2014

Characterizing Forced Communication in Networks

Samuel C. Gutekunst
Harvey Mudd College

Characterizing Forced Communication in Networks

Samuel C. Gutekunst

Susan E. Martonosi, Advisor

Mohamed Omar, Reader

HARVEY MUDD COLLEGE

Department of Mathematics

May, 2014

Abstract

This thesis studies a problem that has been proposed as a novel way to disrupt communication networks: the load maximization problem. The load on a member of a network represents the amount of communication that the member is forced to be involved in. By maximizing the load on an important member of the network, we hope to increase that member's visibility and susceptibility to capture. In this thesis we characterize load as a combinatorial property of graphs and expose possible connections between load and spectral graph theory. We specifically describe the load and how it changes in several canonical classes of graphs and determine the range of values that the load can take on. We also consider a connection between load and liquid paint flow and use this connection to build a heuristic solver for the load maximization problem. We conclude with a detailed discussion of open questions for future work.

Contents

List of Figures

List of Tables

Acknowledgments

I gratefully acknowledge the support, guidance and patience of my advisor, Professor Susan Martonosi, during this entire project. I thank the entire faculty and staff of the Harvey Mudd Mathematics Department for their enthusiasm and ability to inspire students, both in and out of the classroom. Lastly, I would like to thank my family and friends at Harvey Mudd for their support and advice over the past four years.

Chapter 1

Introduction

One prominent and relatively modern application of mathematics has been to study networks. Networks can informally be thought of as models of the way objects and/or people interact, and a substantial amount of mathematical interest developed as communication and transportation networks were built. As cities were planned, for example, and mass public transportation systems grew, it became natural to ask how to best create these systems; how could mathematical tools let you build a train or subway system that moved as many people as possible on routes that were as convenient as possible in the most cost-effective way?

As people saw mathematics applied to answer these questions, interest in network theory grew. Networks arise in virtually every academic discipline and their study has led to rich connections between mathematics and these disciplines. One of the most popular, and important, modern applications is to social networks. Social networks model the way people interact, and through "friend graphs" on sites like Facebook, many people have unwittingly seen some of the mathematical relationships underlying network theory. These social networks can be thought of as *communication networks*, networks which models the way objects communicate. As in a social network, these objects can be people, but they can also be technology involved in communication, like computers, servers and phone systems.

One useful way that mathematics has been used to study networks is through the study of *resilience*. Roughly speaking, resilience refers to the robustness of a network and can be modeled in many different ways. We will intuitively say that a network is resilient if, when parts of it are removed, the overall network can still continue to function. A resilient transportation network comprising a subway system, for example, would be one where people could still efficiently get to almost all stations if a few stations had to be shut down for maintenance. Likewise, a social network consisting of people who communicate only through in-person conversation would be resilient if news could still quickly pass through the network if any small group of people left the network.

The importance of resilience can be viewed from two different perspectives. First, we want our own networks to be resilient: we want to build some redundancy in any network so that, if a few pieces of the network shut down, it can still continue to function. Second, we might be trying to exploit a network's lack of resilience. We might, for example, view a criminal organization as a social network. Here we want to know the network's weaknesses and we might ask questions like: are there members of the network whose arrest will effectively cripple the organization?

While we might be able to intuitively understand the resilience of small networks, many networks are too large to be comprehended intuitively. Moreover, the trivial ways of making networks as resilient as possible are often impractical. For example, one might build a resilient network by connecting every pair of members. Doing so, however, quickly leads to a very large number of direct connections. Consider, for instance, the network of public airports in the United States. There are more than 5,000 public airports in the United States, so that an airline wishing to operate a single daily direct flight between each pair of airports would require more than 12 million flights each day.

Many models for calculating or exploiting the resilience of a network have been proposed. In what follows we will discuss a relatively new metric for resilience, the *load*. The load on a member of a network can be thought of as the amount of information forced to be routed through that member. In a social network, this information might be communication flowing between people, and a higher load on a person correlates with higher visibility of that person.

This model can be applied to networks with critical elements. For example, most commercial airlines have a hub, an airport through which many flights are transferred in order to connect many pairs of cities with air service. Airlines like to use hubs because they allow them to efficiently connect pairs of cities. We saw previously 12 million flight legs would be required to directly connect every pair of 5,000 airports. Using a hub, an airline could only operate flight routes between the hub and each other airport. Then passengers can fly between any pair of cities with just 5,000 flight legs, though almost all routes would require a connection at the hub. In addition, such a network would be very vulnerable. Its lack of resilience stems from having so much dependence on its hub: if the hub was shut down, perhaps because of weather, the airline would not be able to operate a single flight.

These types of networks, where there are a few critically important members, are quite common. They may not be as concentrated or vulnerable as in the previous example, but many networks still rely heavily on a few key actors. In a criminal network, it might be the leader or it might be the person who finances the criminal activities. In a communications network, it might be a central server through which all communication is routed.

The load on one of these key members measures how much information must pass through them. We will formalize these ideas in the next chapter, and we will primarily apply them to communication and social networks. We will also use a specific example, introduced below, of a transportation network to show how these concepts might be applied in a commercial setting.

1.1 Motivation

The concept of load is particularly natural in communication networks but it can also be applied to many other types of networks. We will start by introducing a more concrete example of load in a transportation network, and we will refer back to this example throughout parts of this thesis to provide context for our work. This example focuses on network destabilization and is, to be candid, somewhat contrived and is to be taken as a lighthearted example. At the same time, it should not detract from the importance of resilience and methods of network disruption. One particularly relevant application of this work is to the destabilization of terrorist networks. Much of the research about network disruption focuses on terror networks, and indeed the initial motivation for the concept of load comes from terrorist networks (see Martonosi et al. (2011)). Appropriately discussing the structure of terror networks, however, is beyond the scope of this thesis and the interested reader should consult works by Hoffman (2006) and Sageman (2004).

Suppose that the company *Cheap Unequaled Transportation* (CUT) owns an airline. The set of airports that the airline serves and the routes between airports that passengers can take serve as our network.

An airline operator, *Flights out of this World*, (FloW) owns the Menger airport. They let other airlines, including CUT, fly through the Menger airport, but the other companies have to pay a considerable fee to FloW. Company FloW wants to acquire another hub, so they will buy out and renovate a different airport. At the same time, this acquisition will cost FloW a tremendous amount of money, and to complete their renovations, they need to raise profits during the renovation period.

As they renovate, they will effectively close the airport they acquire for several years, causing other airlines to reroute flights through different airports. The key question for FloW is then: *which airport should we temporarily close so that company CUT has to route more flights through the Menger airport, increasing our profits*? This question is essentially asking how to select an airport to remove from the network so that the load on the Menger airport increases the most. Though contrived, this example shows how a company might be interested in the load metric.

In the remainder of this thesis we will examine the relationship between the structure of networks and load. We will start by discussing what the load actually can look like and then highlight connections to a beautiful field of mathematics: spectral graph theory. Our ultimate goal is to contribute theorems and empirical results that help characterize the relationship between load and network structure so that a company like FloW could quickly solve their problem.

1.2 Outline

As much as possible, we would like for this thesis to be comprehensible to a motivated and patient underclassmen with some exposure to college mathematics. In particular, we assume familiarity with calculus, linear algebra and proof-based mathematics. For someone familiar with those courses, we will present material with enough background so that this thesis can be self contained and accessible. We will also provide several references for further exploration of related material.

We will begin this thesis, in the next chapter, by providing a more mathematical context for the load metric. We will discuss background from graph and network theory as well as the most important characterizations of load from previous literature.

In the third and fourth chapters, we present new theorems that are contributions of this thesis project. These new contributions fall into two broad categories. First, in Chapter Three we answer questions that help to characterize load as a structural property of networks. We will describe the load, and how it changes, in broad classes of networks. In addition, we will bound how large the load can actually be in a network. These sorts of questions provide insight into the load as a metric as well as intuition about when it is large.

In Chapter Four, we will study load using tools from spectral graph theory. Spectral graph theory studies networks through the eigenvalues and eigenvectors of matrices that can be associated with graphs. Techniques from the field have been successfully applied to many problems in network theory and have been used to study the structural properties of graphs. These techniques have notably been used to study other sorts of flow problems. In this chapter we will highlight one of these problems, the paint spilling problem. We will define this problem formally and provide empirical data connecting it to load.

Chapter Five explains a series of open problems and describes areas for further research. Chapter Six then briefly concludes this thesis, summarizing the main motivations and contributions.

Chapter 2

Background

We begin this chapter by describing the relevant technical background, starting with pertinent details from graph and network theory. We then introduce the method of network disruption studied in this project and describe the work that has been done so far.

2.1 Network and Graph Theory

We previously described a network as a tool to understand the relationships between objects, and we provided examples of communication, transportation and social networks. Networks are often studied and visualized as graphs, which we formally define below. One of the challenges of graph theory is getting used to the vocabulary, so below we emphasize what is relevant to this thesis.

Readers familiar with graph theory may jump to Section 2.2, but we encourage them to at least skim the following to become familiar with the terminology we will use. For a more thorough introduction to the graph theory involved in studying networks, see chapters six through eight of Newman (2010). Readers interested in learning more about graph theory in general are referred to West (2001), an approachable and well-written introductory textbook.

Definition 2.1 (Graphs)**.** *A* ***graph*** $G = (V(G), E(G))$ *is defined as two sets,* $E(G)$ *and* $V(G)$*. Elements of* $E(G)$ *are sets consisting of two elements of* $V(G)$*.* V *is referred to as the* ***vertex set****, and* E*, the* ***edge set****. We typically let* $n = |V(G)|$ *and* $m = |E(G)|$*. If* $\{u, v\} \in E$*, we say* u *is* ***adjacent*** *to* v*, or equivalently that* u *is a* ***neighbor*** *of* v*, and write* $u \sim v$*. An edge* $\{u, v\}$ *is said to be* ***incident*** *to* u *and* v*.*

An example graph G is shown below in Figure 2.1. In this graph the vertex set is

$$V(G) = \{\text{Alice, Bob, Charlie, Dan, Key}\}$$

and the edge set is

$$\{\{\text{Alice, Bob}\},\{\text{Alice, Dan}\},\{\text{Alice, Key}\},\{\text{Bob, Charlie}\},\\ \{\text{Charlie, Dan}\},\{\text{Charlie, Key}\}\}.$$

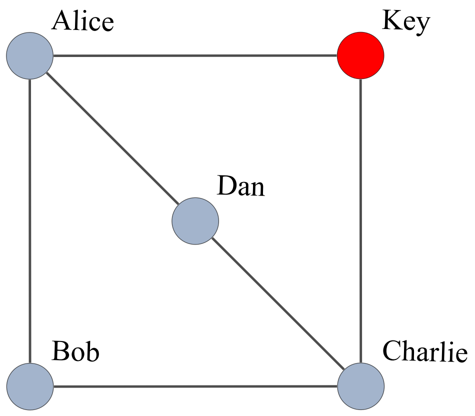


Figure 2.1 Visualizing a social network as a graph.

Elements of the vertex set will be referred to as **vertices**, though the term **node** is also frequently used in related literature. In a network, vertices typically symbolize objects, and edges, relationships between those objects. In our motivating example of an airport network, each airport that CUT operates out of would be a vertex. An edge exists between two vertices if CUT operates a direct flight between the airports corresponding to those vertices. In a social network, vertices might represent people, and edges, ways those people can directly communicate. Such a social network is shown in Figure 2.1. In the picture, Alice and Bob can directly communicate as there is an edge between the vertices representing them. Bob can communicate with the Key vertex (who might represent a leader in the network), but his communication is indirect and must pass through at least one other person.

The previous definition of a graph can be modified in several ways. We will study **simple graphs**, which are graphs that do not allow loops and repeated edges. A loop is an edge of the form $\{v, v\}$ where $v \in V(G)$.

Having no repeated edges means we do not allow $\{u, v\}$ with $u, v \in V(G)$ to appear in $E(G)$ multiple times (in other words, in a simple graph G, $E(G)$ is a set of distinct two-element subsets of $V(G)$, each consisting of two distinct elements).

Our graphs are also undirected,which means that $u \sim v$ is equivalent to $v \sim u$. In a directed graph, we would alternatively define $E(G)$ to be a subset of $V(G) \times V(G)$ so that edges would be expressed as (u, v) with $u, v \in V(G)$. This edge would be described as an edge from u to v, and would be used, for example, to model situations where a person represented by u could send information to the person represented by v, but not vice versa. Again, we will assume that our edges are undirected. In this case, we refer to the **degree** of a vertex as the number of vertices it is adjacent to. Thus, in Figure 2.1, Bob has degree 2 while Alice has degree 3.

Adjacency and edges allow us to represent direct connections between vertices, but we also want to express indirect connections. In Figure 2.1 communication can still flow between Bob and the Key vertex by using Alice or Charlie as an intermediary. Because communication can still flow between them, we say that Bob and the Key vertex are **connected**. This, more formally, means that there exists a *path* between Bob and the Key vertex:

Definition 2.2 (Paths)**.** *Let G be a graph with vertices u and v. A u-v **path** is a sequence of vertices $u = w_1, w_2, ..., w_k = v$ such that w_i is adjacent to w_{i+1} and no vertex occurs twice in the list. The **length** of a path is the number edges traversed between u and v, equal to $k - 1$. Two paths are **edge disjoint** if no edge of G occurs in both paths, and a set of paths are **pairwise edge disjoint**, or just **edge disjoint**, if every pair of paths in the set is edge disjoint.*

In Figure 2.1, there are three paths between Bob and Alice. These are the path of length one corresponding to moving directly from Bob to Alice, which we denote by P_1, the path of length three expressed in a list as: Bob, Charlie, Dan, Alice, which we denote by P_2, and another path of length three: Bob, Charlie, Key, Alice, which we denote by P_3. P_1 and P_2 are edge disjoint, so the set $\{P_1, P_2\}$ is also edge disjoint. Because P_2 and P_3 share the edge between Bob and Charlie, they are not edge disjoint and both of the sets $\{P_1, P_2, P_3\}$ and $\{P_2, P_3\}$ are not edge disjoint. The maximum number of edge disjoint paths between Alice and Bob is then two.

If a path P is listed as $w_1, \ldots, w_n$, we will refer to w_1 and w_n as **endpoints** of P. The other vertices, $w_2, \ldots, w_{n-1}$, will be described as **internal** vertices of P. In our motivating example, a path between two vertices u and v means that there exists a series of flights that will take a passenger from airport u to airport v. Any internal vertices on that path correspond to the airports where a passenger would experience connections while flying from airport u to airport v.

Definition 2.3 (Cycles)**.** *A **cycle** is a sequence of vertices $u = w_1, w_2, ..., w_k = u$ such that such that w_i is adjacent to w_{i+1} and u is the only vertex that appears twice in the list. A graph G is said to be **acyclic** if it contains no cycles. The **length** of a cycle is the number of distinct vertices it contains, namely $k - 1$.*

Intuitively, a cycle is a path from w_1 to w_{n-1} followed by a return to w_1 using an edge $\{w_1, w_{n-1}\}$. There are multiple cycles of length four in Figure 2.1. One example is Alice, Bob, Charlie, Dan, Alice. Note that we could define the same cycle moving in either direction, and we could equivalently define it as starting from any vertex in the cycle.

