Structure and Dynamics of Information in Networks

David Kempe
Department of Computer Science
University of Southern California

March 20, 2018

Preface

The present notes are derived from a course taught at the University of Southern California. The focus of the course is on the mathematical and algorithmic theory underpinning the connections between networks and information. These connections take two predominant forms:

- Network structure itself encodes a lot of information. For instance, friendships between individuals let us draw inferences about shared interests or other attributes (location, gender, etc.). Similarly, hyperlinks between documents indicate similar topics, but can also be interested as endorsements, and thus hint at quality. The list of scenarios in which network structure helps us interpret information about individual nodes continues beyond this list, and will be explored in more detail throughout these notes.

- Networks also play a crucial role in disseminating or gathering information. This applies both to social networks, in which communication between individuals happens naturally, and computer networks, which are designed explicitly to facilitate the exchange of information and distributed computations. We will draw analogies between the two types of networks, and investigate the mathematical underpinnings of the diffusion of information over networks in the later chapters of these notes.

These notes are designed to accompany a one-semester graduate-level course in computer science. We assume a solid mathematical background, and familiarity with basic algorithmic techniques, including flows and cuts; the necessary topics are covered in textbooks by Kleinberg and Tardos [246] and Cormen et al. [110]. In particular, we will assume familiarity with Linear Programming (see, e.g., [96, 225]) and the concept of approximation algorithms [201, 372], as well as with basic notions of randomization in algorithms [288, 300]. A basic understanding of graph theory is also required (see, e.g., [53, 126]).

0.1 Further Reading

Several recent books cover topics overlapping with the content of these notes. "The Structure and Dynamics of Networks" [313] by Newman, Barabási and Watts collects a number of recent papers on network analysis and epidemic processes on networks, predominantly with a physics bent in the type of analysis. The collection "Network Analysis" edited by Brandes and Erlebach [67] covers many of the algorithmic and mathematical foundations of static network analysis.

The recent book "Social and Economic Networks" [213] by Matthew Jackson covers several of the same problems studied here, in particular the diffusion of innovations and modeling and properties of networks. Its focus tends to be on essentially uniformly random models of networks, and mean-field approximations in the style also performed in the physics literature. The book "Social Network Analysis" by Wasserman and Faust [378] covers the tools and concepts employed by sociologists in the analysis of social networks in great detail; the book "Social Network Analysis: A Handbook" by Scott [350] gives a more concise and basic introduction to these topics.

The recent book "Community Detection and Mining in Social Media" [363] covers many of the same topics as the present notes. The focus there is more on the data aspect (whereas the present notes focus more on proofs), so the two sets of notes are complementary.

0.2 Acknowledgments

The course I taught at the University of Southern California was strongly influenced by a similar course taught by Jon Kleinberg at Cornell University (CS685 at the time; now known as CS6850). The selection of topics, style of presentation, and organization owe a great deal to Jon's insights and teaching styles. Frankly, Jon should be listed as a co-author; unfortunately, his well-known modesty is preventing him from accepting co-authorship.

The foundation of these notes were scribe notes produced by the students of the course CSCI 599 at USC in the Spring of 2005. The students in the class at the time were Farnoush Banaei-Kashani, Shishir Bharathi, Fang Bian, Yuriy Brun, Iftikhar Burhanuddin, Karthik Dantu, Ramki Gummadi, Shyam Kapadia, Salim Khan, Shiva Kintali, Nupur Kothari, Chansook Lim, Fragkiskos Papadopoulos, Animesh Pathak, Ranjit Raveendran, Dustin Reishus, Jyungryun Seo, Viral Shah, Affan Syed, Ashish Vaswani, and Hui Zhang. The notes have also strongly profited from interactions with students who took the class in the Spring of 2008. I would especially like to thank Miquel Camprodon for catching many mistakes in earlier versions of these notes. Nevertheless, there are surely many more mistakes and omissions in them, and I would be most grateful for any feedback.

Contents

Chapter 1

Background

1.1 The early visionary

This course is about the interplay between link network structure and information. There are two natural ways in which these can interact: The information can spread by using the edges of the network, or the network itself can be (part of) the information to be analyzed. Naturally, these two views are not mutually exclusive: often, we will want to analyze networks whose function it is to spread information. Similarly, the mere existence of a network to go with other information will often allow for information to spread as well. Lest this discussion be too abstract, let us begin by considering two of the most prominent types of networks in this course:

1. The World Wide Web (WWW) consists of web pages and the links between them. The links add significantly to the (textual and other) information content of the pages. They allow a focused navigation of the body of text. Furthermore, they indicate which pages may be about related topics, or more relevant, by virtue of heavy linking patterns. Thus, the WWW predominantly falls into the second category above. However, since the owners of web pages will also have access to the pages they link to, or pages further down chains of links, information tends to propagate along links as well, as pages copy important information from others.

2. Social Networks are made up of individuals and their relationships, such as friendships, collaborations, romantic relationships, etc. They serve many different functions, including social support, economic exchanges, and — quite crucially — the spread of information. Thus, they fall more into the first category. However, the mere fact that an individual forms links with others tends to imply that they might care about similar information, or share similar interests. Thus, the network structure also reveals a great deal of information about the individuals, beyond what could be discerned with standard socio-economic questions.

The great proliferation of networked bodies of information, and the much more ready availability of data on all kinds of networks, have recently led to a significant increase in interest in computational analysis of such networks. In many scenarios, such analysis, beyond its scientific interest, also has significant commercial or other benefit. For an example, look no further than the importance of web search, which we will discuss in more detail in Chapter 2.

Before embarking on our exploration of many of the algorithmic questions in this broad domain, we should revisit one of the visionary thinkers, who anticipated many of the recent developments and challenges well before computers were used pervasively, or most of the technologies were available.

In his visionary article "As We May Think" from 1945 [75], V. Bush is articulating the next great research challenge for a generation of scientists. The backdrop for the article was the success of the Manhattan Project, in which the development of the Atomic Bomb during World War II managed to unite and focus a large fraction of the scientists of the US on a common project, with significant scientific success. In his article, Bush articulates his vision that scientists should continue to strive for solving the next big challenge in a similar collaborative effort. The big challenge he wants to see addressed is the vast amount of information available, and the lack of organization thereof. In order to deal with the overwhelming amount of information, Bush suggests having machines which would help with storing, organizing, and physically accessing such information.

Technologically, Bush suggests the use of microfilm for storage, head-mounted cameras for individuals which would let them take pictures easily, and scanner/photography technology. The core of his envisioned technological approach is called "Memex", and could be considered the equivalent of a modern desk and workstation. It consists of a desk with multiple embedded monitors, which can display text, photos, etc. Books and other documents (pictures, data) are stored in the Memex, and can be easily accessed.

The visionary part is his view of creating *associations* between documents. When reading, say, a paper, a reader may need to look up a definition in another paper or book. At that point, having opened both documents to the relevant pages, a button would allow the reader to create an association between the two documents. Thus, Bush really anticipated hyperlinks here. His reasoning is that the rigid structure of a library index does not correspond to the associative type of thinking performed by humans. Bush anticipates not only creating such associations, but also sharing them with others, so that others will not have to perform the same type of work. He calls an annotated document a *trail*. Given the importance of such trails for others, he even anticipates a profession of *trail blazers*, whose job it is to read and process documents, and create associations helpful for others. Notice how virtually all of these visions have come to pass in the WWW: the WWW itself is a giant shared trail, and many sites (Yahoo!, Open Directory, Wikipedia) can be construed as trail blazers.

Bush already anticipates a number of important research challenges in this context. Besides the obvious technological ones, two important directions are privacy and automated processing. If trails, or personal information, are shared, then it becomes important to specify what types of information should not be leaked, and which ones are safe to share. Privacy in data mining has become a huge research area, albeit one which this course will not touch upon. The second question is how to automatically create associations, or mine the associations for useful information that was not obviously contained in the data originally. This question is becoming even more paramount as the amount of information in the WWW keeps increasing dramatically, to the point that the trails themselves are now so numerous that the reader may not be able to identify the most pertinent ones. These types of questions will be among the core topics of this course.

1.2 Graph structure in the web

The most prominent example of a large body of networked information is the World Wide Web. It quintessentially meets the focus of this course: the hyperlinks between pages containing textual information (or other formats) naturally enrich the information content, and allow us to draw additional inferences about the content, its importance, or relationships between multiple pages. Given the central importance of the web, we begin with an exploration of some of its basic graph-theoretic properties.

The presentation here closely follows the paper "Graph Structure in the Web" by Broder et al. [70]. It treats the World Wide Web as a directed graph, whose nodes are the *static pages*, and whose edges are the *hyperlinks* between them. This graph is also called the *Web graph*. The reported data are from a crawl in May 1999, and thus several orders of magnitude smaller than they would be today. Nevertheless, it is interesting to investigate some of the relative sizes of different parts of the Web.

