Lai, Adilson Motter, Takashi Nishikawa, Kwangho Park, and Liang Zhao. Complex networks: dynamics and security. *Pramana Journal of Physics*, 2005.

[103] Yee Wei Law, Lodewijk van Hoesel, Jeroen Doumen, Pieter Hartel, and Paul Havinga. Energy-efficient link-layer jamming attacks against wireless sensor network MAC protocols. In *Proc. ACM workshop on Security of ad hoc and sensor networks (SASN)*, pages 76–88, Nov 2005.

[104] Heejo Lee and Jong Kim. Attack resiliency of network topologies. *Parallel and Distributed Computing : Applications and Technologies*, 2004.

[105] Jeremie Leguay, Timur Friedman, and Vania Conan. Dtn routing in a mobility pattern space. *ACM Special Interest Group on Data Communication*, 2005.

[106] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. 1999.

[107] Aruna Balasubramanianm Brian Neil Levine and Arun Venkataramani. Enabling Interactive Applications for Hybrid Networks. *The Annual International Conference on Mobile Computing and Networking*, 2008.

[108] Feng Li, Jie Wu, and Avinash Srinivasan. Thwarting blackhole attacks in distruption-tolerant networks using encounter tickets. *IEEE International Conference on Computer Communications*, pages 2428–2436, April 2009.

[109] Lun Li, David Alderson, Reiko Tanaka, John C. Doyle, and Walter Willinger. Towards a theory of scale-free graphs: definition, properties, and implications (extended version). Technical report, California Institute of Technology, 2005.

[110] Lun Li, David Alderson, Walter Willinger, and John Doyle. A first principles approach to understanding the Internet's router-level topology. *ACM Spectial Interest Group on Data Communication*, 2004.

[111] Yong Liao, Kun Tan, Zhensheng Zhang, and Lixin Gao. Estimation based erasure-coding routing in delay tolerant networks. *The International Wireless Communications and Mobile Computing Conference*, 2006.

[112] Christian Locher, Bjorn Scheuermann, Christian Wewetzer, Andreas Luebke, and Martin Mauve. Data aggregation and roadside unit placement for a vanet traffic information system. *ACM International Workshop on VehiculAr Inter-NETworking*, 2008.

[113] Damien Magoni. Tearing down the internet. *IEEE Journal on Selected Areas in Communications*, 2003.

[114] Bojan Mohar. The laplacian spectrum of graphs. *Proceedings of the Sixth Quadrennial Conference on the Theory and Applications of Graphs*, 1998.

[115] Bojan Mohar and Svatopluk Poljak. *Eigenvalues in combinatorial optimization.* Springer-Verlag, 1993.

[116] S. Murphy. BGP Security Vulnerabilities Analysis. Technical Report draft-ietf-idr-bgp-vuln-00, IETF draft, February 2002.

[117] Thang Nguyen Bui and Curt Jones. Finding good approximate vertex and edge partitions in NP-hard. *Information Processing Letters*, 1992.

[118] Seung-Taek Park, Alexy Khrabrov, David M. Pennock, Steve Lawrence, C. Lee Giles, and Lyle H. Ungar. Static and dynamic analysis of the internet's suseptibility to faults and attacks. *IEEE Conference on Computer Communications*, 2003.

[119] F. Pellegrini. *SCOTCH and LIBSCTOCH User's Guide*, 4.0 edition, 2006.

[120] A. Pentland, R. Fletcher, and A. Hasson. DakNet: Rethinking Connectivity in Developing Nations. *IEEE Computer*, 37(1):78–83, Jan 2004.

[121] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Wireless sensor networks*, 2004.

[122] Alex Pothen, Horst D. Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 1990.

[123] Kevin Poulsen. The backhoe: A real cyberthreat. *Wired*, Jan 2006.

[124] R. Preis and R. Diekmann. *The PARTY Partitioning- Library User Guide*, 1.1 edition, 1996.

[125] Anirudh Ramachandran and Nick Feamster. Understanding the network-level behavior of spammers. In *Proc. ACM SIGCOMM*, pages 291–302, 2006.

[126] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A.-L. Barabasi. Hierarchical organization of modularity in metabolic networks. *Science*, 2002.