In many graphs, there exist paths between all pairs of vertices. In our motivating example, this would mean that it is possible, with enough connecting flights, to go between any pair of airports owned by CUT. These graphs are referred to as *connected*:

Definition 2.4 (Connectedness)**.** *A graph G is said to be **connected** if, for any $u, v \in V(G)$, there exists a u-v path. A subset S of $V(G)$ is similarly said to be **connected** if, for every $u, v \in S$ there exists a u-v path. If the graph is not connected, maximal connected subsets of vertices are called **components**.*

A "maximal connected subset" informally means any subset of vertices that is connected and cannot be made bigger without resulting in a subset that was no longer connected.

We use the idea of a path to define the distance between two vertices:

Definition 2.5 (Distance in a Graph)**.** *Let G be a graph with vertices u and v. The **distance** between u and v, denoted by $d(u, v)$, is the length of the shortest u-v path. We refer to shortest paths as **geodesics**. If no u-v path exists, we use the convention that $d(u, v) = \infty$.*

Above we have defined the basic tools that we will use from network and graph theory. These tools allow us to mathematically define what it might mean for a vertex to be important.

2.2 Centrality and Load

One common method of network disruption is to remove a critical vertex and it is thus important to formalize measures of a vertex's prominence in a graph. We mathematically formalize that prominence through a vertex's *centrality*. There are many measures that try to describe how central a vertex is in a graph, and we will pay particular attention to the following:

Definition 2.6 (Centrality)**.**

(a) The ***degree centrality*** *of a vertex v in a graph G is its degree.*

(b) The ***closeness centrality*** *of v is defined as*

$$\frac{n}{\sum_{u \in V(G)} d(u,v)}.$$

(c) The ***betweenness centrality*** *of v is defined as*

$$\sum_{u,w \in V(G)} \frac{g_{u,w}(v)}{g_{u,w}},$$

where $g_{u,w}$ is the number of geodesics between u and w, while $g_{u,w}(v)$ is the number of geodesics between u and w containing v as an internal vertex.

The degree centrality is a relatively simple measure of a vertex's importance within a graph and assumes that the more important a vertex is, the more neighbors it will have. Closeness centrality measures how close a given vertex v is to every other vertex, and betweenness centrality counts the number of geodesics v is involved in. If communication is expected to flow as efficiently as possible it will flow through geodesics. The betweenness centrality then measures the number of times v lies between two vertices on a path where efficient communication is expected to flow. (Note that the expressions defining these centrality measures can be scaled without changing their meaning, and the reader is advised that they appear in different forms in different literature. For consistency, we have used definitions from Newman (2010).)

It many networks, however, it is not the case that communication will tend to flow along the shortest possible path. We thus consider the load on a given vertex k which measures the number of distinct paths that are forced to include k.

Definition 2.7 (Load)**.** *The* ***load*** *on a vertex k in a graph G is*

$$L(k,G) := \sum_{u,v \neq k} n_{u,v}(G) - \sum_{u,v \neq k} n_{u,v}(G \backslash k),$$

where $n_{u,v}(G)$ is the maximum number of edge disjoint u-v paths in a graph G and $G \backslash k$ is the graph G with the vertex k and all edges incident to k removed.

The first term in this definition sums over all pairs of non-key vertices and counts the maximum number of edge disjoint paths between them. In a communication network, this term can be intuitively thought of as adding together all of the distinct ways pairs of people can communicate. The second term repeats the calculation, but in the network with the key vertex removed. The result effectively measures the amount of communication the key vertex is forced to participate in.

This intuition extends beyond communication. In our motivating example, we want to increase the number of CUT's flights that are forced to fly through the Menger airport; this is analogous to increasing the load on the Menger airport. FloW, by closing another airport, is removing a vertex from the graph representing CUT's transportation network. FloW wants to find the airport whose removal most increases the load on the Menger airport in CUT's network. This problem is known as a special case of the *load maximization*, or LoMax, problem which is the main study of this thesis.

2.3 LoMax

To formalize the load maximization problem we need to define the load effect of a subset $S \subset V(G)$ on a key vertex.

Definition 2.8 (Load Effect)**.** *Let $S \subset V(G)$ be a subset of vertices in a graph G such that $k \notin S$. Then the **load effect** of S on k is:*

$$\varepsilon(S, k, G) := L(k, G \backslash S) - L(k, G),$$

where $G \backslash S$ is the graph G with every vertex in S removed.

The load effect of a subset of vertices with respect to a key vertex k thus measures how much the load on k increases when that subset is removed.

To disrupt a network, our goal is to increase, as much as possible, the load on the key vertex. In social networks, the key vertex is often a leader of the network whose removal from the network would substantially affect the network's ability to function. The leader is thus typically expected to be a highly central vertex.

Definition 2.9 (LoMax)**.** *Let G be a graph with a key vertex k. The **load maximization**, or **LoMax**, problem is:*

$$\max_{S \subset V(G \backslash k)} \varepsilon(S, k, G).$$

The special case of LoMax that FloW would be interested in is the **single-LoMax problem**. This problem adds the constraint that the subsets considered must contain exactly one vertex. Algebraically, single-LoMax can be expressed as

$$\max_{v \in V(G\backslash k)} \varepsilon(\{v\}, k, G).$$

One might wonder if or when it is possible to increase the load on a fixed vertex in a graph. The former question will be answered in the affirmative by the following example, and answering the latter is one of the primary goals of research into load.

Example 2.1 (Computing Load and Solving LoMax)**.** *Consider the network shown in Figure 2.1 and let $S = \{Dan\}$. The maximum numbers of edge-disjoint paths between pairs of vertices in G, $G\backslash k$, $G\backslash S$ and $(G\backslash S)\backslash k$ are shown in Table 2.1 and were computed as in the remarks following Definition 2.2. The load on the key vertex in G is, following Definition 2.7:*

$$\begin{aligned} L(k, G) &= \sum_{u,v \neq k} n_{u,v}(G) - \sum_{u,v \neq k} n_{u,v}(G\backslash k) \\ &= 13 - 12 \\ &= 1. \end{aligned}$$

Similar computations show that $L(k, G\backslash S) = 3$ so that removing Dan from the network increases the load on the key vertex to 3. By trying all subsets of vertices, it can be shown that this is optimal. Thus, an optimal solution to the LoMax problem is $S = \{Dan\}$, and $\varepsilon(S, k, G) = 2$.

In the preceding example, we made the comment that, to show that S was optimal, it would be sufficient to consider all 2^{n-1} possible subsets of $V(G\backslash k)$, where n is the number of vertices in G. This strategy, however, becomes exponentially more difficult to compute with the size of the network and is very inefficient. One principal question, then, is characterizing the subsets of vertices with large load effects on highly central vertices.

2.4 Approaches to LoMax

Martonosi et al. (2011) first presented the LoMax and single-LoMax problem with applications to disrupting terrorist networks. They argued that

Maximum Number of Edge Paths in Figure 2.1 and Related Graphs				
Endpoints (u,v)	$n_{u,v}(G)$	$n_{u,v}(G\backslash k)$	$n_{u,v}(G\backslash S)$	$n_{u,v}((G\backslash S)\backslash k)$
Alice, Bob	2	2	2	1
Alice, Charlie	3	2	2	1
Alice, Dan	2	2	n/a	n/a
Bob, Charlie	2	2	2	1
Bob, Dan	2	2	n/a	n/a
Charlie, Dan	2	2	n/a	n/a
Total	13	12	6	3

Table 2.1 Computing the load in Example 2.1.

classical network disruption methods are not always effective. These methods, for example, tend to focus on preventing all communication through a network (which is very rarely feasible) or increasing the length of geodesics. The latter would, intuitively, make it more difficult for communication to flow between members of the network, but might not be helpful since covert networks may already use long paths of communication.

Martonosi et al. also introduced two broad ways of considering the LoMax problem: computing solutions efficiently and identifying graph-theoretic structural properties of optimal solutions. This thesis primarily explores the latter, but its importance is motivated by the difficulty of implementing LoMax solvers, and we now briefly discuss previous work in this area.

2.4.1 Solving LoMax Computationally

The LoMax problem was described in Martonosi et al. (2011), and the authors presented a genetic algorithm that tended to find good (but not necessarily optimal) solutions to the LoMax problem. Paul (2012) then implemented LoMax as a mixed integer linear program. This program found the subset of vertices S that maximized $\varepsilon(S,k,G)$, and tended to run successfully in under four minutes on graphs with up to 30 vertices and 68 edges. However, even with some simplifications, it failed to determine the optimal subsets on graphs with 30 vertices and more than 75 edges.

Quickly solving LoMax on large networks seems to be computationally intractable, so in both Martonosi et al. (2011) and Paul (2012), the authors began to more theoretically examine the subsets of vertices whose removal most increases the load on a key vertex. If the subsets of vertices that solved the LoMax problem could be structurally characterized, new and more efficient LoMax solvers could be developed by searching for subsets with those properties.

2.4.2 Characterizing Optimal Solutions

Below, we summarize the previous theoretical results relevant to this thesis, beginning with the original results in Martonosi et al. (2011). In the following, let G be a graph with key vertex k. Then:

Theorem 2.1 (Theorem 1 in Martonosi et al. (2011)). *If k has degree two, and $v \in V(G)$ is such that $v \sim k$, then $\varepsilon(\{v\}, k, G) \leq 0$.*

Theorem 2.2 (Theorem 2 in Martonosi et al. (2011)). *Suppose G is a cycle on n vertices (so that every vertex has degree two and the graph is connected). Then for any $v \in V(G)$, $\varepsilon(\{v\}, k, G) \leq 0$.*

Theorem 2.3 (Theorem 3 in Martonosi et al. (2011)). *If there is a single edge-disjoint path between k and a vertex v, then $\varepsilon(\{v\}, k, G) \leq 0$.*

Theorem 2.4 (Theorem 4 in Martonosi et al. (2011)). *Let $v \neq k$ be a vertex and C be a set of edges whose removal disconnects G into two components, G_k and G_v, respectively containing k and v. Let $v_1, \ldots, v_p$ be vertices in G_v that, in G, are adjacent to vertices in G_k, and let $k_1, \ldots, k_s$ be vertices in G_k that, in G, are adjacent to at least one vertex in G_v. Suppose further that v_j has at least $\lfloor |C|/2 \rfloor$ edge disjoint paths to every other v_i using just vertices in $G_v \backslash i$ and k_l has at least $\lfloor |C|/2 \rfloor$ edge disjoint paths to every other k_i using just vertices in $G_k \backslash k$. Then $\varepsilon(\{v\}, k, G) \leq 0$.*

These theorems highlight connections between cycles and vertices whose removal increases the load on a key vertex. Theorem 2.3 is equivalent to saying that *vertices whose removal increases the load on a key vertex k must be in at least one cycle with k* and Theorem 2.2 adds that *there must be some additional structure in this cycle*.

Continuing the work initiated by Martonosi et al. (2011), Paul (2012) proved several new theorems. Paul (2012) extended several of the above theorems to the full LoMax case. For example, the following is a full LoMax analogue of 2.3:

Theorem 2.5 (Theorem 5.7 in Paul (2012)). *If there is a single edge-disjoint path between k and a vertex v and $S \subset V(G)$ is such that $v, k \notin S$, then*

$$\varepsilon(S, k, G) \geq \varepsilon(S \cup \{v\}, k, G).$$

Just as removing an individual vertex v with a single edge disjoint path to k does not increase the load on k, adding v to any subset of vertices already removed will not increase the load effect on k. This theorem thus says that we need never consider removing v. Paul uses this idea to construct graph simplification algorithms. Theorems like this highlight the ways theoretical study of load can be used to improve solvers for LoMax.

Many of the initial theorems in Martonosi et al. (2011) focused on the load effects of single vertices. There are many ways to extend these results to larger subsets. For example, Paul (2012) considers the load effect of pairs of vertices:

Theorem 2.6 (Theorem 5.11 Paul (2012))**.** *If*

$$\varepsilon(\{u,v\},k,G) > \max\{0,\varepsilon(u,k,G),\varepsilon(v,k,G)\},$$

then neither u nor v has a single edge-disjoint path to k.

Thus, one approach to theoretically examining LoMax is to prove theorems about the load effects of single vertices and then extend them to larger subsets of vertices. This is an approach we will take occasionally in Chapter Three, and the data we present in Chapter Four will also first focus on single vertices.

Results like the above are very important for a large number of reasons. They help us to understand the structure of networks, and, in single-LoMax, immediately let us eliminate vertices from consideration. They also often form the basis of theorems that can be applied to the full LoMax problem. These theorems could be incorporated into future LoMax solvers to potentially increase the sizes of graphs on which LoMax can be run.

We now turn to the primary contribution of this thesis: extending our theoretical understanding of load.

Chapter 3

Characterization of Load

In this chapter we provide results that help characterize load. The load on a vertex, just like the degree of a vertex, can be thought of as simply a number assigned to a given vertex in a given graph. One might then wonder how the load can vary: what range of values can it take on and, in particular, how large can it be? What is it on vertices in well-known classes of graphs with lots of structure that are often studied to build intuition? Moreover, how does the load in those classes of graphs change when subsets of vertices are removed?

We intuitively think of load on a key vertex as the amount of flow forced through that vertex. In the problems that we expect research on load to be applied to, this key vertex is often highly central. One might thus wonder if it is always possible to increase the load on these vertices. While it is easy to find small graphs for which the load cannot be increased, and while it is similarly straightforward to create trivial large graphs where the load cannot be increased (for example, the graph with no edges), one might ask if it is always possible to increase the load on highly central vertices in large graphs with many edges.

This chapter will address the above questions. We begin, in Section 3.1, by describing both the load and changes in load in several well-known classes of graphs. In Section 3.2 we discuss the range of values the load can take on. In particular, we show that the maximum possible load on any vertex in any graph of size n is of order n^3 and construct a graph for which this bound is achieved. In Section 3.3 we then describe how the load on a key vertex can change in any graph when vertices of low degree are removed. We conclude, in Section 3.4, by briefly answering our final question: is it always possible to increase the load on vertices in large graphs with many edges?

3.1 Combinatorial Results

The load on a key vertex in a given graph is computable in polynomial time, meaning that the time it takes to compute the load scales as a polynomial function of the number of vertices. Similarly, the load effect on a key vertex when a specific subset is removed is computable in polynomial time. For some classes of graphs, however, we do not need to compute the maximum number of edge disjoint paths between pairs of vertices to determine the load on a key vertex. Instead, we can directly compute the load on vertices in these graphs using formulas. Some of these formulas are used implicitly in proofs of the theorems discussed previously, but most of them are neither explicitly nor implicitly discussed. These examples may not be realistic networks, but the following combinatorial results also serve as examples of the load and load effects on key vertices in different types of graphs, and thus provide intuition about load.