1.2.1 Connectivity Structure

The crawl user by Broder et al. [70] contained ca. 203 million nodes and 1.5 billion edges. Of these, roughly 91% were contained in one large weakly connected component. The fact that there is only one such component is not surprising since there is only *one* World Wide Web. It would be rather surprising to have two completely disconnected large components. The exact percentage of pages in the large component should be taken with a grain of salt. After all, one may argue that small components not connected to or from the "bulk" of the Web may have a much smaller chance of being discovered.

In looking at these numbers, one may suspect that a few highly connected "hubs" are really in charge of connecting all of the other nodes to each other. This suspicion turns out to be incorrect. Even when all nodes of degree more than 5 are removed, there is still a large component with more than 59 million pages. This shows that the connectivity structure is highly "decentralized", and utilizes most of the Web.

Looking at *strongly connected components* (SCCs), the picture is somewhat different. There is still only one strongly connected component (SCC), containing approximately 56 million pages that are *mutually reachable*. Again, the existence of only one large SCC should not be surprising, for the same reason as the weakly connected component. In addition to the giant SCC, the strong connectivity structure contains the parts labeled IN, OUT, tendrils, and tubes in the diagram; they are all discussed below.

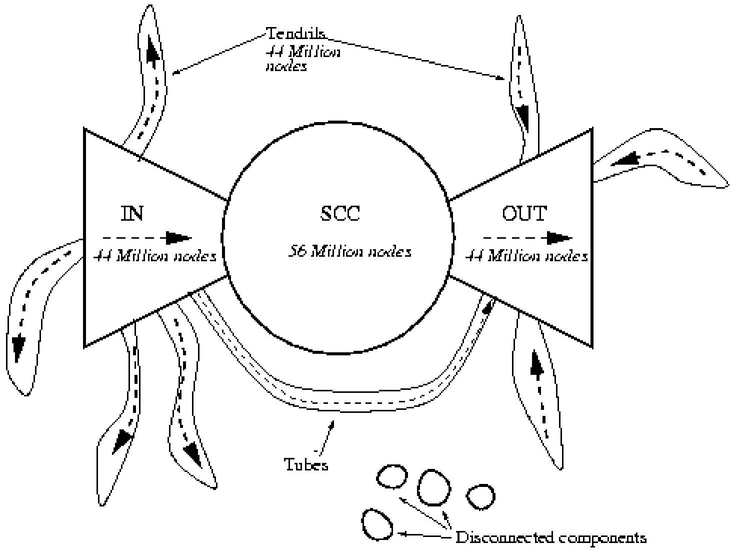


Figure 1.1: Connectivity of the web (diagram taken from [70])

Figure 1.1 gives an overview of the structure. IN denotes the set of all nodes from which the giant SCC is reachable, but which in turn are not reachable from SCC. Similarly, OUT is the set of all nodes reachable from SCC, but not able to reach SCC themselves. Each of them contains about 44 million pages.

Hanging off IN and OUT are *tendrils* containing nodes that are reachable from portions of IN, or that can reach portions of OUT, without passage through SCC. *Tubes* are made up of nodes on paths from IN to OUT, but not passing through the giant SCC. Tubes and tendrils together comprise another 44 million pages. Finally, there are some completely disconnected strongly connected components, whose contribution to the total size of the Web is rather negligible.

The depiction of the strong connectivity structure in Figure 1.1 resembles a bow tie, and the connectivity structure has therefore often been referred to as "bow tie structure" of the web. Notice, however, that *every* directed graph has this structure when drawn with respect to one particular strongly connected component. The bow tie structure really carries no information about the Web specifically; what is more interesting is the relative sizes of the different components.

Notice that the size of some components, most notably IN and the tendrils of OUT, are more likely to be underestimated than the others. The reason is that these components suffer from a "seeding" problem: how to discover a new page if it is not reachable from an already crawled one? Some solutions to this problem are manual submission of web pages and random generation of names, but nevertheless, one would expect that these two components are more undersampled than those which are by definition reachable from known nodes.

Despite these sampling issues, the above structure and relative sizes can be expected to be quite stable, even to missed pages. The reason is that any paths from OUT to IN, from SCC to IN, or from OUT to SCC, would already have been discovered if they existed. So these components will not be contracted.

An interesting consequence of the numbers depicted above is the fraction of node pairs that can reach each other. If we select two nodes at random, here will be a directed path from one node to the other with probability roughly 30%. This can be computed by a simple case distinction over the different components in which the two nodes could land. For such a path exists either when both nodes are in SCC, or one is in IN and the other in SCC or OUT, or the first one in SCC and the second in OUT. (There are some additional, but much less frequent, cases involving tendrils and tubes.) Looking at the sizes of the components then shows that the probability of such selections is about 30%, contrary to the once popular belief that almost all node pairs are connected to each other via directed paths.

1.2.2 Path Lengths

Once we have established the fraction of page pairs which are connected to each other, a second interesting question becomes how many hops separate an average or worst-case pair of pages. These can be computed using invocations of BFS from all start nodes, in time $O(nm)$. Unfortunately, given the orders of magnitude of m and n, this running time is too slow. No faster algorithm for calculating the diameter of a graph is currently known.

Broder et al. [70] instead randomly sample node pairs, and measure their distance. This provides a lower bound on the diameter (since the maximum distance may have eluded the algorithm), and a fairly accurate estimate of the average distance. Using these techniques, Broder et al. gave a lower bound of 28 on the diameter of SCC, and a lower bound of 503 on the diameter of the WWW. The much larger bound on the entire WWW is likely due to long chains of pages (chapters of a book, for instance), which would likely be either in the OUT component, or inside tendrils. The *average* distance between sampled pairs of nodes in the SCC was found to be roughly 16 if the edge direction was considered and roughly 7 if the edge direction was disregarded.

Given that time $O(mn)$ is too slow for extremely large graphs for computing the diameter, an interesting approximation algorithms question would be for which types of graphs random sampling gives a *good* lower bound on the diameter of the graph, i.e., trying to find a structural characterization of graphs for which many node pairs have nearly the maximum distance.

1.2.3 Degree Distributions

Another interesting statistic is the degree distribution of web pages. How many links go into or out of the "typical" page? Broder et al. [70] denote by p_i the number of pages with i incoming links, and by q_i the number of pages with i outgoing links. Their analysis finds that $p_i \sim i^{-2.1}$, and $q_i \sim i^{-2.72}$, for several orders of magnitude of i. Thus, both the indegree and outdegree seem to follow a power law.

Remark 1.1 A power law is a degree distribution of the form $p_i = C \cdot i^{-\alpha}$, for some constants $C, \alpha > 0$. It can be recognized "heuristically" by drawing the distribution in a log-log plot. Then, $\log p_i = \log C - \alpha \log i$, so the power law appears as a straight line with slope $-\alpha$ and intercept $\log C$. In order to be meaningful, a power law should normally manifest itself over at least three orders of magnitude. We will discuss power laws in much more detail in Chapter 6.

How interesting is it to observe a power law degree distribution? By itself, a power law does not tell us much about the structure of the graph: there are still many completely different graph structures which would all have the same degree distribution. In particular, the term "power law graph" does not really carry much meaning.

However, one sense in which the power law observation is somewhat interesting is that it clearly rules out the simplest graph model — namely the Erdős-Rényi $G(n, p)$ model [141] as an accurate description of the web graph. In an Erdős-Rényi graph of n nodes, each edge is present *independently* with probability p. The result is that the degree distributions are Binomial. In particular, the degrees will be sharply concentrated around their expectation pn. Erdős-Rényi graphs are often used as default models, absent a better understanding of the graph structure, and the degree distribution observed by Broder et al. rule out that $G(n, p)$ could be an accurate model.

We should also stress here that the interesting part of a power law is that it captures a fairly large number (only polynomial decay) of nodes with high values of i, i.e., large in-degree or out-degree. Not too surprisingly, the indegrees have a heavier tail, since it requires less effort to *receive* a significant number of incoming links than to *create* a significant number of outgoing ones.

1.3 Further Reading

The early history of the WWW, including Bush's paper and many other position documents and milestones, are described in a short history of the WWW from 1945–1995, available online [108]. For a personal account of the early days and invention of the Web from one of its inventors, Tim Berners-Lee, see his book "Weaving the Web" [40].

Another early paper analyzing graph properties of the Web is [244]. In fact, it also presents some graph models discussed later, web search algorithms, and algorithms for community detection. Beyond degree distributions and path lengths, it analyzes the frequencies of certain dense subgraphs, and connectivity properties of local subgraphs.

Some of the mathematical issues that will be explored in this course are described nicely in a survey of research directions accompanying a talk given by David Donoho [131].