[127] Y. Rekhter and P. Gross. RFC 1772: Application of the Border Gateway Protocol in the Internet, March 1995.

[128] Arnold L. Rosenberg and Lenwood S. Heath. *Graph Separators with Applications*. Springer, 2001.

[129] Kimaya Sanzgiri, Bridget Dahill, Daniel LaFlamme, Brian Neil Levine, Clay Shields, and Elizabeth Belding-Royer. Authenticated Routing for Ad hoc Networks. *IEEE/ACM Journal of Selected Areas in Communications: Special issue on Wireless Ad hoc Networks (JSAC)*, 23(3):598–610, March 2005.

[130] Andrew J. Seary and William D. Richards. Partitioning networks by eigenvectors. *In Proceedings of the International Conference on Social Networks*, 1995.

[131] Norbert Sensen. Lower bounds and exact algorithms for the graph partitioning problem using multicommodity flows. *European Symposium on Algorithms*, 2001.

[132] B. Smith and J. Garcia-Luna-Aceves. Securing the Border Gateway Routing Protocol. In *Proc. of Global Internet*, pages 103–116, November 1996.

[133] Bradley R. Smith, Shree Murthy, and J.J. Garcia-Luna-Aceves. Securing Distance Vector Routing Protocols. In *Proc. ISOC NDSS*, Feb 1997.

[134] John Solis, N. Asokan, Kari Kostiainen, Philip Ginzboorg, and Jörg Ott. Controlling resource hogs in mobile delay-tolerant networks. *Computer Communications*, 33(1):2–10, January 2010.

[135] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi S. Raghavendra. Spray and wait: An efficient routing scheme for intermittently connected mobile networks. *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, 2005.

[136] Carson W. Taylor. The anatomy of a power grid blockout. *IEEE Power and Energy Magazine*, 2006.

[137] Md Yusuf Sarwar Uddin, Ahmed Khurshid, and Hee Dong Jung. Denial in dtns. Technical report, University of Illinois, 2010.

[138] Danica Vukadinovic, Polly Huang, and Thomas Erlebach. On the spectrum and structure of internet topology graphs. *Second international workshop on Innovative Internet Computing Systems*, 2001.

[139] Tao Wan, Evangelos Kranakis, and P.C. van Oorschot. Pretty Secure BGP (psBGP). In *Proc. ISOC Symposium on Network and Distributed System Security (NDSS)*, 2005.

[140] Yong Wang, Sushant Jain, Margaret Martonosi, and Kevin Fall. Erasure-coding based routing for opportunistic networks. *ACM Special Interest Group on Data Communication*, 2005.

[141] David Watts. Security and vulnerability in electric power systems. *35th North American Power Symposium*, 2003.

[142] Jorg Widmer and Jean-Yves Le Boudec. Network coding for efficient communication in extreme networks. *ACM SIGCOMM workshop on Delay-tolerant networking*, 2005.

[143] Wizzy digital courier. http://www.wizzy.org.za.

[144] Henry Wolkowicz and Qing Zhao. Semidefinite programming relaxations for the graph partitioning problem. *Discrete Applied Mathematics*, 1996.

[145] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood. The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks. In *Proc. ACM Mobihoc*, pages 46–57, May 2005.

[146] H Yang, H Y. Luo, F Ye, S W. Lu, and L. Zhang. Security in mobile ad hoc networks: Challenges and solutions. *Topics in Wireless Security*, 2004.

[147] Wenrui Zhao, Yang Chen, Mostafa Ammar, Mark D. Corner, Brian Neil Levine, and Ellen Zegura. Capacity Enhancement using Throwboxes in DTNs. *Proceedings of IEEE International Conf on Mobile Ad hoc and Sensor Systems (MASS)*, 2006.

[148] Shi Zhou and R. J. Mondragon. Redundancy and robustness of AS-level Internet topology and its models. *Electronics Letters*, 2004.

[149] Shi Zhou and Raul J. Mondragon. The missing links in the BGP-based AS connectivity maps. *Passive and Active Measurement Workshop*, 2003.

[150] Haojin Zhu, Xiaodong Lin, Rongxing Lu, Yanfei Fan, and Xuemin Shen. Smart: A secure multilayer credit-based incentive scheme for delay-tolerant networks. *IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY*, 58(8), October 2009.