Before looking at graphs, we state one basic proposition which implies that we only need to look at the component of a graph containing k. This proposition can be used to simplify calculations in networks consisting of multiple components. It will also be useful in the propositions below when removing vertices in such a way that disconnects the graph.

3.1.1 Components

Proposition 3.1 (Components)**.** *Let G be a graph having multiple components. Let G_k denote the component containing k. Then $L(k, G) = L(k, G_k)$, and in determining the load effect of a subset S, we need only concern ourselves with the load effect of $S \cap V(G_k)$ on G_k.*

Proof. This proposition follows because the only vertices that have an effect on the load on k, or on the load effect of a subset S on k, are those that have at least one edge disjoint path containing k. This requires both endpoints to be in component G_k. More formally:

$$L(k, G) = \sum_{u,v \neq k} n_{u,v}(G) - \sum_{u,v \neq k} n_{u,v}(G \backslash k).$$

If there is a u-v path, both vertices must be in the same component and so we separately consider the case when $u, v \in G_k$:

$$
\begin{aligned}
&= \sum_{u,v \notin G_k} n_{u,v}(G) - \sum_{u,v \notin G_k} n_{u,v}(G \backslash k) \\
&\qquad\qquad + \sum_{\substack{u,v \in G_k: \\ u,v \neq k}} n_{u,v}(G) - \sum_{\substack{u,v \in G_k: \\ u,v \neq k}} n_{u,v}(G \backslash k).
\end{aligned}
$$

Note that $\sum_{u,v \notin G_k} n_{u,v}(G) - \sum_{u,v \notin G_k} n_{u,v}(G \backslash k) = 0$ since no paths in components outside G_k can use k. Thus:

$$
\begin{aligned}
&= \sum_{\substack{u,v \in G_k: \\ u,v \neq k}} n_{u,v}(G) - \sum_{\substack{u,v \in G_k: \\ u,v \neq k}} n_{u,v}(G \backslash k) \\
&= L(k, G_k).
\end{aligned}
$$

□

For the sake of brevity, in the following proofs we will not always make a strict bookkeeping argument.

3.1.2 Paths

We begin by looking at path graphs. The path graph on n vertices, P_n, is a connected graph with $n - 1$ edges. In P_n, the vertices can be ordered as $v_1, v_2, \ldots, v_n$ with $v_1 \sim v_2$, $v_2 \sim v_3, \ldots, v_{n-1} \sim v_n$ defining the edges. An example is shown in Figure 3.1a.

Proposition 3.2 (Paths). *Using the notation above, let G be P_n. Then*

$$L(v_i, G) = (i - 1) * (n - i)$$

for all i.

Proof. Any pair of vertices in P_n will have exactly one path between them. The paths that contribute to the load are those that use v_i as an internal vertex. To obtain such a path, one endpoint must be a vertex v_j with index $j < i$, and the other, a vertex v_k with index $k > i$. There are $(i - 1)(n - i)$ such choices. □

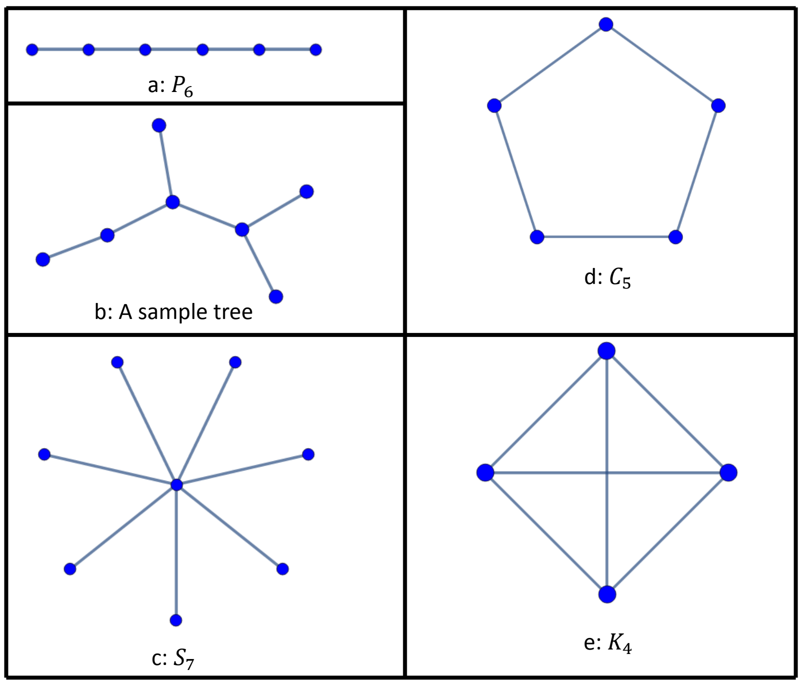


Figure 3.1 Types of graphs having a closed-form expression for the load on a given vertex.

Note that the $L(v_i, G) = 0$ for $i = 1$ and $i = n$. Consider, however, P_{2j+1}. Then $L(v_{j+1}, P_{2j+1}) = j^2$. It can be shown that v_{j+1} is the vertex in P_{2j+1} with the largest load, and thus in a path graph on n vertices the load on a given vertex is between 0 and roughly $\frac{n^2}{4}$.

We can also study the load effect of arbitrary subsets of vertices in path graphs. Removing a vertex other than v_1 or v_n will disconnect the graph, and using Proposition 3.1, we only need to study the component containing the key vertex v_i.

Proposition 3.3 (Load Effects in Paths). *Let G be the graph P_n and let $S \subset V(G \backslash v_i)$. We consider three cases:*

- *If S contains only vertices with indices less than i, let v_j be the vertex in S with maximum index. Then $L(v_i, G \backslash S) = (i - j - 1)(n - i)$. Thus $\varepsilon(S, k, G) = -j(n - i)$.*
- *If S contains only vertices with indexes greater than i, let v_j be the vertex*

in S with minimum index. Then $L(v_i, G\backslash S) = (i-1)(j-i-1)$. Thus $\varepsilon(S,k,G) = -(n-j+1)(i-1)$.

- *If S contains vertices with indexes both greater and less than i, let v_j be the vertex in S with maximum index such that $j < i$ and let v_l be the vertex in S with minimum index such that $l > i$. Then $L(v_i, S) = (i-j-1)(l-i-1)$ so $\varepsilon(S,k,G) = (i-j-1)(l-i-1) - (i-1)(n-i)$.*

Proof. We consider the second case, and the remaining cases are analogous. By 3.1, $L(v_i, G\backslash S)$ will be the same as $L(v_i, P_{j-1})$ and we apply 3.2. To compute the load effect, note that

$$(i-1)(j-i-1) - (i-1)(n-i) = -(i-1)(n-j+1).$$

□

By including v_{i+1} or v_{i-1} in S, $L(v_i, G\backslash S) = 0$. Thus, the removal of a single vertex can decrease the load on any path graph to 0.

3.1.3 Trees

Paths are special cases of trees, defined to be connected, acyclic graphs (or equivalently, any connected graph on n vertices with $n-1$ edges). An example tree is shown in Figure 3.1b. In general, let k be a key vertex in a tree and denote the degree of k by $d(k)$. Then $G\backslash k$ consists of $d(k)$ components, each of which is a tree. The edges adjacent to k can be labeled $e_1, \ldots, e_{d(k)}$, each of which connects k to a tree in $G\backslash k$. These trees in $G\backslash k$ can be respectively labeled $T_1, \ldots, T_{d(k)}$, where T_i is the tree adjacent to k via e_i. Finally, let n_i denote the number of vertices in T_i so that $n_1 + \ldots n_{d(k)} = n-1$.

Proposition 3.4 (Trees)**.** *Let G be a tree on n vertices using the notation above. Then $L(k,G) = \sum_{i<j} n_i n_j$.*

Proof. The proof is analogous to the proof of Proposition 3.2. It follows by considering all pairs of vertices which pass through k. □

Therefore, to determine the load on any vertex k in a tree, sum over each pair of edges incident to k and, for each pair, multiply the number of vertices connected to each edge in $G\backslash k$. For the general case of removing subsets we do not provide explicit formulas. Still, we note the following:

Proposition 3.5 (Load Effects in Trees)**.** *Let G be a tree on n vertices. If k has $d(k)$ neighbors, then at least $d(k)-1$ vertices must be removed to decrease the load on k to 0.*

Proof. Let S be a subset of vertices of $G\backslash k$ such that $|S| < d(k) - 1$. Then k has at least two neighbors, u and v, in $G\backslash S$, connected by a single path through k. Thus the load on k in $G\backslash S$ will be at least one. □

We also consider a special tree, S_n the star graph on n vertices. This tree consists of a single vertex k adjacent to $n-1$ neighbors (and each neighbor has degree 1). An example is shown in Figure 3.1c. This network corresponds to a single highly central vertex, and would model, for example, an airline transportation network with a single hub. Then:

Proposition 3.6 (Load and Load Effects in Star Graphs). $L(k, S_n) = \binom{n-1}{2}$. *If* $S \subset V(G\backslash k)$, *then*

$$\varepsilon(S, k, S_n) = \binom{n-1-|S|}{2} - \binom{n-1}{2}.$$

Proof. This proof follows from Proposition 3.4 and the fact that $G\backslash S = S_{n-|S|}$. □

Note that the load on the center vertex in a star graph is approximately $n^2/2$, so that the load on the center vertex of S_n is asymptotically twice as large as the maximum load on a vertex in P_n.

3.1.4 Cycles

After studying trees, the next step is to consider cycles. C_n, the cycle on n vertices, can be visualized as adding a single edge between v_1 and v_n in P_n. An example is shown in Figure 3.1d. The load on any vertex in a cycle is exactly the same as the load on the center vertex of a star graph:

Proposition 3.7 (Cycles). $L(k, C_n) = \binom{n-1}{2}$.

Proof. For each pair of vertices u and v (with $u, v \neq k$), there are two distinct paths from u to v corresponding to moving clockwise and counterclockwise through C_n. One of these paths will contain k as an internal vertex, and so each of the $\binom{n-1}{2}$ pairs contributes 1 to the load. □

Removing any subset of vertices from C_n will create a set of components, each of which is a path graph. Removing the two vertices adjacent to k will reduce the load to 0.

Proposition 3.8 (Load Effects in Cycles)**.** *The load effect of a single vertex $S = \{v\} \neq \{k\}$ on a cycle C_n is*

$$(d(k,v) - 1)(n - d(k,v) - 1) - \binom{n-1}{2}.$$

If $|S| > 1$, label the vertices in the cycle as follows:

- *Label k as v_1.*
- *Move clockwise through the cycle, labeling the clockwise neighbor of v_i with v_{i+1} until you reach v_n.*

Let v_i be the vertex in S with minimum index and let v_j be the vertex in S with maximum index. Then the load effect of $S \subset V(G\backslash k)$ on C_n is:

$$(i-2)(n-j) - \binom{n-1}{2}.$$

Proof. First consider the case where $|S| = 1$. As noted above, $G\backslash v$ will be a path graph. We thus apply Proposition 3.2 to compute $L(k, G\backslash v)$. There will be $d(k,v) - 1$ vertices to one side of k in this path, and $n - d(k,v) - 1$ on the other side.

If $|S| > 1$, then $G\backslash S$ is still a path graph. There will be $i - 2$ vertices to one side of k (with indices v_2 through v_{i-1}) and $n - j$ vertices on the other side (corresponding to v_n through v_{j+1}). □

3.1.5 Complete Graphs

Finally, we consider the load and load effects on complete graphs. K_n, the complete graph on n vertices, is defined as the graph where every pair of vertices is adjacent so that K_n has all possible $\binom{n}{2}$ edges. An example is shown in Figure 3.1e. These graphs often resemble small social networks where "everyone knows everyone". We have:

Proposition 3.9 (Complete Graphs)**.** $L(k, K_n) = \binom{n-1}{2}$.

Proof. For each pair of vertices u and v (with $u, v \neq k$), there are $n - 1$ distinct paths from u to v (one path corresponding to taking the edge between u and v and $n - 2$ paths of length 2 going through each of the other vertices in K_n). There are at most $n - 1$ edge disjoint paths from u to v since $d(u) = n - 1$. Thus we have found the maximum number of edge disjoint paths between any pair of vertices in G. Noting that $G\backslash k = K_{n-1}$, analogous reasoning shows that the maximum number of edge disjoint paths between any pair of vertices in $G\backslash k$ is $n-2$.

Thus, for each pair of vertices in G, there is exactly one path that is forced to go through k, and there are $\binom{n-1}{2}$ paths considered in computing the load. □

We can similarly compute the load effect of any subset:

Proposition 3.10 (Load Effects in Complete Graphs)**.** *Let* $S \subset V(G\backslash k)$*. Then* $\varepsilon(S, k, K_n) = \binom{n-1-|S|}{2} - \binom{n-1}{2}$.

Proof. This proof follows because removing $|S|$ vertices from G yields $K_{n-|S|}$, and we can use the previous proposition to directly compute the load after removing the vertices. □

3.2 The Range of Load

When defining a metric like load it is natural to wonder about the range of values that the metric can take on. The degree of a vertex in a graph, for example, can only take on values between 0 and $n-1$. In this section we similarly begin to characterize the range of values the load can take on.

It is straightforward to show that the load must always be at least zero (and that there do exist graphs where the load is zero); that the load is nonnegative follows from its definition:

$$\begin{aligned} L(k, G) &= \sum_{u,v \neq k} n_{u,v}(G) - \sum_{u,v \neq k} n_{u,v}(G\backslash k) \\ &= \sum_{u,v \neq k} (n_{u,v}(G) - n_{u,v}(G\backslash k)). \end{aligned}$$

Between any pair of vertices u and v, there are at least as many edge disjoint paths in G as there are in $G\backslash k$, and so $n_{u,v}(G) - n_{u,v}(G\backslash k) \geq 0$. Hence all terms in the sum defining load are nonnegative and the entire sum is as well.

In addition, this bound is best possible. The trivial graph with n vertices and no edges clearly has load zero: there are no paths between any pair of vertices in that graph. Moreover, the load can still be zero in connected graphs. For example, we saw that the load on one of the endpoints (v_1 or v_n) of a path graph is zero. This information is summarized in the following proposition:

Proposition 3.11 (Lower Bound on Load)**.** *In any vertex v of a connected graph G, $L(v, G) \geq 0$ and this bound is tight.*

We will use the word *tight* to denote when a bound is best possible. A lower bound is tight if it is both a valid lower bound and no larger lower bound can exist, and a tight upper bound is defined analogously. One way to show that a bound is tight is to show that it is both a valid bound and that it is achieved. For example, the lower bound in Proposition 3.11 is tight because we argued that it was a lower bound and that it was achieved by the endpoints of a path graph.