Chapter 2

Web Search using Links, and Spectral Analysis

In this chapter, we investigate the use of web graph structure to improve web search. This application cuts straight to the heart of this course, in that the network structure augmenting the textual information contained in web pages can be used to significantly improve the performance of information extraction algorithms. Our focus, after a brief discussion of the issues encountered in web search in general, will be on two algorithms: Hits and PageRank. The latter underlies the successful search engine Google.

Both of the algorithms are based on computing eigenvectors of certain matrices representing the Web graph. The reader unfamiliar with (or rusty on) basic linear algebra might find this a good time to consult standard linear algebra texts such as Strang's [360] or Horn and Johnson's [208]. The eigenvector based algorithms in PageRank and Hits also bear some similarity with techniques used in analyzing textual data, specifically Latent Semantic Analysis. We therefore turn our attention to that topic in Section 2.5.

2.1 Challenges In Searching

Writing a search engine involves many challenges. The first type are technical: writing a crawler that deals with the many types of formats, and — more challenging — all the types of errors human authors of web pages introduce, is itself a complicated task. In addition, crawlers also have to avoid overloading web servers and deal with connection timeouts. A good crawler needs to repeat crawls frequently, to avoid the problem of having stale or outdated information available in searches. This is particularly important for searches about recent or upcoming events, to which the correct answers changes frequently.

A more recent important trend is the development of the "deep web": information which is not readily available in web pages, but rather stored in data bases, and accessible only through query forms. It is not clear how such information could (or should) be accessible to search engines, and how it can be returned in response to queries.

Designing the actual search engine to deal with the large numbers of queries encountered in practice is also a significant engineering challenge, and solving it successfully has played as large a part as algorithmic insights in the success of current search engines. However, in this course, we will focus more on the algorithmic challenges inherent in answering different types of queries. For this purpose, it is useful to have a brief look at the different types of queries commonly posed to search engines. This list is neither exclusive nor exhaustive, but highlights some of the issues a web search engine designer will have to take into account.

1. *Specific queries*, such as "Does Netscape support the JDK 1.1 code-signing API?" As we will discuss in detail below, the difficulty in handling specific queries is *scarcity*: there are very few or no pages which contain the required information or keywords, and it is often difficult to determine the identity of these pages.

2. *Broad-topic queries*, such as “Find information about the Java programming language.” Here, we have the opposite problem: the number of pages that could potentially be relevant is far too large for a human user to digest. We will need to devise techniques to narrow down or rank the set of pages returned.

3. *Similar-page queries*, such as “Find pages ‘similar’ to java.sun.com.” The problem here is to determine in what respects the pages are supposed to be similar to the present one. This requires, at least implicitly, forming an estimate of the topic or content of a page.

4. *Current events*. Here, the challenge is that the data base needs to be very up to date, and hence the crawling needs to be regular. For such dynamic changes, one might fear that link structure will not change fast enough to help improve accuracy.

5. *Navigational queries*, such as “Home page of the Sun corporation.” Such a query essentially replaces the formation of a bookmark as navigational help. Therefore, it is important that the best fit be clearly identified and listed first.

6. *(Personalized) Opinion*, such as “The best hotel under $100 in Los Angeles.” Here in particular, the search engine might even need to know more about the opinions of the searcher, but also needs to aggregate opinions expressed in pages to form some kind of consensus.

In dealing with these different types of queries, the first broad class of algorithmic challenges is the “classic” *information retrieval (IR)* problem of extracting useful information from a large body of text or other sources. For instance, different authors use different formats (html, pdf, xml, jpg, etc.), different authoring styles, or different terminology to refer to the same idea or object. Extracting the meaning or intent of text has been a traditionally hard problem, and is a research area of its own. We will discuss a very basic technique in Section 2.5, but will otherwise not make any attempts to cover this area in this class.

In the context of web search, the traditional IR problem is further complicated by the fact that the authors of the individual pages may have goals vastly differing from those of the search engine or the searcher. Many web sites are actively misrepresenting their content or importance in order to attract more visitors. Common techniques include adding interesting words even when they do not relate to the page’s content (called *content spamming*), or linking to a page from other pages such as wikis or blogs (called *link spamming*), in order to make the page appear more relevant. As a result, there has been a lot of recent research work on dealing with such web spam, and more generally, the area of *adversarial information retrieval* has been very active.

Here, we will spend more time exploring solutions to the problems caused by *specific* and *broad* queries. A sparse query may have only few (or no) results, even though the searcher’s *intent* would result in relevant answers. A broad query might have thousands or millions of results, and it becomes imperative to extract the ones most likely to be useful. Both commonly occur because the user — not knowing which pages will answer his query — does not know which particular keywords to enter to have the engine return these queries.

The problem of sparse queries can be at least partially solved using traditional IR techniques, such as augmenting a search by known synonyms, or other words frequently used by other searchers together with the search terms. However, the link structure provides perhaps much more useful additional information. For instance, imagine that the search is for “car manufacturers”. The searcher’s intent is probably to find the websites of Toyota, Honda, etc. However, some or many of these sites may not contain the exact search terms. To deal with this issue, we can notice that there will likely be many websites containing the phrase “car manufacturers”, and *linking to* the sites of actual car manufacturers. Thus, using information contained in linking (or linked-to) pages, we can infer additional information about a web site.

Link structure may also help us deal with the *abundance* problem: the number of pages containing a query can be extremely large, and must be ordered by *relevance*. To estimate a page’s relevance, we can consider the number and nature of links it receives, while positing that incoming links implicitly express some notion of endorsement. Thus, other things being equal, pages with more links could appear more relevant. These ideas essentially underlie most of the successful search engines used today, and we will elaborate on the underlying mathematics in more detail in the next sections.

2.2 The HITS Algorithm

The HITS algorithm was suggested by Kleinberg [240], and is one of the first algorithms to use link structure in order to deal with the abundance problem. It has two components: first, it constructs a focused subgraph of the web; second, it uses "hub" and "authority" weights to identify particularly relevant pages.

The first component of HITS deals with the problem mentioned above in the context of sparse queries: that some of the most relevant pages might not contain the query terms. To deal with this issue, the algorithm first identifies a core set of approximately 200 pages using standard text-based search. This set is expanded by adding all the pages linked to from the core set, and some pages linking to the core. Since many pages could conceivably link to one page in the core set, the number of in-links is restricted to about 50, which are chosen heuristically. In total, this generates a focused induced subgraph G of $n \approx 3000$ nodes.

The second component of HITS aims to deal with the issue of broad searches, and attempts to find the most "relevant" pages in G. It is based on the observation that a page could be relevant for two reasons: (1) it could be a good *authority*, containing genuine information about the topic, or (2) it could be a good *hub*, collecting links to many good authorities.

The distinction, and the introduction of hubs, requires a bit more justification. Why are we not simply considering authorities and their linking behavior to each other? In fact, we will consider this approach in the context of the PageRank algorithm (which forms the core of Google's ranking system) in Section 2.3. One justification for considering hubs in addition to authorities is that much of the World Wide Web forms a *competitive* environment, in which multiple authoritative pages on the same topic may not link to each other. For example, in the context of "car manufacturer" web pages discussed previously, we would not expect the most authoritative pages — such as Honda, Toyota, or Ford — to recommend each other to a customer. Any connection between these pages will have to be *indirect*, by virtue of other pages considering all of them authoritative.

To identify which authorities are good, we use the observation that good authorities should be pointed to by (many) good hubs. In turn, a hub is good if it points to (many) good authorities. The tacit assumption here is that links constitute an endorsement. The definition of what constitutes a good hub or authority is circular. However, we can translate it into an actual algorithm as follows.

Each page v has authority weight a_v and hub weight h_v, initialized to 1. In any one iteration, these are updated, by having authorities inherit authority weights from the hubs that point to them, and hubs inherit hub weight from the authorities they point to. Thus, one iteration updates

$$\begin{aligned} a'_v &= \sum_{u \to v} h_u \qquad \text{for all } v \\ h'_v &= \sum_{v \to u} a'_u \qquad \text{for all } v. \end{aligned}$$

Notice that after one iteration, the authority weight of v is the number of pages pointing to v, which is intuitively related to the authority of a page. Further iterations refine this notion by giving more weight to hubs that are deemed more relevant.

If we let $\mathbf{a}, \mathbf{h} \in \mathbb{R}^n$ denote the vectors of all authority and hub weights, we can express the same operations in terms of matrices as follows. Let B be the adjacency matrix of G, i.e., $B_{ij} = 1$ iff there is an edge $i \to j$. Then, we can rewrite the update rules as

$$\begin{aligned} \mathbf{a}' &= B^\intercal \cdot \mathbf{h} \\ \mathbf{h}' &= B \cdot \mathbf{a}'. \end{aligned}$$

Hence, the update rule for authority weights can be written as $\mathbf{a}' = (B^\intercal B) \cdot \mathbf{a}$, and for hub weights as $\mathbf{h}' = (BB^\intercal) \cdot \mathbf{h}$. The matrix $B^\intercal B$ is called the *co-citation matrix*. We can understand it intuitively as counting the number of pages pointing to both i and j.

To ensure that this update rule converges to a solution, we need to normalize the weights after each update, for instance such that $\sum a_v^2 = 1$ and $\sum h_v^2 = 1$ (otherwise, the values would keep growing).

We will revisit the issue of whether the normalized rule converges in a moment. However, we first want to explore what it would converge to, if at all. Convergence means that subsequent iterations do not cause any changes, so $\mathbf{a}, \mathbf{h}$ would have to be *fixed points*, i.e., $\mathbf{a}' = \beta\mathbf{a}$ and $\mathbf{h}' = \gamma\mathbf{h}$, where β, γ are the respective normalization constants ensuring that $\sum(a'_v)^2 = 1$ and $\sum(h'_v)^2 = 1$.

Substituting the expressions we derived above gives us that $\beta \cdot \mathbf{a} = (B^\intercal B) \cdot \mathbf{a}$ and $\gamma \cdot \mathbf{h} = (BB^\intercal) \cdot \mathbf{h}$. Thus, the authority weights are exactly an eigenvector of $B^\intercal B$ with eigenvalue β, and the hub weights are exactly an eigenvector of $BB^\intercal$ with eigenvalue γ.

Now, we can return to the issue of convergence of the iterative process (along with the question *which* of the eigenvectors are returned). Since the matrix $B^\intercal B$ is symmetric, $\mathbb{R}^n$ has a basis of orthonormal eigenvectors $\omega_1, \omega_2, \ldots, \omega_n$ of $B^\intercal B$. Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the corresponding eigenvalues with $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$. If the graph corresponding to $B^\intercal B$ is connected, then by the Perron-Frobenius Theorem for non-negative matrices, $|\lambda_1| > |\lambda_2|$, ω_1 has all non-negative entries, and each ω_i for $i > 1$ has at least one negative entry. Because the eigenvectors form a basis, we can write the starting vector of authority weights as $\mathbf{a}_0 = \sum_{i=1}^n \alpha_i \omega_i$, and the previous properties imply that $\alpha_1 \neq 0$. The next iteration of authority weights then gives

$$\mathbf{a}_1 \quad = \quad (B^\intercal B) \cdot \mathbf{a}_0 \quad = \quad \textstyle\sum_{i=1}^n \alpha_i (B^\intercal B) \omega_i \quad = \quad \sum_{i=1}^n \alpha_i \lambda_i \omega_i.$$

By induction, we can now show that for any k,

$$\mathbf{a}_k \quad = \quad \sum_{i=1}^n \alpha_i \lambda_i^k \omega_i. \tag{2.1}$$

Hence, as $k \to \infty$, the normalized authority weights converge to $\mathbf{a}_\infty = \omega_1$. Therefore, authority weights are just the first eigenvector of $B^\intercal B$. By an identical argument, the hub weights converge to the first eigenvector of $BB^\intercal$.

In retrospect, we can thus think of the authority weight computation as using the *power iteration* method (see, e.g., [180]) to compute the first eigenvector of $B^\intercal B$. Of course, one could instead use different techniques to compute the eigenvector. Usually, power iteration is quite fast. However, as we can see from Equation (2.1), its convergence rate depends on the *spectral gap* $|\lambda_1| - |\lambda_2|$. In fact, if $|\lambda_1| = |\lambda_2|$, the method may not converge at all. On the other extreme, if the spectral gap is bounded by a constant, independent of the number of nodes n, then the power iteration will converge exponentially fast. (To see this, simply use triangle inequality for the difference between $\mathbf{a}_k$ and ω_1.)

2.2.1 Extensions

Having been so successful with interpreting the top eigenvector of the co-citation matrix and its transpose, we may be interested in interpreting its other eigenvectors as well. It turns out that they can be considered as identifying interesting hub-authority structure in subgraphs of G, and will prove useful in identifying more specific topic communities within G. We will return to this issue in Section 3.3.1.

The approach of HITS can be easily extended to help with finding pages related to a given page. Dean and Henzinger [118] suggest simply building the focused subgraph starting at the given page u. All pages pointing to u, and pointed to by u, are included (up to a given size limit). In addition, all pages pointed to by parents of u, and pointing to children of u, are also included, up to a degree constraint per page. ([118] suggest heuristics for deciding *which* links to retain if the degree of a page is too large.) Once the focused subgraph has been determined in this way, one simply computes the authority scores and outputs the top authorities. Naturally, looking at web graph communities containing the page u is another way to uncover related pages (see, e.g., Section 3.3.1). We will investigate communities in more detail in Chapter 3.

2.3 PageRank

In the previous section, we looked at and analyzed the HITS algorithm, based on Hubs and Authorities. HITS was designed with a scenario in mind where authorities might not cite each other. Hence, HITS uses the idea of conferring authority indirectly through hubs. In scenarios where authorities do cite each other, such as academic research work, it may be more appropriate to have direct conferral of authority between nodes. This is the idea behind the PageRank [68] algorithm.

Which algorithm is "better" may be hard to evaluate, and will depend on the query, among others. It raises the more general question of how one can even evaluate the performance of an algorithm when its objective function is not clearly defined.

2.3.1 A first approach

Based on our intuition above, we would like to say that a page has high PageRank if it is pointed to by many pages of high PageRank. On the other hand, if a page points to many other pages, it presumably will not confer a lot of authority to all of them, but rather divide its authority evenly. This suggests the following update rule, starting from an arbitrary vector, such as $p_i = 1/n$ for all i: the new PageRank is $p'_i = \sum_{j \to i} \frac{1}{d^+(j)} \cdot p_j$, where $d^+(j)$ is the out-degree of j.

We can immediately see that at any fixpoint of this update rule, the solution has to be the solution to a linear system in n variables. We could also converge to the fixpoint by iteratively applying the update rule. A more concise way of writing the update rule can be obtained as follows. Let M be the square matrix with rows and columns corresponding to web pages, and $M_{ij} = \frac{1}{d^+(i)}$ if $i \to j$ and 0 otherwise. With $\mathbf{p}_k$ denoting the vector of PageRank weights after k iterations, the update rule can be written as $\mathbf{p}_{k+1} = M^\intercal \cdot \mathbf{p}_k$. Notice that the operation has the useful property of "Mass Conservation": the sum of the PageRank weights of all pages is always exactly 1, as it is only redivided along outlinks.

2.3.2 A Problem and Solution

The above solution has a problem: it may not converge, and when it does, the result may not be what we are looking for. Specifically, if a web page has no outlinks (except, say, one to itself), then it will never pass on PageRank weight to any other nodes, but it will receive PageRank weight. Hence, all of the weight will collect at sinks, or, more generally, in the sink strongly connected components. All other nodes will have PageRank 0, which is certainly not corresponding to our intuition. In addition, which sinks will end up with all the weight will depend on the starting vector, so the PageRanks are not independent of the starting assignment, which is another undesirable property. Finally, in the case of a cycle of two nodes, the weight will oscillate between those nodes, but not converge.

There is a simple way to fix all of those problems at once. To motivate the approach, we notice that we can view M as the matrix of transition probabilities of a random walk on the web graph. When the random walk is at a node i, it chooses uniformly among all outgoing links, and follows that link. Hence, the first attempt above corresponds to computing the probabilities of being in a given state i in the limit of infinitely many steps. We can now modify the random walk as follows: for some small $\epsilon \approx 1/7$, with probability $(1-\epsilon)$, the new Markov Chain does exactly the same as the old random walk. With the remaining probability ϵ, the new Markov Chain chooses a uniformly random vertex and jumps to it. This random process can intuitively be motivated by the model of a random surfer who gets bored and jumps to a uniformly random page with probability ϵ in each step.

Remark 2.1 While this random surfer model provides a reasonable motivation for the Markov Chain approach, using it in computation should not be taken as implying that the model is an accurate representation of actual surfers. Nor should it be assumed that modeling the behavior of a human surfer accurately would necessarily result in a good search engine. After all, search engines are supposed to *augment* human search capacity. It is better to consider the Random Surfer model as a decent intuition for a technical hack necessary to make a Markov Chain ergodic.

In terms of matrices, we can express the new process as follows. Let $1_{p \times q}$ denote the $p \times q$ matrix of all ones. The update step from the distribution $\mathbf{p}_k$ (at some time k) to $\mathbf{p}_{k+1}$ can now be written as follows:

$$\mathbf{p}_{k+1} \quad = \quad (1-\epsilon)M^\intercal \cdot \mathbf{p}_k + \epsilon \cdot \tfrac{1}{n} 1_{n \times 1} \quad = \quad ((1-\epsilon)M^\intercal + \epsilon \cdot \tfrac{1}{n} 1_{n \times n}) \cdot \mathbf{p}_k \quad =: \quad M' \cdot \mathbf{p}_k.$$

To show that this new version of PageRank converges, we can use the following well-known theorem about Markov Chains.