3.2.1 Bounding the Maximum Load from Above

Constructing a tight upper bound for the load is more difficult. In what remains, we will show that the load on any vertex in any graph is at most of order n^3. Here the phrase *of order* n^3 means that the maximum possible load is bounded above by some constant multiple of n^3. We will also show the stronger statement that the maximum possible load grows asymptotically with n^3. More formally:

Definition 3.1 (Order)**.** *A function $f(n)$ is of **order** $g(n)$ if there exists some constant c_1 such that, for all sufficiently large n, $f(n) \leq c_1 g(n)$. We use big-O notation and write $f \in O(g)$. If, for sufficiently large n, we also have $f(n) \geq c_2 g(n)$, then we use big-Θ notation and write $f \in \Theta(g)$.*

The qualifier that a property holds "for sufficiently large n" means that there exists some natural number N such that, for $n > N$, the property holds. We will show that load is in $\Theta(n^3)$. We are also interested in the constants c_1 and c_2 and we will show that $c_1 \leq \frac{1}{4}$ and that $c_2 \geq \frac{1}{8} - \epsilon$ (for any $\epsilon > 0$). We will not be able to construct a tight upper bound of $c_1 n^3$, but we will be able to tightly show that the upper bound is $\Theta(n^3)$.

Note that, when we consider the order to be something like n^3, smaller terms like $c_3 n^2$, $c_4 n$ and c_5 become negligible (where the c_i are constants). When we want to show that the load is of order n^3, we might bound it as $c_1 n^3 + O(n^2)$. This notation means that we are effectively disregarding terms that negligibly contribute to the load as n grows.

To tightly show that the maximum possible load on any vertex in any graph of size n is $\Theta(n^3)$, we first bound the load from above as being at most n^3. Next, we construct a graph and identify a vertex for which the load on that vertex is approximately $c_2 n^3$. Though we will not show that this vertex has as high as possible of a load among all graphs of size n, we will show that it has the highest load in an infinite family of graphs (and namely, in a family of graphs with high load). These steps are handled separately below.

3.2.2 The Load on any Vertex in any Graph is at most $O(n^3)$

Proposition 3.12 (Upper Bound). *Let $|V(G)| = n$. Then the load on any vertex k in G is bounded above by $\frac{1}{4}n^3$.*

Proof. Using the definition of load,

$$L(k, G) = \sum_{u,v \neq k} n_{u,v}(G) - \sum_{u,v \neq k} n_{u,v}(G \backslash k)$$

$$= \sum_{u,v \neq k} (n_{u,v}(G) - n_{u,v}(G \backslash k)).$$

Consider any pair of vertices $u, v \neq k$. Let $l_{u,v}(k)$ denote the number of edge disjoint u-v paths that include k that contribute to $n_{u,v}(G)$. Any path from u to v that includes k must use two edges incident to k. k has at most $n-1$ neighbors, so there are at most $\frac{n-1}{2}$ edge disjoint u-v paths that contain k. That is, $l_{u,v}(k) \leq \frac{n-1}{2}$.

Note also that

$$n_{u,v}(G \backslash k) \geq n_{u,v}(G) - l_{u,v}(k).$$

This statement follows because there are at least $n_{u,v}(G) - l_{u,v}(k)$ edge disjoint paths from u to v in $G \backslash k$. Namely, the paths counted by $n_{u,v}(G)$ but not $l_{u,v}(k)$. Rearranging the formula and using our bound for l yields:

$$n_{u,v}(G) - n_{u,v}(G \backslash k) \leq l_{u,v}(k) \leq \frac{n-1}{2}.$$

Then:

$$L(k, G) = \sum_{u,v \neq k} (n_{u,v}(G) - n_{u,v}(G \backslash k))$$

$$\leq \sum_{u,v \neq k} \frac{n-1}{2}$$

$$= \binom{n-1}{2} \frac{n-1}{2}.$$

Finally,

$$\binom{n-1}{2}\frac{n-1}{2} = \frac{1}{4}(n-1)(n-1)(n-2) \leq \frac{1}{4}n^3.$$

□

Thus, the maximum possible value that the load can take on is $\frac{1}{4}n^3$. In particular, it is of order n^3 and the constant c_1 used above is at most $\frac{1}{4}$. Below we will conjecture that the constant is in fact $\frac{1}{8}$.

3.2.3 A Graph for which the Load Is $\Theta(n^3)$

Proposition 3.13 (The Maximum Load is at Least $c_2 n^3$.)**.** *Let $|V(G)| = n \geq 5$ Then there exist graphs with a key vertex whose load is $\frac{1}{8}n^3 - O(n^2)$.*

Proof. First suppose that n is odd so that $n = 2m - 1$. Then consider the graph G obtained by taking two complete graphs on m vertices and overlaying them so that they share a single vertex. This single vertex will be our key vertex, and this process is illustrated below.

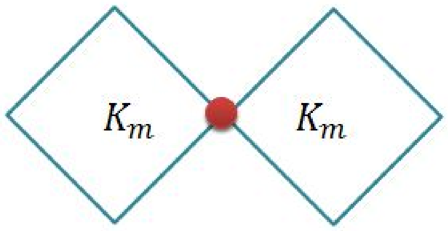


Figure 3.2 The load on the shared, red vertex is $O(n^3)$.

To compute the load we first compute $\sum_{u,v\neq k} n_{u,v}(G)$. For any $u, v \neq k$, both u and v have degree $m-1$. Thus $n_{u,v}(G) \leq m-1$. If u, v are in the same component of $G\backslash k$, then $m-1$ edge disjoint paths can be found in G: consider the edge between u and v as well as the $m-2$ paths of length two of the form u, w, v (where w is in the same component of $G\backslash k$ as u and v). Hence, if u, v are in the same component of $G\backslash k$, $n_{u,v} = m-1$.

If u and v are in different components of $G\backslash k$, we can still find $m-1$ paths. One path is of the form u, k, v. The remaining $m-2$ paths are of the form u, u', k, v', v where u' and v' are respectively in the same component of of $G\backslash k$ as u and v. These paths correspond to starting at u, moving to another vertex u' in the same component of of $G\backslash k$ to take a new edge to k, then moving from k along a new edge to v' in the same component of of

$G \backslash k$, and finally moving from v' to v. Again, and therefore in all cases, we have $n_{u,v} = m - 1$.

Since there are $n - 1 = 2m - 2$ non-key vertices we can compute the first term in our formula for the load:

$$\begin{aligned}
\sum_{u,v \neq k} n_{u,v}(G) &= \sum_{u,v \neq k} (m-1) \\
&= \binom{2m-2}{2}(m-1) \\
&= (m-1)(2m-3)(m-1).
\end{aligned}$$

Next, consider the graph $G \backslash k$. This graph has two components, each of which is a complete graph on $m - 1$ vertices. The only pairs of vertices for whom edge disjoint paths exist are those in the same component. In a K_{m-1}, there are $\binom{m-1}{2}$ pairs of non-key vertices, and there are at most $m - 2$ edge disjoint paths between each pair (as the degree of each vertex is $m - 2$). We can identify $m - 2$ such paths: between u, v in the same component of $G \backslash k$, there is the path obtained by taking the edge from u to v and there are $m - 3$ paths of the form u, w, v (for each of the $m - 3$ other vertices w in the same component). Thus, $n_{u,v}(G \backslash k)$ equals zero if u, v are in different components and $m - 2$ otherwise. We have two equivalent components and so:

$$\begin{aligned}
\sum_{u,v \neq k} n_{u,v}(G \backslash k) &= 2 \sum_{\substack{u,v \text{ in the} \\ \text{first component}}} n_{u,v}(G \backslash k) \\
&= 2 \sum_{\substack{u,v \text{ in the} \\ \text{first component}}} (m-2) \\
&= 2\binom{m-1}{2} m - 2.
\end{aligned}$$

We can simplify the right expression to $(m-1)(m-2)(m-2)$. Therefore the load is:

$$\begin{aligned}
L(k, G) = \sum_{u,v \neq k} n_{u,v}(G) - \sum_{u,v \neq k} n_{u,v}(G \backslash k) \\
= (m-1)(2m-3)(m-1) - (m-1)(m-2)(m-2).
\end{aligned}$$

Substituting in for $m = \frac{n+1}{2}$ and simplifying, we obtain that the load on the key vertex of this graph is:

$$L(k, G) = \frac{1}{8}(n^3 - n^2 - 5n + 5) = \frac{1}{8}n^3 - O(n^2).$$

If n is even, consider making a graph on $n-1$ vertices as described above and adding a single isolated vertex. This graph will still have a load of $\frac{1}{8}n^3 - O(n^2)$. □

By putting together Propositions 3.12 and 3.13, we obtain the following theorem:

Theorem 3.1 (Max Load is $\Theta(n^3)$)**.** *Let $|V(G)| = n$. Then the maximum possible load on a graph is bounded between $\frac{1}{8}n^3 - O(n^2)$ and $\frac{1}{4}n^3$. In particular, it is $\Theta(n^3)$ with a constant between $\frac{1}{8}$ and $\frac{1}{4}$.*

Proof. This result follows directly from the previous two propositions. □

3.2.4 A Conjecture about the Maximum Load

While we have only been able to bound the maximum possible load by $\frac{1}{4}n^3$, we conjecture that the load on the vertex used in Proposition 3.13 achieves the maximum load. That is:

Conjecture 3.1 (Conjectured Max Load)**.** *The maximum possible load, on any vertex in any graph, is $\frac{1}{8}n^3 - O(n^2)$.*

This statement remains a conjecture, but below we present intuition for why it might be true. We also present a template for proving the conjecture.

Intuitively, the shared vertex k in the graph in Figure 3.2 has a very large amount of flow through it. Using our motivating example, if these vertices were airports, then the two components of $G \backslash k$ could be thought of as separate countries, with all flights between those countries going through a fixed airport. There are many distinct (i.e. edge-disjoint) ways to fly between a pair of airports, but all routes between countries must pass through the fixed airport k. It might seem like the load on a fixed vertex would be as high as possible in these types of graphs where all the flow between two large sets of vertices had to pass through a single vertex.

We have generated data on the load on all vertices in all graphs of size up to $n = 7$, and for these small graphs it seems that the load is maximized when the structure is of that in Proposition 3.13. Looking at the load on all possible vertices in all possible graphs of some size is computationally intensive, and we have not generated data on any larger graphs. However, the limited data we have suggests that the maximum load on a vertex k in a graph G is as large as possible when $G\backslash k$ is disconnected and consists of two complete graphs of equal size.

One might then wonder if it is optimal, in general, to have $G\backslash k$ be disconnected into two components (and moreover, into two components of equal size rather than some other number of components of potentially different sizes). We will now show that neither of these alternative options are preferable: if you suppose that $G\backslash k$ is disconnected into complete components (while G is connected), then the load on k is at most $\frac{1}{8}n^3 + O(n^2)$.

Theorem 3.2 (Proving Conjecture 3.1 in a Special Case)**.** *Let k be a vertex in a connected graph G such that $G\backslash k$ is disconnected into several complete components. The load on k is at most $\frac{1}{8}n^3 + O(n^2)$.*

Proof. The main part of this proof will proceed using induction but requires first introducing some new notation. Suppose that $G\backslash k$ has j components. We will say that the i-th component has n_i vertices in it and we let $\alpha_i = \frac{n_i}{n-1}$. For convenience, suppose we have indexed our components so that $0 \leq n_j \leq \ldots \leq n_2 \leq n_1$. Note that $\sum_i n_i = n - 1$.

This proof will require several steps and we will break it into separate claims below.

Claim 3.1. *If each component is a complete graph, then the load on k is*

$$\sum_{1\leq i<l\leq j} n_i n_l^2 + \sum_{1\leq i\leq j} \binom{n_i}{2}.$$

Proof. First, note that;

$$\begin{aligned}
L(k,G) &= \sum_{u,v\neq k} n_{u,v}(G) - \sum_{u,v\neq k} n_{u,v}(G\backslash k) \\
&= \sum_{\substack{u,v\neq k \\ u,v \text{ in different} \\ \text{components}}} n_{u,v}(G) + \sum_{\substack{u,v\neq k \\ u,v \text{ in same} \\ \text{component}}} (n_{u,v}(G) - n_{u,v}(G\backslash k)) \\
&= \sum_{1\leq i<l\leq j} n_i n_l^2 + \sum_{1\leq i\leq j} \left(\binom{n_i}{2} n_i - \binom{n_i}{2}(n_i - 1) \right).
\end{aligned}$$

In the last line, the first summand can be found by considering all possible $n_i n_l$ pairs of vertices between the i-th and l-th components. For each pair of vertices, say u in the i-th component and v in the l-th, there are n_l edge-disjoint paths between them. First, there is the path from v to k to u. In addition, since $n_l \leq n_i$ and each component forms a complete graph, you can move from v to any of the other $n_l - 1$ vertices in the l-th component, then through k to any vertex in the i-th component, and finally, from that vertex to u.

The second summand follows because, in the i-th component, there are $\binom{n_i}{2}$ pairs of vertices. This component is a complete graph on n_i vertices and so there are $n_i - 1$ paths between any pair of these vertices. There is an additional path, only in G, between vertices u and v by taking the edge from u to k, then the edge from k to v. □

Claim 3.2. *The term*

$$\sum_{1 \leq i \leq k} \binom{n_i}{2}$$

is at most n^2 *and so we are only concerned with bounding*

$$\sum_{1 \leq i < j \leq k} n_i n_j^2.$$

Proof. Note that

$$\sum_{1 \leq i \leq k} \binom{n_i}{2} = \frac{1}{2} \left(n_1(n_1 - 1) + \ldots + n_j(n_j - 1) \right)$$

$$\leq \frac{\sum_{i=1}^{j} n_i^2}{2}.$$

We also note that $\frac{\sum_{i=1}^{j} n_i^2}{2}$ is at most $(n-1)^2/2$. To see this, write $n - 1 = n_1 + n_2 + \ldots + n_j$. Thus

$$(n-1)^2 = (n_1 + \ldots + n_j)(n_1 + \ldots n_j)$$

$$= \sum_i n_i^2 + \sum_{i \neq l} n_i n_l.$$

Since $n_i \geq 0$, we have that

$$(n-1)^2 \geq \sum_i n_i^2,$$

and the claim follows. □

Remark 3.1. *As a consequence of the previous claim, the only term that is potentially $O(n^3)$ in our sum is*

$$\sum_{1\leq i<l\leq j} n_i n_l^2.$$

In what follows we will use induction to bound this term by $\frac{1}{8}n^3$. To do so, first note that

$$\sum_{1\leq i<l\leq j} n_i n_l^2 = (n-1)^3 \sum_{1\leq i<l\leq j} \alpha_i \alpha_l^2.$$

We will thus concern ourselves with bounding

$$\sum_{1\leq i<l\leq j} \alpha_i \alpha_l^2.$$

Claim 3.3. *Suppose $j > 2$. Then*

$$\sum_{1\leq i<l\leq j} \alpha_i \alpha_l^2 \leq \frac{1}{8}.$$

Again, assume that $\sum_{i=1}^{j} \alpha_i = 1$ and $\alpha_i \geq 0$.