Theorem 2.2 *If the Markov Chain M is irreducible and aperiodic (the gcd of all cycle lengths is* 1*), then it converges to a unique* stationary probability *π, i.e., $\pi = M^{\intercal} \cdot \pi$.*

The term "irreducible" means that the corresponding graph (of non-zero entries of M) is strongly connected. Certainly, our matrix M' is both irreducible and aperiodic, as we added a jump probability of ϵ/n between any pair of vertices, i.e., the graph is now complete. (Notice that for M that are not aperiodic, oscillations might occur, as in the example mentioned above of a cycle of length 2. Similarly, the probabilities of the random walk may depend on where the random walk started. This also applies if M is not irreducible. On the other hand, if M is irreducible and aperiodic, then the probabilities are independent of the staring vector.)

Remark 2.3 For students who have seen this before, a very simple Markov Chain coupling argument (about the simplest possible) shows that the mixing time of this chain is $1/\epsilon$, i.e., a constant. Simply couple two copies of the chain as follows: for both chains, use the same random choice for whether to jump randomly or follow an edge. If following an edge in both, make the choices independently (unless the chains are already in the same state, in which case make the same choice). If jumping uniformly, couple the choices to have both chains jump to the same node. It is easy to see that each chain individually is a faithful copy of the original Markov Chain M'. Also, the expected time until the chains jump uniformly is $1/\epsilon$, and after that point, both chains will always be in the same state. Thus, the expected time to couple is constant.

As a result of the constant mixing time, the convergence is exponentially fast: within $O(\log \frac{1}{\delta})$ iterations, the error is at most δ. Regarding computation, we are now dealing with a very large and dense matrix. However, the fact that most entries are equal to ϵ/n allows for efficient matrix computation techniques to be applied nevertheless.

The stationary probability π of the matrix M' is again the top eigenvector, much like for the HITS algorithm. Thus, in principle, we could apply techniques other than the original update rule (which again corresponds to the power iteration). However, given the exponentially fast convergence of power iterations, there is really no need for other techniques. Of course, in practice, to perform computations of this size, several computational tricks need to be applied.

We have not yet described how the PageRank algorithm uses the values $\mathbf{p} = \pi$. At query time, the set of candidate pages is narrowed down by using text-based matches, as well as several other heuristics. The other pages are then simply sorted by decreasing p_i values. (In practice, Google uses significantly more heuristics to determine the ordering, but the above is a first approximation of the original approach in the actual search engine.)

The attentive reader will have noticed two differences between the descriptions of PageRank and HITS: using the adjacency vs. co-citation matrix, and using the entire web graph vs. a focused subgraph. These two issues are really orthogonal, and one can consider combining the "focused subgraph" of Section 2.2 with computing the PageRank measure at query time. The result would likely yield more relevant query results than plain PageRank. However, it would require recomputing the corresponding PageRank values at query time, which is not feasible for practical applications.

If an algorithm such as HITS or PageRank does use focused graphs, there are several more natural heuristics by which one can improve the quality of search results. For instance, different weights can be associated with different links, depending on how much relevant text (e.g., query terms) appears close to the link's anchor. Similarly, different weights can be assigned to different pages based on content-based similarity to the query topic. Both Chakrabarti et al. [79] and Bharat and Henzinger [44] suggest various heuristics to use the content of pages to alter the matrix, and obtain improved search results. Chakrabarti et al. [79] found that most of the relevant text of links occurs between 25 bytes before and 50 bytes after the anchor, and suggest taking such weights into account. Bharat and Henzinger [44] suggest using the first set of textual matches as a "comparison point" for the expanded focused graph, and prune or weigh down pages which are very different.

2.4 Topic-sensitive PageRank

As discussed above, computing topic-specific PageRank values for each query at query time is too time-consuming to work on the scale required by search engines nowadays. On the other hand, pre-computing the PageRank values for all queries and storing them is too memory-intensive. Using the same PageRank values regardless of the specific query can yield unsatisfactory results.

A middle path has been proposed by Haveliwala [197]. The idea is to precompute PageRank values for a few "landmark" topics, and then express queries as a combination of these topics. Calculating the PageRank values from the pre-computed ones may then be significantly easier than recomputation from scratch.

Assume that there are T topics, and each topic t is characterized by its restart probability vector $\mathbf{f}_t$, i.e., the probability distribution (over pages) with which a new random page is chosen when the Markov Chain jumps. The idea of using topic-specific restart probabilities was proposed by Rafiei and Mendelzon, who suggest jumping to a uniformly random page containing the search term [332]. (The context of their work is computing topic-specific PageRank scores to find out which topics a page is particularly known for.) The corresponding update rule is then $\mathbf{p}' = (1-\epsilon)M^{\mathsf{T}} \cdot \mathbf{p} + \epsilon \cdot \mathbf{f}_t$. By defining the matrix $F_t = [\mathbf{f}_t \mathbf{f}_t \ldots \mathbf{f}_t]$, we can express this new rule as

$$\mathbf{p}' \quad = \quad ((1-\epsilon)M^{\mathsf{T}} + \epsilon \cdot F_t) \cdot \mathbf{p} \quad =: \quad M_t \cdot \mathbf{p}.$$

The corresponding PageRank values for topics t are now the stationary probabilities $\pi_{\mathbf{f}_t}$ of M_t.

From the PageRank values for topics t, we can compute those for queries q as follows. We can consider a query as a convex combination of topics, i.e. $q = \sum_t \gamma_t \cdot t$ (where $\sum_t \gamma_t = 1$). Then, we can identify with q the reset vector $\mathbf{f}_q = \sum_t \gamma_t \mathbf{f}_t$. The interesting observation is that the corresponding stationary probabilities $\pi_{\mathbf{f}_q}$ can be obtained as convex combinations of the probabilities for the landmark topics:

Lemma 2.4 *For each query q with $\mathbf{f}_q = \sum_t \gamma_t \mathbf{f}_t$, we have $\pi_{\mathbf{f}_q} = \sum_t \gamma_t \cdot \pi_{\mathbf{f}_t}$.*

Proof. We will show that the vector on the right is stationary for M_q. As the stationary distribution is unique, this proves that it is equal to $\pi_{\mathbf{f}_q}$. In order to do so, we use the fact that each $\pi_{\mathbf{f}_t}$ is stationary for its corresponding M_t, and then use the linearity of matrix-vector multiplication and summation:

$$\begin{aligned}
\textstyle\sum_t \gamma_t \cdot \pi_{\mathbf{f}_t} &= \textstyle\sum_t \gamma_t \cdot ((1-\epsilon)M^{\mathsf{T}} \cdot \pi_{\mathbf{f}_t} + \epsilon \cdot \mathbf{f}_t) \\
&= (1-\epsilon)M^{\mathsf{T}} \textstyle\sum_t \gamma_t \cdot \pi_{\mathbf{f}_t} + \epsilon \sum_t \gamma_t \mathbf{f}_t \\
&= (1-\epsilon)M^{\mathsf{T}} \textstyle\sum_t \gamma_t \cdot \pi_{\mathbf{f}_t} + \epsilon \mathbf{f}_q \\
&= M_q \cdot (\textstyle\sum_t \gamma_t \cdot \pi_{\mathbf{f}_t}). \qquad \blacksquare
\end{aligned}$$

Hence, the PageRanks of linear combinations of topics can be efficiently computed from precomputed topic PageRanks.

2.4.1 Single-Word Queries

To go even further, one could try to pre-compute the PageRanks for every single-word query. At first, this may seem very daunting, as the number of words is far in excess of 100000, and hence, it appears as though the storage requirement would be larger than $10^5 \cdot 10^9 = 10^{14}$ PageRank values. However, more careful indexing may reduce this requirement significantly, as most words do not appear in most pages. A simple back-of-the envelope calculation observed by Domingos and Richardson [130] goes as follows. Let w denote a word, and i a page. Further, let p_w be the number of pages containing w, and s_i the number of words contained in page i, and $x_{w,i} = 1$ if word w appears in page i. Then, the total required index size is

$$\textstyle\sum_w p_w \quad = \quad \sum_{w,i} x_{w,i} \quad = \quad \sum_i s_i.$$

Notice that the average page contains only about a few hundred words, so the last sum is only about a few hundred times the number n of pages in the web. While this is not yet quite feasible, it is not too far removed from current technology.

For multi-word queries, the situation is a lot more bleak: the number of distinct pairs that appear in web pages is much higher, and the same kind of simplistic analysis does not work: the sum now becomes $\sum_i s_i^2$, which may be much larger. Devising good precompuation techniques for multi-word queries seems like a promising direction for further research.

2.5 Text-Based Information Retrieval

We have so far hinted several times at the importance (and difficulty) of text-based information retrieval. Clearly, it plays a key role in web search. It also turns out that some of the core techniques in information retrieval use ideas similar to the eigenvector-based ones we discussed for HITS and PageRank.