Proof. We prove this by induction. For the base case let $j = 2$. Then we want to show that

$$\sum_{1\leq i<l\leq j} \alpha_i \alpha_l^2 = \alpha_1 \alpha_2^2 \leq \frac{1}{8}.$$

Because $\alpha_1 = 1 - \alpha_2$, we want to maximize $(1-\alpha_2)\alpha_2^2$. Because $\alpha_1 \geq \alpha_2$ and $\alpha_1 + \alpha_2 = 1$, we have that $0 \leq \alpha_2 \leq \frac{1}{2}$. Setting the derivative of $(1-\alpha_2)\alpha_2^2$ with respect to α_2 equal to zero, we find that the maximum value occurs at the endpoints of the interval where $\alpha_1 = \alpha_2 = \frac{1}{2}$, in which case $\alpha_1\alpha_2^2 = \frac{1}{8}$. Hence, $\alpha_1\alpha_2^2 \leq \frac{1}{8}$, completing our base case. Note also that this shows that, if $G\backslash k$ has two complete components, then both components will have half of the vertices to maximize the load on k.

Now we continue on to the inductive step. We will assume that $\sum_{1\leq i<l\leq j} \alpha_i\alpha_l^2 \leq \frac{1}{8}$ for any α_i such that $\alpha_i \geq 0$ and $\sum_{i=1}^{j} \alpha_i = 1$. We will use this to show that $\sum_{1\leq i<l\leq j+1} \alpha_i\alpha_l^2 \leq \frac{1}{8}$ when $\sum_{i=1}^{j+1} \alpha_i = 1$. Note that:

$$\sum_{1\leq i<l\leq j+1} \alpha_i\alpha_l^2 = \sum_{1\leq i<l\leq j} \alpha_i\alpha_l^2 + \left(\sum_{i=1}^{j} \alpha_i\alpha_{j+1}^2\right). \tag{3.1}$$

Now set $\alpha'_i = \frac{\alpha_i}{1-\alpha_{j+1}}$. Then, since $\sum_{i=1}^{j} \alpha_i = 1 - \alpha_{j+1}$, we have that $\sum_{i=1}^{j} \alpha'_i = \sum_{i=1}^{j} \frac{\alpha_i}{1-\alpha_{j+1}} = 1$. Thus $\alpha'_1, \ldots, \alpha'_j$ satisfy the conditions of our inductive hypothesis and so

$$\sum_{1 \le i < l \le j} \alpha'_i \alpha'^2_j \le \frac{1}{8}.$$

This implies that

$$\frac{1}{(1-\alpha_{j+1})^3} \sum_{1 \le i < l \le j} \alpha_i \alpha_j^2 = \sum_{1 \le i < l \le j} \alpha'_i \alpha'^2_j \le \frac{1}{8}.$$

Returning to Equation 3.1, we have:

$$\sum_{1 \le i < l \le j+1} \alpha_i \alpha_l^2 = \sum_{1 \le i < l \le j} \alpha_i \alpha_l^2 + \left(\sum_{i=1}^{j} \alpha_i \alpha_{j+1}^2 \right)$$

$$\le \frac{(1-a_{j+1})^3}{8} + \alpha_{j+1}^2 \sum_{i=1}^{j} \alpha_i.$$

Since $\sum_{i=1}^{j} \alpha_i = 1 - \alpha_{j+1}$:

$$= \frac{(1-a_{j+1})^3}{8} + \alpha_{j+1}^2 (1 - \alpha_{j+1}).$$

As previously, we note that $\alpha_{j+1} \le \frac{1}{j+1} \le \frac{1}{3}$. With this constraint, the maximum value of the load is when $\alpha_{j+1} = 0$, in which case the maximum value is $\frac{1}{8}$. Hence

$$\sum_{1 \le i < l \le j+1} \alpha_i \alpha_l^2 \le \frac{1}{8}.$$

□

This completes both the proof of our final claim and the proof of the entire theorem. That is, we have shown that the load, on a graph consisting of several complete components sharing a single vertex, is at most $\frac{1}{8}n^3 + O(n^2)$. The $O(n^2)$ potentially comes from the term $\sum_{1 \le i \le k} \binom{n_i}{k}$ in our second claim. □

The preceding theorem is a bit complicated, and so we will highlight the following:

Corollary 3.1 (Load when Removing k Disconnects G)**.** *If G is a connected graph such that $G\backslash k$ is disconnected into complete components, then the load on k is asymptotically as high as possible when $G\backslash k$ is separated into two components of equal size.*

Proof. We achieve a load of $\frac{1}{8}n^3 - O(n^2)$ in this case, and above we have argued that you can never get a better bound on the constant in front of the n^3 term. □

Corollary 3.2 (The Structure of Potential Counterexamples to Conjecture 3.1)**.** *If there exists a graph G on n vertices with some vertex k such that $L(k, G) > \frac{1}{8}n^3 + O(n^2)$, then $G\backslash k$ does not consist of several complete components sharing a single vertex.*

In other words, the result above says that, to prove our conjecture and show that an upper bound on the load of $\frac{1}{8}n^3 + O(n^2)$ is tight, we need only consider graphs such that $G\backslash k$ is connected or that its components must not be complete. One might be able to separately consider these cases and exploit the additional information to show the bound of $\frac{1}{8}n^3 + O(n^2)$ is tight.

Lastly:

Remark 3.2. *The load on a key vertex seems to be quite large when $G\backslash k$ is disconnected. One possible approach to increasing the load on k could be to remove some subset of vertices S such that $G\backslash S$ is connected but $G\backslash(S \cup \{k\})$ is not.*

3.2.5 One Way to Prove Conjecture 3.1

We conclude this section by mapping a proof of Conjecture 3.1. To prove our conjecture, it suffices to prove the following:

Conjecture 3.2 (Load Between Vertices in the Same Component)**.** *The contributions to the load of pairs of vertices in the same component of $G\backslash k$ is $O(n^2)$.*

Note that this agrees with what we saw in every connected graph studied in the previous section.

Proposition 3.14 (Conjecture 3.1 Follows from Conjecture 3.2)**.** *If Conjecture 3.2 holds, then the maximum load is $\frac{1}{8}n^3 + O(n^2)$.*

Proof. At the beginning of the argument, we considered

$$\begin{aligned} L(k,G) &= \sum_{u,v\neq k} n_{u,v}(G) - \sum_{u,v\neq k} n_{u,v}(G\backslash k) \\ &= \sum_{\substack{u,v\neq k \\ u,v \text{ in different} \\ \text{components}}} n_{u,v}(G) + \sum_{\substack{u,v\neq k \\ u,v \text{ in same} \\ \text{component}}} (n_{u,v}(G) - n_{u,v}(G\backslash k). \end{aligned}$$

We were able to use the fact that each component of our graph was a complete graph to bound the second sum as $O(n^2)$. As long as we can make that step, the rest of our proof will follow logically. In this case it would show that the maximum load is $\frac{1}{8}n^3 + O(n^2)$ on any graph where $G\backslash k$ consists of at least two components.

We could, however, also modify our argument so that it applies to graphs where $G\backslash k$ is connected: let G be a graph such that $G\backslash k$ is connected. Let G' be the graph obtained by adding a single vertex u and connecting it only to k. Then:

$$L(k,G') = L(k,G) + (n-1).$$

This follows because u is of degree one and cannot be used as an internal vertex in any path. Hence, the only difference between $L(k,G)$ and $L(k,G')$ will be what comes from the paths that use u as an endpoint. Assuming that G is connected, each vertex will have exactly one path to u and this path must go through k. There are $n-1$ other vertices in $V(G)\backslash k$, and so the load on k in G' will be exactly $n-1$ more than the load on k in G.

However, $G'\backslash k$ will consist of two components: $G\backslash k$ and u. Thus, $L(k,G') \leq \frac{1}{8}n^3 + O(n^2)$, and

$$L(k,G) = L(k,G') - (n-1) \leq \frac{1}{8}n^3 + O(n^2).$$

□

One way to prove the above Conjecture 3.2 would be to consider any graph G such that $G\backslash k$ is disconnected, then show that the load changes by at most $O(n^2)$ when you make each component complete.

3.3 Structural Theorems

We now return to theorems that help characterize the load. Our first theorem states Proposition 3.1 more formally:

Theorem 3.3 (Vertices in Different Components)**.** *Let G be a graph, and let $S = \{v_1, ..., v_n\} \subset V(G)$ be such that $G \backslash S$ is disconnected. Further, let i be a vertex in a different component of $G \backslash S$ than k, and let $T = S \cup \{i\}$. Then $\varepsilon_k(G, T) = \varepsilon_k(G, S)$.*

Proof. Let G_k and G_i respectively denote the components of $G \backslash S$ containing k and i. Note that, since i is not in G_k, $G \backslash T$ will also contain G_k as a distinct component. Moreover, the only paths of flow that can contribute to the load on k are those that occur between vertices in G_k, as these are the only paths that can include k. Since the components containing k are identical in $G \backslash T$ and $G \backslash S$, the loads on k will be the same. The proof then follows from the remarks in the proof of Proposition 3.1. □

Next, we add a corollary to Theorem 2.3. This corollary adds another interpretation to Theorem 2.3, and can be viewed as adding a new member (with few connections) to the network. This could correspond to the CUT company adding a new airport to its network, but only offering flights between this new airport and one other destination (perhaps because customers in one region requested a specific route to an airport CUT had not previously operated out of). If FloW already determined which airport they wanted to buy, they might be interested in how much would change when CUT made this move.

Corollary 3.3 (Adding Vertices of Degree One)**.** *Adding a vertex v of degree one cannot decrease the load on a key vertex in any graph G.*

Proof. It is equivalent to consider the load effect of v on k in the graph G' which we define to be the graph after v has been added. Because v will be degree one, it will have a single edge disjoint path to k. The load effect of removing this vertex is nonpositive, which is equivalent to stating that adding this vertex cannot decrease the load. □

Moreover, while adding v can only increase the load, we can exactly characterize the change in load if v's single edge is incident to the key vertex k.

Proposition 3.15 (Adding Vertices of Degree One Adjacent to the Key Vertex)**.** *Let G be connected, let v be a vertex not in G. Let $G + v$ be the graph obtained by adding v to G and adding a single edge between v and a key vertex k. Then $L(G + v, k) = L(G, k) + n - 1$, where $n = |V(G)|$.*

Proof. This statement was proved in Proposition 3.14.

□

This proposition ends up providing an upper bound for the increase in load when we add a vertex of degree one anywhere in our graph:

Proposition 3.16 (Adding Vertices of Degree One Not Adjacent to the Key Vertex)**.** *Let G be connected, let v be a vertex not in G. Let $G + v$ be the graph obtained by adding v to G and adding a single edge between v and some other vertex u. Then $L(G + v, k) \leq L(G, k) + n - 1$, where $n = |V(G)|$. Further, if $u \neq k$ then this inequality is strict.*

Proof. Since v has degree one, it cannot be used as an internal vertex in any path. Thus, the only change in load will be due to the new paths which have v as one endpoint. There are $n - 1$ such paths ending with v and some other vertex $u \neq k$ in G, which may or may not go through k. Thus, we can add at most $n - 1$ to the load. If v is not adjacent to k, at least one of these paths (namely, the path of length one between v and v's neighbor) will not go through k and the inequality is strict. □

The preceding two propositions give us insight into the potential effect of adding (or removing) a single vertex of degree one from a graph, and it is natural to ask what happens when we add vertices of larger degree. As soon as we consider vertices of degree two, however, what we say becomes much less specific:

Proposition 3.17 (Adding Vertices of Degree Two)**.** *Let G be a graph with key vertex k. Let $G + v$ be the graph obtained by adding a single vertex of degree two to G. Then*

$$L(k, G) - \binom{n}{2} \leq L(k, G + v) \leq L(k, G) + \binom{n}{2},$$

where $n = |V(G)|$.

Proof. The preceding proposition follows because v has degree two, and thus can add at most one edge disjoint path between any pair of vertices. It can add a path that must go through k, increasing the load by one for each pair of vertices. Alternately, it can be used to reroute a single path that was previously forced to go through k, decreasing the load by one for each pair of vertices. Hence the load on k in $G + v$ can be either greater or less than the load on k in G, but it always must be within $\binom{n}{2}$ of the load on k in G. □

Remark 3.3. *The bounds in Proposition 3.16 are optimal in the sense that there exist graphs such that, as n goes to infinity, $L(k, G + v)$ asymptotically approaches*

$L(k, G) + \binom{n}{2}$ and there exist graphs such that $L(k, G + v)$ asymptotically approaches $L(k, G) - \binom{n}{2}$.

In the case where $G + v$ has a higher load on k than G, consider a path graph P_n where k is one of the endpoints. Adding a vertex of degree two by connecting it to the two endpoints of the path creates a cycle on $n + 1$ vertices, increasing the load on k from 0 to $\binom{n}{2}$.

To see that graphs exist such that $L(k, G + v)$ asymptotically approaches $L(k, G) - \binom{n}{2}$, consider G to be the cycle C_n. Add v so that it is adjacent to k's two neighbors, creating a detour around k. The load decreases from $\binom{n-1}{2}$ to 1.

3.4 Random Graphs

A final direction of research has been to analyze the properties of random graphs. Empirically, Martonosi et al. (2011) found that if k is selected to be a highly central vertex, there are almost always vertices whose removal increases the load on k in certain types of random graphs. There are many different models for constructing random graphs, each with its own parameters. Martonosi et al. (2011) considered multiple such models, and in each model and with each choice of parameters they used, they almost always found vertices whose removal increased the load on a highly central vertex.

For our purposes, a highly central vertex will be a vertex with high degree, betweenness and closeness centrality. We can show, however, that no matter how large n is, there exist graphs with highly central key vertices such that there is no way to increase the load on the key vertex. Moreover, these graphs need not be trivial in the sense that they can have arbitrarily many edges, and the key vertex can be chosen so that it is the unique vertex with the highest degree, betweenness and closeness centrality.

We show this result in the following theorem, but first clarify what we mean by there exist graphs with a desired property and "arbitrarily many edges." We do not just mean that for every M, there exists a graph G having $m > M$ edges that has the desired property. Instead, we will make a stronger statement and consider the **edge density**. The edge density is the proportion of all possible edges in the graph, namely

$$\frac{|E(G)|}{\binom{n}{2}},$$

where $n = |V(G)|$. The edge density is then between 0 and 1. When we say there exists a graph with "arbitrarily many edges" we mean that there exists a graph whose edge density is arbitrarily close to 1 so that the proportion of edges it has is as high as possible.

Theorem 3.4 (Central Vertices whose Load Cannot be Increased)**.** *Let $\epsilon > 0$ be given. There exists a connected graph with edge density at least $1 - \epsilon$ such that no vertex can be removed to increase the load on the vertex with the highest betweenness, closeness and degree centrality k.*

Proof. Let $n > \frac{3}{\epsilon}$ and consider the graph G_n on n vertices obtained as follows: start with a K_{n-1}, select one vertex k from the graph, then add an n-th vertex v by connecting it directly to k. Note that $d(v) = 1$, $d(k) = n - 1$, and $d(u) = n - 2$ for all other u in $V(G_n)$. Such a graph is shown below, in Figure 3.3, for $n = 5$.