The most naïve search strategies for text-based information retrieval would just return pages that contain the query term(s). There are two problems with this approach: synonyms (different words having the same meaning) and polysemy (the same word having more than one meaning). Synonyms may lead the search engine to miss relevant pages, because the exact query terms may not appear in all relevant pages. Polysemy may lead the search engine to return irrelevant pages; the pages may contain the search term, but in a different context and with a different meaning than the user intended.

Consider the following table. There are 6 pages, and 6 words occurring on the pages. Page 1 contains words 1, 2, and 3, and so on. Imagine searching these pages for word 3. A naïve search engine would simply return pages 1, 2, and 4, because these pages all contain the query word. However, we might argue that page 3 is also relevant: while it does not contain the exact query term, it is very similar to pages 1 and 2 that do contain the query term. Similarly, page 4 may not be considered (as) relevant: while it does contain the query word, it is very similar to pages 5 and 6 that do not contain the query word.

	Word 1	2	3	4	5	6
Page 1	x	x	x			
2	x	x	x			
3	x	x				
4			x	x	x	x
5				x	x	x
6				x	x	x

To put these observations on a more rigorous and general footing, we can use techniques from Spectral Analysis of Data [26], which is also known as Latent Semantic Analysis [121] in the Information Retrieval community. We consider the table as a matrix, where the cells with an 'x' are 1, and the cells without an 'x' are 0. This matrix is called the *term-document matrix.*

In general, this matrix can be used for representing a variety of data sets, where rows index objects in the data set, columns index attributes of those objects, and the $[i, j]$ entry of the matrix represents the value of the j^{th} attribute in the i^{th} object. Some examples of interest are where both rows and columns refer to web sites and the $[i, j]$ entry indicates that site i has a link to the site j; another is that rows index individuals, columns index products, and the $[i, j]$ entry indicates whether individual i is interested in, or has previously purchased, product j.

The main tool in extracting the latent structure from the matrix A is the *singular-value decomposition* (see, e.g., [208]):

Lemma 2.5 *Let A be an $m \times n$ matrix (e.g., document-term matrix). Then, A can be written as $A = U \cdot \Sigma \cdot V^{\mathsf{T}}$, where U is an $m \times k$ orthonormal matrix, V is an $n \times k$ orthonormal matrix ($k = rank(A)$), and*

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sigma_k \end{bmatrix}$$

is a $k \times k$ diagonal matrix with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k > 0$. This is called the singular value decomposition (SVD) of A, and the σ_i are called the singular values.

Let us look more closely at the entries of A. We can write the entry $a_{i,j} = \sum_{\ell=1}^{k} u_{i,\ell} \cdot \sigma_\ell \cdot v_{j,\ell}$. We can consider each $\ell = 1, \ldots, k$ to be a "concept". Then, we can interpret $u_{i,\ell}$ as "how much is page i about concept ℓ", and $v_{j,\ell}$ as "how much does word j belong in concept ℓ", and σ_ℓ as the "importance" of concept ℓ. Note that in representing A in this way, we are making the implicit assumption that concepts behave linearly: the frequency with which a word occurs on a page is the *sum* over all concepts, and there are no superlinear amplifications, or sublinear reductions.

Row i of U can be seen as a k-dimensional description of page i. Concept $\ell = 1, \ldots, k$ corresponds to a column of U (or a column of V), so the k-dimensional description of i expresses it in terms of the concepts. Notice that the same column in U and V correspond to the same concept. Also, notice that since U and V are orthonormal, concepts are orthogonal.

Following our intuition, we would expect "similar" pages to have similar concept vectors (again, this is predicated on the assumption that concepts behave linearly). We can then measure the similarity of two pages by comparing their concept vectors. If there is a small angle between the vectors (or a large inner product), then the two pages are about the same (or very similar) concepts. However, up to this point, we are still simply using all the data in the matrix A: merely rephrasing it in terms of the SVD does not yet give improved search results.

The important issue which we wanted to address was that the entries of the term-document matrix were not derived from an ideal generation process materializing the concepts in the form of terms. Rather, real matrices have "errors". More formally, if the world were about $k \ll \min(m, n)$ concepts, then an "ideal world" matrix would have rank k. In the real world, web pages do not conform to our ideal concepts. People write web pages, and their individual tendencies and writing styles vary. This will cause A to have rank (almost) $\min(n, m)$. The concepts $\ell = k + 1, \ldots, \min(n, m)$ are "error-correcting" concepts explaining peculiarities of A. Hence, to get at the "true" contents of pages, we would like to prune out the entries derived from those concepts.

How do we do this? First, we determine the "right" k. Having computed the SVD $A = U \cdot \Sigma \cdot V^\intercal$, we "blank out" all concepts for $\ell > k$, by defining

$$\Sigma_k = \begin{bmatrix} \sigma_1 & 0 & & \cdots & & 0 \\ 0 & \ddots & 0 & & & \\ & 0 & \sigma_k & 0 & & \vdots \\ \vdots & & 0 & 0 & & \\ & & & & \ddots & \\ 0 & & \cdots & & & 0 \end{bmatrix}$$

and $A_k = U \cdot \Sigma_k \cdot V^\intercal$. In effect, that gets rid of all concepts for $\ell > k$, and thus only retains $U_k = [\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_k]$ and $V_k = [\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_k]$. (The $\mathbf{u}_i$ and $\mathbf{v}_i$ are the columns of U and V, respectively.) The resulting matrix A_k is still an $m \times n$ matrix, but it now has rank k. The following lemma (see, e.g., [208]) shows that it approximates A well.

Lemma 2.6 *The matrix A_k is the best rank-k approximation to A: it minimizes $\|A - B\|_2$ over matrices B of rank k, where $\|A - B\|_2 = \max_{\|x\|_2=1} \|(A - B) \cdot x\|_2$.*

Given a document-term matrix A and a query q, we would like to find the pages in A that are most relevant to q. First, we compute A_k (for some choice of k). We consider the space $\mathbb{R}^k$ as the "concept space", into which we can map both words and pages based on their rows in U and V. Specifically, each page i is identified with the i^{th} row of U_k, and each word j with the j^{th} row of V_k. To find pages relevant to the query, we simply output the pages closest to q in concept space.

How do we choose a good k? A good k is one with $\sigma_k \gg \sigma_{k+1}$. If no such k exists, then A cannot be approximated well with low rank, although we will still want to choose some k — it just does not come with good guarantees. In the end, choosing the right k is somewhat of a heuristic.

The result of this approach is that we can (hopefully) extract meaningful concepts, and thus identify two pages as similar, and relevant to a query, even if the terms occurring in them are quite different. Conversely, even if a search term occurs in a document, we may be able to filter it out as irrelevant if other terms force it to lie far away in concept space from the point at which our query is located.

Similar ideas can be applied to several other scenarios:

- Collaborative Filtering (or Recommendation Systems), i.e., the reconstruction of the missing data items. Finding similarities between users (again, in some "concept space") lets us predict which unseen items a user might like.
- Web site ranking.
- Shopping Predictions.

Singular-value decomposition has found similar applications in a number of domains. In different fields, it (or close variants) is often also called *Factor Analysis*, *Latent Semantic Analysis*, or *Principal Component Analysis*.

2.6 Further Reading

Two surveys by Bianchini et al. [47] and by Langville and Meyer [258] provide in-depth discussions of many of the mathematical, algorithmic, and technological challenges involved in link-based search engines. The survey by Langville and Meyer has meanwhile been expanded into a comprehensive book [259].

Some of the technical infrastructure underlying successful search engines is described in [42, 170]. The focus in this chapter, on the other hand, is on link-based inferences about relevance and content, and more specifically spectral methods in this context.

A good, though somewhat outdated, overview of spectral techniques in link-based web search is given by Borodin et al. [58, 59], Among the many suggested changes or improvements in HITS and PageRank are the following: SALSA [265] performs a random walk on the co-citation graph, but normalizes probabilities. Thus, high-degree hubs *divide* their hub weight among the authorities they link to, rather than giving full hub weight to each.

An idea similar to PageRank was around the same time proposed by Marchiori [276], building on previous work by Frisse [165] for singly authored hypertext corpora. The relevance of i is its own content, plus a weighted sum of the contents of pages reachable from i. The weights decrease exponentially, and if page i has multiple outgoing edges, the assumption is that they will be visited in the order maximizing the relevance of i. A similar idea is used by Boyan et al. [60], who also aggregate relevance scores of pointed-to pages with geometrically decreasing weights. In addition, [60] proposes taking into account user feedback to train better retrieval functions and learn the weights for different features. Using *clickthrough data* (information on which pages were or were not clicked by the searcher) was pursued much more extensively subsequently by Joachims et al., for example [219, 331].