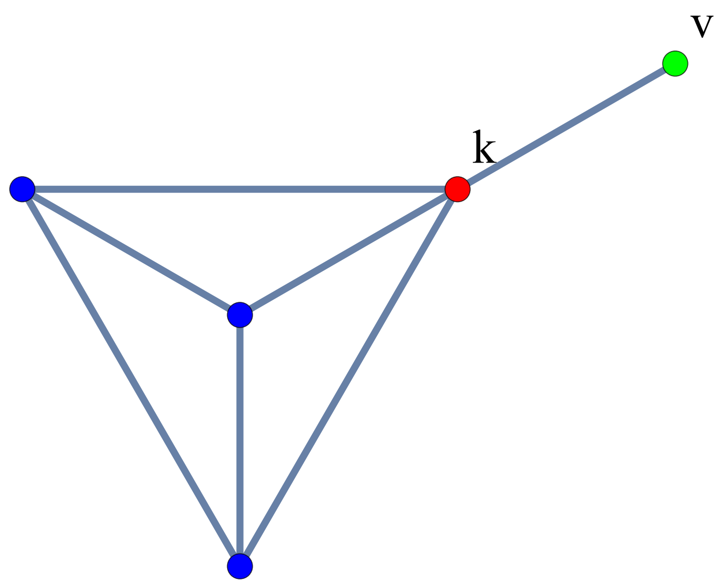


Figure 3.3 Example of a graph used in Theorem 3.4 for $n = 5$. The red vertex is the vertex chosen to be k and the green vertex is the vertex v added in the proof. This example is of G_5.

Our proof contains three main steps. We begin by showing that k has the highest betweenness, closeness and degree centrality. Next we compute the load on k in G_n and show that it cannot be increased by removing any subset of vertices. Finally, we compute the edge density, which approaches one as n goes to infinity.

First, we claim that k is the most central vertex of G_n. The remarks above suffice to show it has the highest degree centrality. It also has the highest closeness centrality: this follows because it is a distance one away from every other vertex in the graph, while every other vertex is a distance two away from at least one vertex (any $u \neq k$ in the K_{n-1} is distance two away from v.) Finally, k also has the highest betweenness centrality. This statement follows because k is the only vertex appearing in the middle of a geodesic (every other vertex only appears as the endpoints of geodesics).

Next, note that the load on k is $\binom{n-2}{2} + n - 2$. This follows because the added edge $\{v, e\}$ cannot be used in any paths that do not contain v as an endpoint and so v cannot be a internal vertex. We then have $n-2$ paths between v and vertices other than k in the K_{n-1}, and all of these paths will not be present in $G\backslash k$. We also must consider all paths with endpoints u, w, both not equal to k or v. Since no other paths can use v as an internal vertex, we only need to examine paths between pairs of vertices in the original K_{n-1} that are entirely contained in the K_{n-1}. Using Proposition 3.9, the load contributions from these vertices will be $\binom{n-2}{2}$.

We now show that the load on k in G will be reduced by the removal of any subset of $V(G_n)$ (not including k). First, suppose that our subset contains v. Removing v reduces the graph to a K_{n-1}, so by Proposition 3.9, removing v will decrease the load on k to $\binom{n-2}{2}$. Removing any subset $S \subset V(G_n)$ including v (but not k) is equivalent to first removing v and the removing $S\backslash v$, which we define to be S'. The load on k in $G\backslash S$ is then equal to the load on k in $K_{n-1}\backslash S'$. Using the proof in Proposition 3.10, this is just $\binom{n-2-|S'|}{2}$, and since $|S'| \geq 0$:

$$\begin{aligned} L(k, G_n\backslash S) &= \binom{n-2-|S'|}{2} \\ &< \binom{n-2}{2} + n - 2 = L(k, G_n). \end{aligned}$$

Hence, any subset including v decreases the load.

Next, consider removing any subset $|S|$ that does not contain v (or k). Doing so reduces the graph G_n to $G_{n-|S|}$, which has load $\binom{n-2-|S|}{2} + n - 2 - |S|$. Thus:

$$\begin{aligned} L(k, G_n\backslash S) = L(k, G_{n-|S|}) &= \binom{n-2-|S|}{2} + n - 2 - |S| \\ &< \binom{n-2}{2} + n - 2 = L(k, G_n). \end{aligned}$$

Hence, any subset not including v also decreases the load. Since the removal of any nontrivial subset decreases the load on k, there is no way to increase the load on k.

Finally, the edge density, or the proportion of all possible $\binom{n}{2}$ edges, of this graph is

$$\frac{\binom{n-1}{2}+1}{\binom{n}{2}} = \frac{n^2-3n+4}{n^2-n}.$$

Note that:

$$\begin{aligned}
\frac{n^2-3n+4}{n^2-n} &\geq \frac{n^2-3n}{n^2} \\
&= \frac{n-3}{n} \\
&\geq \frac{\frac{3}{\epsilon}-3}{\frac{3}{\epsilon}} \\
&= \frac{3-3\epsilon}{3\epsilon} = 1-\epsilon.
\end{aligned}$$

Thus our edge density is at least $1-\epsilon$, completing our proof. $\square$

Chapter 4

Spectral Approaches and the Paint Spilling Problem

In our introduction we described spectral graph theory as a powerful field that studies the structural and combinatorial properties of graphs through the eigenvalues and eigenvectors of associated matrices. We have thus far been studying load as a combinatorial property, so one might hope that spectral graph theory may be a useful tool for studying load.

Moreover, spectral graph theory has previously been used to study other types of flow problems. Spielman (2010), for example, discusses problems involving liquid paint flow as well as electrical flow. Paint, electricity and communication ostensibly all flow differently, but one might hope that they flow in related ways. If different types of network flow are correlated, we might be able to develop heuristic approaches to LoMax that take advantage of flow problems and that can be solved more efficiently.

In this chapter, after introducing the matrices studied, we specifically focus on empirically connecting the paint flow problem to load. Here wet paint flows through a network and eventually dries, accumulating at vertices. We examine special vertices, where either an abnormally large or an abnormally small amount of paint dries. We specifically look to see if these special vertices are also special vertices in the LoMax problem (vertices we either want or do not want to remove). After presenting empirical data about these connections, we will consider a heuristic approach to LoMax motivated by them.

4.1 Spectral Graph Theory Background

In this section we briefly describe the matrices frequently studied in spectral graph theory. For more detail, the interested reader is referred to Spielman (2012), Spielman (2010) and Chapter 11 of Newman (2010). Spielman (2012) and Spielman (2010) are sets of lecture notes used by Professor Daniel Spielman, who teaches spectral graph theory at Yale University. These notes are accessible and thorough, and they describe many compelling applications of spectral graph theory.

To consider matrices that can be used to describe graphs we require some ordering on the vertices. Such an ordering will allow us to index the rows and columns of the matrices we will use. Any ordering can be used, but it must be used consistently. When we refer to the graph in Figure 2.1, we will assume that the vertices are ordered alphabetically. That is, Alice will be referenced as vertex 1, Bob, as vertex 2, etc.

Perhaps the most intuitive matrix is the **adjacency matrix** denoted A. It is the $n \times n$ matrix where $A_{i,j}$ is 1 if vertex i is adjacent to vertex j and zero otherwise. The adjacency matrix for Figure 2.1 is:

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

Note that the matrix is symmetric and there are zeros along the diagonal. Since there is an edge between Alice and Bob, the adjacency matrix has a 1 in both the first row, second column and the second row, first column. This matrix allows us to determine which pairs of vertices are adjacent, and from the matrix we can uniquely recreate the graph. It is thus a natural way to store information about a graph.

Another common matrix is the **degree matrix**, D. This $n \times n$ matrix is diagonal, and $D_{i,i}$ is the degree of vertex i. In our example

$$D = \begin{pmatrix} 3 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 3 \end{pmatrix}.$$

One of the most ubiquitous matrices studied in spectral graph theory, and the final matrix we will discuss, is the Laplacian, denoted L. This matrix can be defined as $L := D - A$. The Laplacian, for the graph in Figure 2.1 is:

$$L = \begin{pmatrix} 3 & -1 & 0 & -1 & -1 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 3 & -1 & -1 \\ -1 & 0 & -1 & 2 & 0 \\ -1 & 0 & -1 & 0 & 3 \end{pmatrix}.$$

Though this matrix might at first appear unnatural, its eigenvalues and eigenvectors has been used to reveal information about the properties of graphs. These applications are discussed in detail in Spielman (2012).

Before discussing the paint spilling problem, we note that a natural question to first ask could be: do the eigenvalues and eigenvectors of matrices associated with graphs reveal anything about load (or the way load changes when vertices are removed)? Indeed, this is the first question we studied when considering spectral graph theory, and one direction of our research has been to generate data and to search for connections between eigenvalues, eigenvectors and load. While we are optimistic that such connections exist, we have not yet formally shown any. We will defer this discussion Chapter Five, where we will address open problems and briefly present preliminary data.

4.2 The Paint Spilling Problem

This problem, and the related mathematics, comes from Spielman (2010) and is discussed in the notes for the eleventh lecture. In this problem a bucket of wet paint is spilled throughout the network according to a specific initial distribution. The paint then flows and dries through the network. At each time step, some proportion of the paint at each vertex will dry, and the paint that remains wet will continue to flow through the network. The question of interest is: when all the paint is dry, how much paint has accumulated at each vertex?

Spielman (2010) starts by having all of the paint at a single vertex, but any initial distribution of the paint is appropriate. We can encode this distribution as a vector $\mathbf{w_0}$ associating to each vertex in the network the proportion of paint initially placed at that vertex. If we consider our original graph in Figure 2.1 having vertex set {Alice, Bob, Charlie, Dan, Key}, the vector

$$\mathbf{w_0} = (\frac{1}{2}, \frac{1}{4}, 0, 0, \frac{1}{4})^T$$

would correspond to spilling half of the paint on Alice's vertex, spilling a quarter of the paint on each of the vertices corresponding to Bob and Key, and spilling no paint on Charlie and Dan. Any vector $\mathbf{w_0} = (w_1, w_2, \ldots, w_n)$ will be appropriate so long as $\sum_{i=1}^{n} w_i = 1$ and $w_i \geq 0$.

At each time step, some proportion α of the paint at each vertex v will dry and become fixed to v. Half of the remaining wet paint will stay as wet paint at v, and the other half will flow uniformly to each neighbor of v. In general, $\mathbf{w_t}$ will be a vector recording the proportion of the paint that is wet remaining at each vertex.

An analogous vector $\mathbf{d_t}$ can be defined to record the proportion of paint that is dry at each vertex at time t. We are interested in

$$lim_{t\to\infty}\mathbf{d_t},$$

the amount of paint that has dried at each vertex when all of the paint has dried. In Spielman (2010) it is shown that:

$$lim_{t\to\infty}\mathbf{d_t} = \beta(I_n - (1-\beta)W)^{-1}\mathbf{w_0},$$

where $\beta = \frac{2\alpha}{1+\alpha}$, I_n is the $n \times n$ identity matrix, W is the walk matrix, equal to AD^{-1}, and A and D are the adjacency and degree matrices discussed above. For example, in the network in Figure 2.1, we have:

$$W = \begin{pmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \end{pmatrix}.$$

The i, j-th entry of W is then 0 if i and j are not adjacent vertices, and if $i \sim j$, then $W_{i,j} = \frac{1}{d_j}$.

With the paint spilling problem we have flexibility in choosing α and $\mathbf{w_0}$. For each choice of these parameters, the distribution of dried paint on the vertices, $lim_{t\to\infty}\mathbf{d_t}$, which we will refer to as $\mathbf{d}_\infty$, may vary. Some vertices will be left with either a particularly small or a particularly large proportion of the dried paint. In the next section we will present empirical data connecting these vertices to vertices that are highlighted in the LoMax problem. Before doing so, we briefly highlight the impact of varying the parameters.

4.2.1 The Impact of Varying α and $\mathbf{w_0}$

Figure 4.1 shows an 18 vertex graph on which we will run the paint spilling algorithm with different choices of α and $\mathbf{w_0}$. The vertex set is $\{1, 2, \ldots, 18\}$ and the vertices are labeled in the figure. We choose the key vertex to be the vertex with the highest closeness centrality: vertex 18. This vertex will be denoted with a flag instead of a circle.

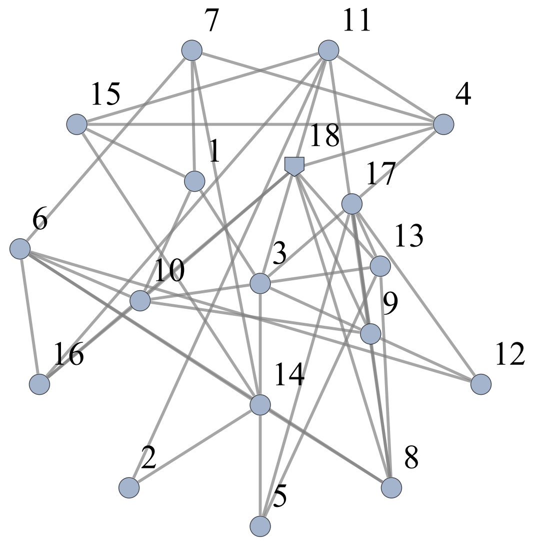


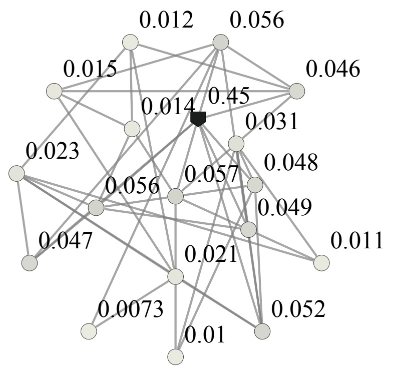
Figure 4.1 The original graph we will use to study the impact of varying parameters in the paint spilling problem. Each vertex is labeled according to the ordering imposed, and the key vertex is vertex 18.

We are interested in connections to the load on highly central vertices (like vertex 18). One might thus start by choosing to have all of the paint placed at the key vertex by setting

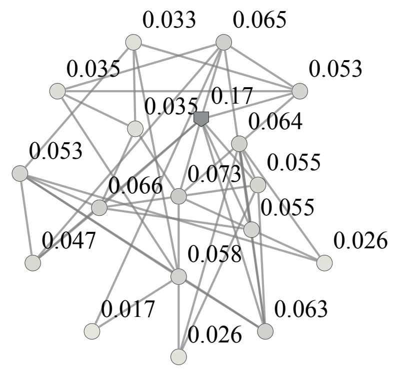
$$\mathbf{w_0^{(k)}} = (0, 0, 0, \ldots, 0, 1)^T.$$

Figures 4.2a, 4.2b and 4.2c show the final paint distribution with this choice of $\mathbf{w_0}$ respectively for $\alpha = 0.25, 0.05$ and 0.01. Each vertex in this figure is colored corresponding to how much paint accumulates at that vertex, with darker colors representing more accumulated paint. (The coloring is relative to the paint accumulated in all vertices in all six images).

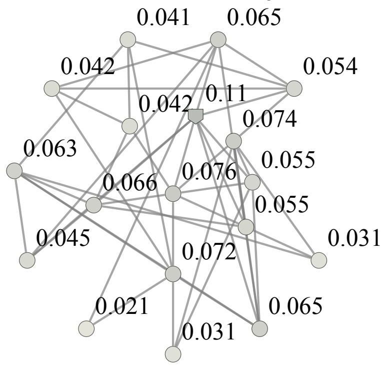
We can see that for a larger α, when paint is drying faster, the distribution of accumulated paint is more varied and biased towards being close



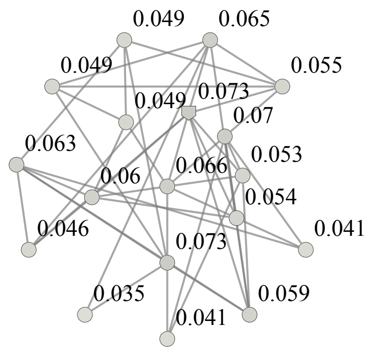
a. $\alpha = 0.25$, $\mathbf{w}_0^{(\mathbf{k})}$



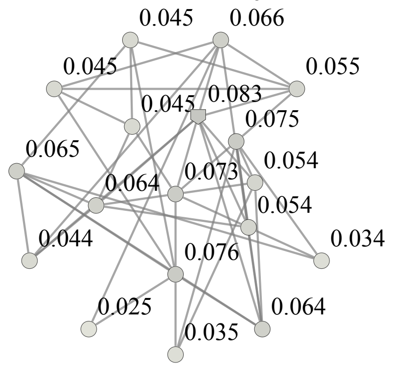
b. $\alpha = 0.05$, $\mathbf{w}_0^{(\mathbf{k})}$



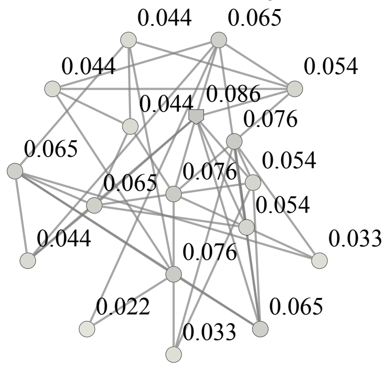
c. $\alpha = 0.01$, $\mathbf{w}_0^{(\mathbf{k})}$



d. $\alpha = 0.25$, $\mathbf{w}_0^{(\mathbf{u})}$



e. $\alpha = 0.05$, $\mathbf{w}_0^{(\mathbf{u})}$



f. $\alpha = 0.01$, $\mathbf{w}_0^{(\mathbf{u})}$

Figure 4.2 The accumulated paint at each vertex using different initial distributions and values of α. Numbers above each vertex represent the ultimate proportion of paint accumulated at each vertex and vertices are shaded according to these numbers over all six graphs, with darker shading reflecting larger proportions of dried paint. They key vertex is shown as a flag. Captions denote the initial distribution and value of α used.

to the key vertex. Note that the key vertex is guaranteed to accumulate at least a proportion of α in the first time step, which partially explains why so much paint is at the key vertex in 4.2a. Similarly, each vertex adjacent to the key vertex accumulates at least a proportion of $\frac{\alpha(1-\alpha)}{d(18)}$ in the next time step, where $d(18)$ is the degree of vertex 18.

As α decreases, the distribution is less varied and biased towards the key vertex. In all three cases, the key vertex, where paint was initially placed and which had the highest closeness centrality, accumulates the most paint. At the same time, it accumulates less paint as α decreases: when $\alpha = 0.25$, nearly 45% of the paint dries at vertex 18, but when $\alpha = 0.01$, only about 11% of the paint dries at that vertex.

Another possible starting distribution is to place the paint uniformly among the vertices, setting

$$\mathbf{w}_0^{(\mathbf{u})} = (\frac{1}{18}, \frac{1}{18}, \ldots, \frac{1}{18})^T.$$

It might seem disconcerting that this distribution, by itself, does not reflect any information about the key vertex. However, we typically consider the load on highly central vertices. The distribution $\mathbf{w}_0^{(\mathbf{u})}$ might correlate more with actual communication flow by allowing flow to begin at each vertex. Thus the vertices that accumulate a particularly large or a particularly small amount of paint in this distribution might correlate with places where communication flow "gets stuck" in the entire network. Figure 4.2 also shows how the paint accumulates with this initial distribution and the same values of α.

Note that Figures 4.2d, 4.2e and 4.2f seem more similar than 4.2a, 4.2b and 4.2c. Though vertices accumulate different amounts of paint, there is less bias towards being close to the key vertex. Similarly, the amount of paint at a given vertex, especially at vertices close to the key vertex, does not change as significantly with α.

To capture both information about the key vertex and overall communication flow, we might also consider taking the average of the previous initial distributions. This corresponds to placing half of the paint at the key vertex and distributing the remaining half of the paint uniformly among the vertices. Here

$$\mathbf{w}_0^{(\mathbf{a})} = \frac{1}{2}\mathbf{w}_0^{(\mathbf{k})} + \frac{1}{2}\mathbf{w}_0^{(\mathbf{u})}.$$

In this case,

$$\begin{aligned}
lim_{t\to\infty}\mathbf{d_t} &= \beta(I_n-(1-\beta)W)^{-1}\mathbf{w_0^{(a)}} \\
&= \beta(I_n-(1-\beta)W)^{-1}\frac{1}{2}\left(\mathbf{w_0^{(k)}}+\frac{1}{2}\mathbf{w_0^{(u)}}\right) \\
&= \frac{1}{2}\beta(I_n-(1-\beta)W)^{-1}\mathbf{w_0^{(k)}}+\frac{1}{2}\beta(I_n-(1-\beta)W)^{-1}\mathbf{w_0^{(u)}}.
\end{aligned}$$

Thus, the proportion of paint accumulated at each vertex with $\mathbf{w_0^{(a)}}$ and some choice of α is just the average of the paint accumulated at each vertex with starting distributions $\mathbf{w_0^{(k)}}$ and $\mathbf{w_0^{(u)}}$, using that choice of α.

Now that we have seen how the choice of initial parameters can affect the final paint distribution, we examine connections between load and the final paint distribution. In the next section we will consider larger graphs and will set $\alpha = 0.01$ so that the paint is drying slowly enough that it has time to flow through the network, but quickly enough so that it readily distinguishes between vertices. We will also use the distribution $\mathbf{w_0^{(k)}}$ that starts with all paint at the key vertex, though similar data can be generated using different values of α and different initial distributions.

4.3 Empirical Data

In this section we examine connections between accumulated paint and load effects in random graphs. We specifically consider *Erdős - Rényi random graphs* using the $G(n,p)$ model. This model creates an n vertex graph where each of the $\binom{n}{2}$ edges is included independently with probability p.

We specifically will generate m random graphs according to the above model. To create each graph, we randomly select a number of vertices $n \in [n_{min}, n_{max}]$ then use the $G(n, \frac{7}{n})$ model to create a graph. In each graph, we choose the key vertex to be the vertex with the highest closeness centrality. (The choice of $p = \frac{7}{n}$ means that np will be fixed to equal 7. Note that the expected degree of a vertex is $(n-1)p$ so that, for large n, fixing $np = 7$ effectively fixes the expected degree of each vertex to be close to 7. In Martonosi et al. (2011), the authors considered 100 vertex graphs with $p = 0.1$ so that $np = 10$. We are considering graphs on a range of vertices and will fix np rather than p so that vertices have similar degrees across all graphs we test. We primarily test graphs with approximately 70 vertices and choose $p = \frac{n}{7}$ so that our value of np is comparable to what was used in Martonosi et al. (2011).)

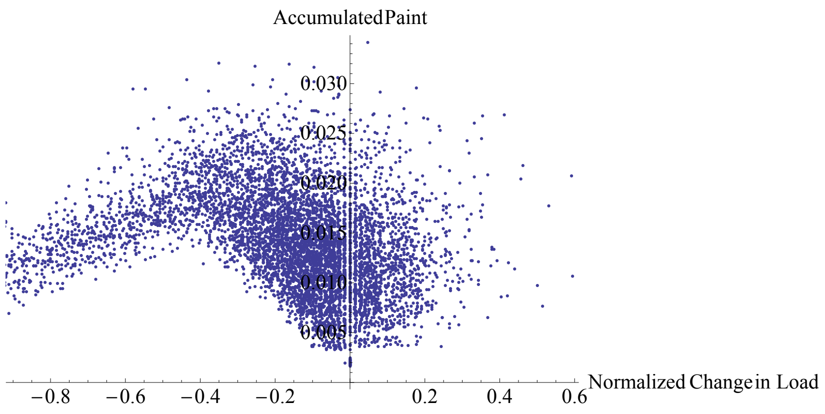


Figure 4.3 Plot of accumulated paint versus normalized change in load on one hundred 60-80 vertex graphs. The parameters were $n_{min} = 60$, $n_{max} = 80$, $m = 100$ and $\alpha = 0.01$.

Our algorithm will then take each graph and run the paint spilling problem with a chosen value of α. For each non-key vertex v in each graph, it plots the accumulated paint at that vertex against the normalized change in load when v is removed (where we normalize by dividing the change in load by n).

We can use the above algorithm to compare the accumulated paint at each vertex to the load effect of that vertex on a highly central vertex (the key vertex) over many graphs. In Figure 4.3, we consider vertices in 100 graphs having between 60 and 80 vertices. Again, we use $\alpha = 0.01$ and $\mathbf{w}_{\mathbf{0}}^{(\mathbf{k})}$. In the figure we see an interesting shape. In general, removing vertices with more accumulated paint tends to decrease the load. The removal of vertices that accumulate little paint tends to either increase or decrease the load substantially.

This trend also can be observed in graphs on a narrower range of vertices and when run on a single graph. Figure 4.4 considers 100 graphs, each of which is constructed on 70 vertices. Figure 4.5 shows the results from a single 200 vertex graph. We see analogous trends, especially in Figure 4.4.

Though the above only considers the load effect of single vertices, it suggests that vertices where a large amount of paint accumulates might also tend to be vertices that we do not want to remove. Intuitively, these vertices might be important for routing communication in a network, and removing them does not tend to reroute communication through a highly central vertex. We conclude this chapter by using this idea to develop a heuristic.

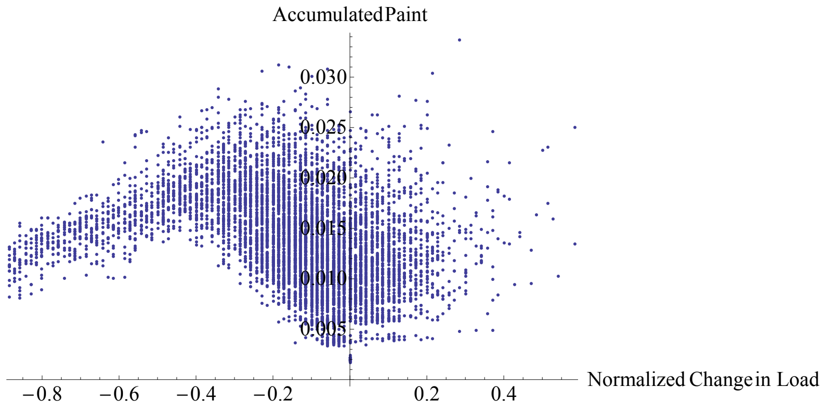


Figure 4.4 Plot of accumulated paint versus normalized change in load on one hundred 70 vertex graphs. The parameters were $n_{min} = 70$, $n_{max} = 70$, $m = 100$ and $\alpha = 0.01$.

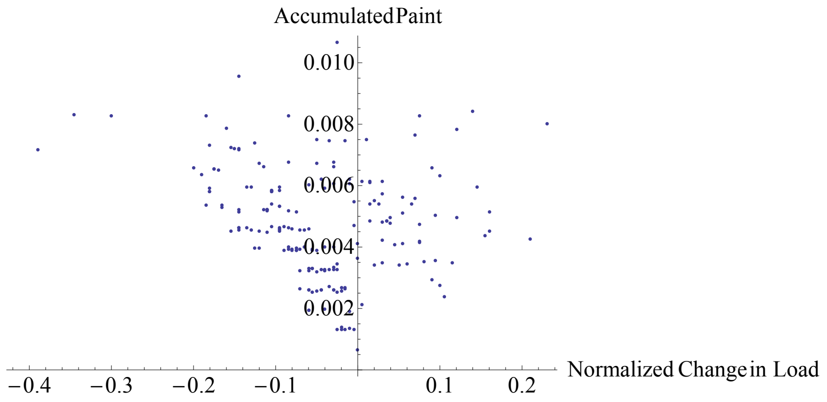


Figure 4.5 Plot of accumulated paint versus normalized change in load on a single 200 vertex graph. The parameters were $n_{min} = 200$, $n_{max} = 200$, $m = 1$ and $\alpha = 0.01$.

4.4 Heuristic Data

We now consider a heuristic approach to LoMax that does not allow you to remove vertices where a large amount of the paint accumulates. This heuristic takes in a graph G and first runs the paint spilling problem with all paint initially distributed at the key vertex. It then creates a set V_{fixed} of size $\lfloor \frac{n}{2} \rfloor$ of the vertices which accumulate the most paint. Finally, it runs LoMax on G with the additional constraint that all vertices in V_{fixed} may not be removed.

By not removing vertices in V_{fixed}, our algorithm only considers removing half of the vertices. Instead of potentially searching 2^n subsets of vertices, this heuristic thus only considers the $2^{\frac{n}{2}} = \sqrt{2^n}$ subsets of vertices of $V(G) \backslash V_{fixed}$. Choosing V_{fixed} to be of size $\frac{n}{2}$ was arbitrary, and the heuristic could be modified to choose V_{fixed} to be of size $f(n)$, for any function f such that $0 \leq f(n) \leq n$. Then at most $2^{f(n)}$ subsets would be considered.

Due to the computational difficulty of finding solutions to LoMax, we cannot run our approach on large graphs. We similarly cannot provide reliable data as to what the ideal function $f(n)$ might be so that the heuristic efficiently finds good solutions to LoMax. We can, however, consider how it works on small graphs using $f(n) = \lfloor \frac{n}{2} \rfloor$ and gather preliminary data to motivate heuristic approaches.

Table 4.1 shows how the solutions found by our heuristic compare to the optimal solutions. We ran both full LoMax and our heuristic on three sets of graphs: graphs of size 15, 18 and 21, randomly created using the $G(n, \frac{5}{n})$ model. (Here we are testing smaller graphs and have set $p = \frac{5}{n}$ rather than $\frac{7}{n}$ so that, especially in the 15 vertex graphs, the probabilities of each edge appearing are more reasonable.) The table records, on average, how quickly and effectively the heuristic was on each set. For each data set, the table shows the average value of the heuristic solution divided by the optimal solution using full LoMax (where by "value" of the solution we mean the value of the load on the key vertex when the subset found by the heuristic or LoMax was removed). The table also records the time it took to find a solution using the heuristic solver as a percentage of the time it took to find the solution to LoMax.