The idea of using eigenvalues as measures of "importance", "authority", or "status" long precedes HITS and PageRank. Katz [228] proposes, and Bonacich [55] elaborates on, an approach similar to PageRank to measure social status in a social network: the status of v is the number of paths from other nodes to v, with path weights exponentially decreasing in the length of the path. This is similar to giving a node a weight derived from the (discounted) sum of weights of its neighbors. Hubbell [209] uses an essentially mathematically equivalent idea to derive a matrix measuring how strongly pairs of nodes are connected. This matrix is then used to identify "cliques" as densely connected subsets of nodes (differing from the mathematical definition of a clique). Pinski and Narin [325] use a technique very similar to PageRank to measure the importance or influence of scientific publications.

Finding authoritative pages on particular topics is also called *topic distillation*. Many heuristics have been proposed to avoid *topic drift* (which happens when a majority of the pages in the focused subgraph are not exactly matching the query) and undue influence of individual nodes. In addition to the two papers discussed above [44, 79], many others propose different heuristics. For instance, Chakrabarti et al. [81] suggest analyzing pages at a more fine-grained level of individual document objects of the page.

In Sections 2.2 and 2.3, as part of our analysis of HITS and PageRank, we also discussed convergence speeds, and remarked that it depends crucially on the spectral gap, the difference between the two largest eigenvalues. By introducing the random jump with probability ϵ, PageRank ensures that the spectral gap is small, and the Markov Chain converges quickly. An experimental study of convergence speeds of PageRank computations is performed by Arasu et al. [19]. They find that PageRank converges faster on the graph of hosts than of pages, and that the Gauss-Seidel method for computing eigenvectors converges faster than the Power Iteration. The convergence speed of PageRank-like computations has received a lot of attention in the numerical analysis community. However, as observed by McSherry [280], performing computations on the right block structure of matrices and reusing values appropriately are sufficient to run a full PageRank computation on an ordinary laptop. At this point, the technological challenges of computing PageRank values (offline) can therefore be considered solved.

A large spectral gap has a second advantage beyond fast convergence: the principal eigenvector does not change much when the matrix is perturbed slightly, e.g., when links are added or deleted, or when rounding errors occur during the computation. In this sense, PageRank is more stable than HITS, a fact pointed out by Ng et al. [316]. Based on this observation, in subsequent work [317], they suggest augmenting HITS by a random jump with probability ϵ as well. Alternatively, one can consider the top k eigenvectors, up to a point where the eigenvalues drop significantly in magnitude. Such an eigenspace would be more stable. For an in-depth discussion of the effects of perturbations on matrix computations, see [180, 357].

In order to analyze formally the empirically observed performance of LSI and related methods, one needs to describe a model of the underlying "ground truth", which is usually a randomized model for generating page content. In this context, Papadimitriou et al. [320] were the first to propose a model under which LSI provably performs well: basically, their model posited that different topics define distributions over terms, and pages are drawn from a distribution of topics. If there were few topics with sufficiently distinct signatures, LSI can recover the latent linear structure.

Similar approaches were applied in the context of web search, where generational models also include the links. For instance, a model of Cohn and Chang [103], based on a similar idea of Hoffman [205] for term-document scenarios, assumes that links are generated probabilistically with topic-dependent probabilities. From the sites that a page links to, a maximum likelihood estimation then tries to infer the actual topics. Cohn and Hofmann [104] extend this model to include both terms within the page and links. Achlioptas et al. [4] take this approach yet one step further: for each page, an (unknown) low-dimensional vector describes to what extent the page is a hub and an authority on each of k topics. Text on a page is generated according to distributions parametrized by the topics, and linking probability depend on the authority weights of the target and the hub weights of the source of the link. The algorithm then uses an appropriate SVD to infer the low-dimensional vectors for pages.

The linearity constraint implicit in the use of SVD techniques has been recognized by the community. As a result, standard PCA has been extended to *Kernel PCA* (see, e.g., [348]). The idea is that the standard inner product (by which one implicitly assumes entries of the matrix to be computed) can be replaced by different notions of inner products. The Kernel of Kernel PCA techniques provides such a notion. Using these techniques requires knowing what inner product is appropriate for the problem at hand.

Implicit in essentially all of the work described in this chapter is the assumption that pages which are "close" in terms of number of hops on the web should be for the most part "similar" in terms of content or topic. Davison [115] and Menczer [282] explicitly test this hypothesis and mostly find it to be true.

To ensure the assumption that links express some notion of endorsement, it is important to deal with *nepotistic links* and *link spamming*. Nepotistic links include navigational aids as well as link spam. Several papers attempt to automatically identify such links. Davison [114] uses machine learning techniques to identify decision tree rules for recognizing nepotistic links from examples. The work of Bharat and Henzinger [44] also implicitly discounts nepotistic links. Zhang et al. [392] argue that the PageRanks of nodes with large numbers of nepotistic links are particularly sensitive to the reset parameter ϵ in PageRank, and can heuristically be identified by varying ϵ.

Despite the intuitive mathematical justification of the search algorithms presented here, the ultimate judgment is the relevance of the results returned. Naturally, in this context, the success of search engines like Google or Yahoo! constitutes some amount of verification. A principled experiment was attempted by Amento et al. [15], who asked 40 human users for the most authoritative documents, and then compared with the results of search algorithms, as well as various features. They found that link-based analysis yields good results overall, but surprisingly, merely the number of pages co-hosted with the given page is a very good predictor as well.

Chapter 3

Community Structure and Clusters in Graphs

In this chapter, we turn to the problem of identifying *communities* from link information. A community is defined informally as a cohesive set of nodes, in the sense that the node set has “many” links within, and “few” links connecting it to other nodes. Of course, “many” and “few” are still not very clearly defined terms, and we will see below how different instantiations of these concepts will lead to different objectives.

But first, let us turn to the question of *why* we are interested in identifying community structure in networks. The most important reason is that community structure may reveal useful information about the network. In a network of friendships, we may identify a large groups of tightly knit friends. In a terrorist network, with edges representing phone calls or other interactions, we may identify a cell. In a graph such as the WWW, we might discover communities of pages pertaining to the same specialized subject. (Here, we would be using the implicit assumption of *homophily*: that pages on similar topics are more likely to link to each other.) For another example, many biologists believe that if a set of proteins contains many pairwise interactions, then these proteins are often involved in some specific common function. In summary, the most important benefit of identifying communities based on links is the functional or content predictions one can obtain from the structure.

Two side benefits may not be as readily apparent. Identifying high-level community structure may permit studying a network at a more appropriate level of detail. This could include replacing communities with one or few meta-nodes, and studying the network at the level of those meta-nodes. Or one could zoom in on one particular community, disregarding other parts of the network. For instance, Dill et al. [127] found empirically that the WWW graph exhibits a lot of self-similarity: within communities defined based on common topics or other sufficiently homophilous criteria, degree distributions and other statistics closely resembled those of the entire web graph. A fairly detailed discussion of these and other motivations can be found, for instance, in Newman’s paper [311].

A further application of community identification is the following: often, social scientists are particularly interested in links that *cross* between communities. Granovetter [188] calls such links *weak ties*. The particular importance of weak ties is that they provide individuals with much more “fresh” information than strong ties to their good friends, presumably because the close friends share the same social circles, and therefore do not have as much novel information. Granovetter [188] reports on data showing that weak ties were much more instrumental than strong ties in helping individuals find jobs. Further corroborating the importance of weak ties in connecting social networks was an experiment analyzing a high-school friendship network. The graph representing only the strong ties, where each person is linked only to their two best friends, contains many small, isolated communities. On the other hand, including links to the top eight friends (which will include weaker ties) results in a graph with a giant connected component, and only a few smaller clusters.

The importance of links crossing between different communities has meanwhile been reiterated and stressed in many other papers. Most notably, Burt (e.g., [73, 74]) advances a theory of *structural holes*, arguing that the existence of distinct communities tends to imply entrepreneurial opportunities for individuals or companies bridging those communities.

In order to pose questions about weak links and structural holes mathematically, and analyze models for their formation or function, it is first necessary to *identify* communities as such. This would also allow us to formalize the problem of identifying weak links or structural holes.

For the remainder of this chapter, we will use the following notation.

Definition 3.1 *Let $G = (V, E)$ be a graph.*

1. *$e(S, T) = E \cap (S \times T)$ denotes the set of edges with exactly one endpoint in S and one in T. $e(S) = e(S, S)$ is the set of edges with both endpoints in S.*
2. *We use $d_S(v) = |e(\{v\}, S)|$ to denote the degree of v within the set S, i.e., number of edges between v and nodes in S.*
3. *The* edge density *of a node-set S is defined as $\frac{|e(S)|}{|S|}$.*

In exploring the spectrum of different definitions of the term "community", there are several parameters along which we can distinguish definitions:

1. Can nodes be in multiple communities? Does each node have to be in at least one community? If these questions are answered "No" and "Yes", respectively, then we are seeking a *partition* of the nodes. As such, the problem of identifying community structure shares a lot of overlap with the problem of *clustering*, and we will explore several objective functions in Sections 3.5 and 3.6.
2. If the answers are "Yes" and "No" instead, then we are looking to identify individual, possibly overlapping communities. These will be "unusually dense" sets of nodes. Here, an interesting distinction is whether we want to include nodes which have overall high degree, or focus on nodes which have "most" of their edges within a set. This distinction will give rise to different definitions in Sections 3.1 and 3.2.
3. If we follow the reasoning about hubs and authorities of Section 2.2, we would suspect that in the web graph, communities could be identified by dense bipartite cores of hubs and authorities. Heuristic approaches based on these ideas are discussed in Section 3.3.