In general, we see that the heuristic tends to find good solutions. It runs much more quickly as only half of the vertices are considered for removal. While the heuristic ran in less than a minute on each 21 vertex graph, the full LoMax solution took more than three and a half hours on each 21 vertex graph.

n	Number of Graphs Tested	Percent of Optimal Solution	Percent of Time
15	50	89.0%	3.05%
18	25	82.1%	0.813%
21	5	76.0%	0.407%

Table 4.1 Performance of the paint spilling heuristic by graph size. The maximum value of the solution found by the heuristic is given as a percentage of the value of the optimal solution. The running time of the heuristic is given as a percentage of the running time for solving LoMax.

This heuristic may or may not be reliable as n grows larger, but we have included data showing that it works reasonably well on small graphs as a "proof of concept." Moreover, it indicates that relationships between different types of network flow exist. We might be able to exploit these relationships to develop efficient heuristics. Connections to other flow problems might also reveal more insight into the types of vertices whose removal most increases the communication forced through a highly central key vertex.

Chapter 5

Future Work

In this chapter we describe, in detail, two open areas directly related to the work in this thesis: characterizing load and applying spectral graph theory. In our discussions of these areas we will explicitly list and motivate several open questions and we will describe questions that can be answered both theoretically and empirically.

The references provided throughout this thesis provide a natural starting point for further investigations into load. A thorough introduction to network theory can be found in Newman (2010), and a similar introduction to graph theory can be found in West (2001). For readers interested in the more specific topics discussed in this thesis, Spielman (2012) and Spielman (2010) provide a phenomenal introduction to spectral graph theory and its applications. A more detailed discussion of the computational implementation of LoMax, and the optimization techniques that went into the solver, can be found in Paul (2012).

5.1 Characterizing Load

The majority of our work in Chapter Three of this thesis was spent characterizing load as a structural property of graphs. There are several additional questions that can be asked. One particularly relevant question relates to Conjecture 3.1: is the maximum possible load, as a function of the number of vertices n, actually $\frac{1}{8}n^3 + O(n^2)$? Is it achieved in any graphs that are fundamentally different from the one presented in Figure 3.2 (i.e., graphs that are not obtained by trivially modifying Figure 3.2)?

Beyond the maximum load problem, there are several other questions that can be asked. We saw, for example, that the load on central vertices in paths as well as on any vertex in a cycle or in a complete graph was $\Theta(n^2)$. Is the load on a highly central vertex in a graph almost always $\Theta(n^2)$? Are there certain features of graphs that distinguish between load being $\Theta(n^2)$ and $\Theta(n^3)$? If such features existed, could they be used to develop heuristics for the LoMax problem? That is, one approximate approach to LoMax might be to take a graph with load $\Theta(n^2)$ and try to remove some subset of vertices to create features associated with load of $\Theta(n^3)$.

We saw, for example, that graphs where $G \backslash k$ consisted of two complete components of equal size had a high load on k. One heuristic approach could then be to remove vertices to, as much as possible, make $G \backslash k$ consist of two components of roughly similar size.

5.1.1 Random Graph Theory and Empirical Data

Answers to questions about random graphs could also reveal information about many of the above questions. In particular, knowledge of the expectation and variance of the load on both general and highly central vertices in an entirely random network would help us to understand typical loads in networks. Moreover, statements about typical random graphs can provide insight into how properties relating to load scale with the size of a network. Does, for example, the probability of there being a subset of vertices whose removal increases the load on a central vertex go to 1 as the size of the network goes to infinity? Can we say something something about the size of the subsets we need to look for? These questions can be asked in the context of generic random graph models (like including each edge with a fixed probability) but they can also be asked about models that describe the types of networks studied in specific applications.

Beyond characterizing load as a property of graphs, some of the deepest questions can provide information that we might be able to incorporate into heuristics. For example, how is the size of solutions to LoMax distributed? That is, if S is the subset whose removal most increases the load on k, how big do we expect S to be as a function of n? How much of a load effect can we expect S to have? Answering the former question might allow us to bound the size of a subset that approximate LoMax solvers search for. Answering the second question would provide an idea of when a good (or close to optimal) solution might have been found and an approximate LoMax solver could stop searching subsets once it finds a good solution.

While the above questions can be answered most accurately by a theoretical study of random graphs, one could also try to address them empirically.

5.2 Spectral Graph Theory

In this thesis we have also started to study network disruption through spectral graph theory. The spectrum of a graph has tended to reveal incredible amounts of information about the structure of a graph and, to the best of our knowledge, has not been applied to the concept of load. In Chapter Four, we thus briefly noted that one might hope to find connections between eigenvalues and eigenvectors of matrices associated with graphs, like the Laplacian (again, defined as $D - A$, where D is the degree matrix and A is the adjacency matrix. See Chapter Four).

5.2.1 The Spectrum of the Laplacian

While considering spectral graph theory, we began to study connections between eigenvalues, eigenvectors and load. We started by examining whether or not the changes in load (on highly central vertices) correlated with changes in specific eigenvalues of a graph. It is worth noting that a graph on n vertices has n eigenvalues, and so $G \backslash S$ will have $n - |S|$ eigenvalues. Hence, G and $G \backslash S$ will have a different number of eigenvalues, and we did not look for correlations between changes in load and changes in the i-th eigenvalue. Instead, we considered changes in the i-th largest and i-th smallest eigenvalues, where i was relatively small. Because G and $G \backslash S$ have different sized vertex sets, we also looked at normalizing the changes in load and the changes in eigenvalues by the size of these graphs.

We also considered connections between eigenvectors and load. An eigenvector can be viewed as a function on the vertex set (if the vertices are ordered $v_1, v_2, \dots, v_n$, then an eigenvector $\mathbf{v}$ assigns the number $\mathbf{v_i}$ to v_i). When viewed as such a function, an eigenvector could provide insight into which vertices to consider (or not consider) removing during LoMax. For example, if one of the eigenvectors assigned the same number to a vertex v and the key vertex k, than removing v might be correlated with a particularly large or a particularly small load effect on k.

While our results were inconclusive in both of the studies described above, these questions are very broad. Also, towards the end of our research we started noticing some connections between the largest eigenvalue and load (normalized by n^2) There are many other ways the eigenvalues and eigenvectors could be connected to load and the LoMax problem. There are also many other ways the eigenvectors, viewed as functions on vertices, could be studied.

Even if no proofs can be established relating the spectrum to the load in a graph, correlations between them could be used in developing heuristics akin to the heuristic presented in Chapter 4. Moreover, hinting at such connections could lead to rich connections between network disruption and spectral graph theory which, in turn, could lead to future work in both fields.

Lastly, in this thesis we have highlighted graphs where the load is abnormally large or small. One way to start addressing spectral questions, suggested at the 2014 Pacific Coast Undergraduate Mathematics Conference, would be to examine the spectra of the graphs we have highlighted. For example, consider the graph from Proposition 3.13. If we label the vertices with 1 through $m-1$ in the first complete graph, then m as the shared vertex, and $m-1$ through $2m-1$ in the last component, the Laplacian is:

$$\left(\begin{array}{ccccc|c|ccccc}
m-1 & -1 & -1 & \cdots & -1 & -1 & 0 & 0 & 0 & \cdots & 0 \\
-1 & m-1 & -1 & \cdots & -1 & -1 & 0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
-1 & -1 & -1 & \cdots & m-1 & -1 & 0 & 0 & 0 & \cdots & 0 \\
\hline
-1 & -1 & -1 & \cdots & -1 & 2(m-1) & -1 & -1 & -1 & \cdots & -1 \\
\hline
0 & 0 & 0 & \cdots & 0 & -1 & m-1 & -1 & -1 & \cdots & -1 \\
0 & 0 & 0 & \cdots & 0 & -1 & -1 & m-1 & -1 & \cdots & -1 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & \cdots & 0 & -1 & -1 & -1 & -1 & \cdots & m-1
\end{array}\right).$$

The vertical and horizontal lines inside the matrix highlight the middle shared vertex. The matrix has the following eigenvalues: $2m-1$, with multiplicity 1, m the multiplicity $2m-4$, 1 with multiplicity 1 and 0 with multiplicity 1. To see this, we first construct $2m-4$ linearly independent eigenvectors with eigenvalue m. There are $m-2$ of the form:

$$\left(\begin{array}{c} 1 \\ -1 \\ 0 \\ \vdots \\ 0 \\ \hline 0 \\ \hline 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{array}\right), \quad \left(\begin{array}{c} 1 \\ 0 \\ -1 \\ \vdots \\ 0 \\ \hline 0 \\ \hline 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{array}\right), \quad \left(\begin{array}{c} 1 \\ 0 \\ \\ \vdots \\ -1 \\ \hline 0 \\ \hline 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{array}\right).$$

and $m-2$ of the form:

$$\left(\begin{array}{c} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \hline 0 \\ \hline 1 \\ -1 \\ 0 \\ \vdots \\ 0 \end{array}\right), \quad \left(\begin{array}{c} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ \hline 0 \\ \hline 1 \\ 0 \\ -1 \\ \vdots \\ 0 \end{array}\right), \quad \left(\begin{array}{c} 0 \\ 0 \\ \\ \vdots \\ 0 \\ \hline 0 \\ \hline 1 \\ 0 \\ 0 \\ \vdots \\ -1 \end{array}\right).$$

where the horizontal bars follow the same structure as above and box off the m-th element.

We also have the eigenvalue $2m-1$, 1 and 0, respectively with eigenvectors:

$$\left(\begin{array}{c} 1 \\ 1 \\ \vdots \\ 1 \\ \hline -2(m-1) \\ \hline 1 \\ 1 \\ \vdots \\ 1 \end{array}\right), \quad \left(\begin{array}{c} -1 \\ -1 \\ \vdots \\ -1 \\ \hline 0 \\ \hline 1 \\ 1 \\ \vdots \\ 1 \end{array}\right), \quad \text{and} \quad \left(\begin{array}{c} 1 \\ 1 \\ \vdots \\ 1 \\ \hline 1 \\ \hline 1 \\ 1 \\ \vdots \\ 1 \end{array}\right).$$

This graph has a particularly structured spectrum with with many repeated eigenvalues and a substantial gap both between the largest and second largest eigenvalue and between the second smallest and third smallest eigenvalue.

5.2.2 Other Flows

In Chapter Four, we showed one way of studying connections between load and the paint spilling problem. We considered correlations between the way wet paint and communication might flow through the same network. There are, however, many other types of flow that have been studied using spectral graph theory. For example, spectral graph theory can be used to study electrical flow in networks (see Spielman (2010)).

Spielman (2014) has suggested studying electrical flow in networks in light of the LoMax problem. The mathematics used to analyze these flow problems could potentially be modified to study LoMax. Moreover, electrical flows might relate to communication flow and networks. That is, one could study electrical flow, see how it correlates with communication flow, and then attempt to develop approximate LoMax solvers like the one we developed in Chapter Four.

Chapter 6

Conclusions

In this thesis we have motivated and defined load and the load maximization, or LoMax, problem which was first introduced in Martonosi et al. (2011). The optimal solution to LoMax forces as much communication as possible through a key vertex. In social or communication networks, the load represents the amount of communication forced through a vertex, and increasing the load might correspond with increasing the visibility of that person. In our motivating problem regarding airport transportation, the optimal solution to LoMax would force as many of CUT's flights through the Menger airport as possible, increasing the profits for FloW.

We briefly described approaches to solving LoMax computationally. However, such approaches tend to be infeasible to implement on large networks. Thus, we began to consider the LoMax problem more theoretically. Continuing the work in Martonosi et al. (2011) and Paul (2012), our goal was to provide insight into load that might help characterize the subsets of vertices whose removal most increases the load on the key vertex.

Our primary contributions fall into two main categories: combinatorial and spectral. In Chapter Three, we worked towards characterizing load as a metric assigned to vertices in graphs. We mathematically described what load looks like in several broad classes of graphs, and just as importantly, mathematically described how the load changes in those graphs when vertices are removed. Those results provide insight into what the load looks like in graphs with a significant amount of structure.

Having computed the load on specific vertices in several types of graphs, our next step was to characterize the range of values load can take on. We were able to show that the maximum load is at most of order n^3, and that it is between $\frac{1}{8}n^3 - O(n^2)$ and $\frac{1}{4}n^3$. We conjectured that the maximum load is in fact only $\frac{1}{8}n^3 + O(n^2)$ and provided several tools that might help prove this statement. We were also able to prove that the maximum load is $\frac{1}{8}n^3 + O(n^2)$ in specific family of graphs.

The final contributions of this chapter were several structural theorems that described how the removal of vertices of low degree can change the load and showed that it need not always be possible to increase the load (even on highly central vertices in graphs with an arbitrarily large proportion of edges present). These combinatorial results provide intuition about what load looks like in large graphs, and they more thoroughly characterize load as a property of vertices in graphs.

In Chapter Four, we specifically looked at spectral graph theory. We started highlighting connections between spectral graph theory and load using empirical data. We studied the paint spilling problem and used it to create a heuristic solvers for LoMax. While we were not able to compare the solutions from our heuristic solver to the optimal LoMax solution on large graphs, it performed efficiently and tended to find good solutions. Moreover, it suggests ways that other heuristic solvers can be developed using problems from spectral graph theory.

Perhaps the most exciting parts of research into load are the vast number of questions that can be asked and the equally vast number of perspectives from which the problem can be studied. We specifically emphasized combinatorics and spectral graph theory in this thesis, and highlighted several open questions in Chapter Five. Our hope is that these questions provide motivation for future study.

Bibliography

Bruce Hoffman. *Inside Terrorism*. Columbia University Press, 2006.

Susan E. Martonosi, Doug Altner, Michael Ernst, Elizabeth Ferme, Kira Langsjoen, Danika Lindsay, Sean Plott, and Andrew S. Ronan. A new framework for network disruption. *CoRR*, abs/1109.2954, 2011.

M.E.J. Newman. *Networks: An Introduction*. Oxford University Press, 2010.

Alice Paul. Detecting covert members of terrorist networks. *Harvey Mudd College Senior Thesis*, 2012.

Marc Sageman. *Understanding Terror Networks*. University of Pennsylvania Press, 2004.

Daniel A Spielman. Spectral Graph Theory fall 2010 lecture notes. http://www.cs.yale.edu/homes/spielman/462/2010/index.html, 2010. Accessed: 2014 March 14.

Daniel A Spielman. Spectral Graph Theory fall 2012 lecture notes. http://www.cs.yale.edu/homes/spielman/561/, 2012. Accessed: 2014 March 14.

Daniel A Spielman. Personal Communication, 2014. 2014 January 31.

Douglas West. *Introduction to Graph Theory*. Prentice Hall, second edition, 2001.