3.1 Dense Subgraphs

Perhaps the simplest and most "obvious" definition of a community is a dense subgraph, i.e., a subgraph with large edge density in the sense of Definition 3.1. If we want to find the *best* community, that would be the subgraph with largest density.

Problem 1 *In an arbitrary graph G, find the densest subgraph S, i.e., the set S maximizing $\frac{|e(S)|}{|S|}$.*

We begin with the decision version of the problem: Given a constant α, is there a subgraph S with $\frac{|e(S)|}{|S|} \geq \alpha$? This constraint can be re-written as:

$$|e(S)| - \alpha|S| \quad \geq \quad 0. \tag{3.1}$$

Each internal edge of S contributes 2 to the total degree in S, and each edge leaving S contributes 1, so the number of internal edges $|e(S)|$ in S is $|e(S)| = \frac{1}{2}(\sum_{v \in S} d(v) - |e(S, \overline{S})|)$. Substituting this in Equation (3.1) gives us

$$\sum_{v \in S} d(v) - |e(S, \overline{S})| - 2\alpha|S| \quad \geq \quad 0, \tag{3.2}$$

or, equivalently,

$$\underbrace{\sum_{v \in V} d(v)}_{2|E|} - \underbrace{(\sum_{v \in \overline{S}} d(v) + |e(S, \overline{S})| + 2\alpha|S|)}_{=:\beta(S)} \quad \geq \quad 0. \tag{3.3}$$

As $2|E|$ is a constant (independent of S), this constraint is satisfiable iff it is satisfied by the set S with minimum $\beta(S)$. To find such a set, we formulate a mincut problem as follows. Consider the graph G' with $V' = V \cup \{s, t\}$, where s is connected to all vertices v with an edge of capacity $d(v)$, and t is connected to all vertices in V with an edge of capacity 2α. The cost of the cut $(S + s, \overline{S} + t)$ in G' is exactly $\beta(S)$. Thus, if the minimum s-t cut $(S + s, \overline{S} + t)$ of G' satisfies the constraint in Equation (3.3), then S is a set of density at least α. If not, then no such set exists. The minimum s-t cut can of course be computed with any of the standard Mincut algorithms (see, e.g., [246]).

Of course, our real goal was to find the *maximum value* of α for which a set S of density α exists. We could accomplish this with a binary search over values of α. A more efficient way uses the *parametric max-flow* algorithm of Gallo, Grigoriadis and Tarjan [168] to compute minimum s-t cuts for *all* values of α in one computation. Since this is a fairly general and useful technique, we state the main result here as a theorem:

Theorem 3.2 (Parametric Maximum Flow [168]) *Let α be a real-valued parameter, and $G = (V, E)$ a graph with non-negative edge capacities c_e which can depend on α in the following way:*

1. *Capacities of edges out of s are non-decreasing in α.*
2. *Capacities of edges into t are non-increasing in α.*
3. *Capacities of edges not incident with s or t are constant.*

Then, the maximum flow and minimum cut for all values of α can be computed simultaneously with one invocation of a modified Goldberg-Tarjan algorithm [176], in time $O(n^2m)$. Furthermore, the minimum cuts $(S_\alpha, \overline{S_\alpha})$ are nested*, in the sense that $S_\alpha \subseteq S_{\alpha'}$ whenever $\alpha \leq \alpha'$.*

Remark 3.3 While we gave an algorithm for the problem of finding the overall densest subgraph, the problem becomes significantly more difficult once we restrict the size of the subgraph. For instance, finding the densest subgraph of exactly (or at most) k nodes is an NP-hard problem, as can be seen easily by reducing from the k-clique problem, setting the density to be $k - 1$.

The densest k-subgraph problem has been studied in several papers [23, 46, 151, 154]. The current best approximation ratio is $O(n^{1/4+\epsilon})$ (in time $O(n^{1/\epsilon})$ for any $\epsilon > 0$ [46].

Only recently did Khot [238] rule out the existence of a PTAS for the densest k-subgraph problem: unless the Unique Games Conjecture [237] is violated, there is some $\alpha > 1$ such that no α-approximation to the densest subgraph can be found in polynomial time. Obviously, the upper and lower bounds are far from matching, and closing this gap is an interesting open question.

A straightforward, but quite useful, generalization of the Densest Subgraph problem specifies a set X of vertices that must be included in the set S.

Problem 2 *Given a graph G, find the densest subgraph S containing a specific vertex set X (i.e., S maximizes $\frac{|e(S)|}{|S|}$ over all sets $S \supseteq X$).*

This allows us to find out what the nodes in X "have in common", by identifying a dense subgraph S containing them, and then inspecting the nodes in the dense subgraph for their attributes.

This generalization can be solved quite easily, simply by giving all edges from s to vertices $v \in X$ a capacity of ∞. This will ensure that all nodes in X are on the s-side of the cut, and the rest of the analysis stays the same.

3.1.1 A 1/2-Approximation

In some real-world graphs, such as the WWW or the graph of all friendships among people in the US, the running time of $O(mn^2)$ from Theorem 3.2 is still prohibitively large. Thus, we are interested in faster algorithms, preferably running in linear or near-linear time. As it is not known how to compute minimum s-t cuts much faster, we will have to settle for an approximation algorithm. The following greedy algorithm was first analyzed by Charikar [84]. We present it immediately for the generalized version (Problem 2) of the Densest Subgraph Problem.

Algorithm 1 A Greedy $\frac{1}{2}$-Approximation Algorithm for finding dense subgraphs

```
Let $G_n \leftarrow G$ .
for $k = n$ downto $|X| + 1$ do
    Let $v \notin X$ be the lowest degree node in $G_k \setminus X$.
    Let $G_{k-1} \leftarrow G_k \setminus \{v\}$.
Output the densest subgraph among $G_n, \ldots, G_{|X|}$.
```

Claim 3.4 *Algorithm 1 is a $\frac{1}{2}$-approximation.*

Proof. Let $S \supseteq X$ be the densest subgraph. If our algorithm outputs S, then it is clearly optimal. If not, then at some point, we must have deleted a node $v \in S$. Let G_k be the graph right before the first $v \in S$ was removed. Because S is optimal, removing v from it would only make it worse, so

$$\tfrac{|e(S)|}{|S|} \geq \tfrac{|e(S-v)|}{|S|-1} \geq \tfrac{|e(S)|-d_S(v)}{|S|-1}.$$

Multiplying through with $|S|(|S|-1)$ and rearranging gives us $d(v) \geq \frac{|e(S)|}{|S|}$.

Because G_k is a supergraph of S, the degree of v in G_k must be at least as large as in S, so $d_{G_k}(v) \geq d_S(v) \geq \frac{|e(S)|}{|S|}$. The algorithm chose v because it had minimum degree, so we know that for each $u \in G_k \setminus X$, we have $d_{G_k}(u) \geq d_{G_k}(v) \geq \frac{|e(S)|}{|S|}$. We thus obtain the following bound on the density of the graph G_k:

$$\begin{aligned}
\frac{|e(G_k)|}{|G_k|} &\geq \frac{\sum_{u \in S} d_S(u) + \sum_{u \in G_k \setminus S} \frac{|e(S)|}{|S|}}{2|G_k|} \\
&= \frac{2|e(S)| + |G_k \setminus S| \frac{|e(S)|}{|S|}}{2|G_k|} \\
&\geq \frac{|e(S)|}{|S|} \cdot \frac{|S| + |G_k \setminus S|}{2|G_k|} \\
&= \frac{|e(S)|}{2|S|}.
\end{aligned}$$

The graph that the algorithm outputs is certainly no worse than G_k, as G_k was available as a potential solution. Hence, the algorithm is a $\frac{1}{2}$-approximation. ■

3.2 Strong Communities

In the previous section, we defined a community as a dense subgraph. That means that the group of nodes overall has many edges. One feature of this definition is that it will "usually" favor the inclusion of nodes with overall high degree. For instance, in terms of communities in the WWW, we would expect that for any sufficiently large set S of nodes, the most commonly linked to other nodes would be yahoo.com, google.com, or cnn.com, even if the nodes in S do share a much more narrow feature. In terms of our introductory discussion, we were requiring many edges within the community, but not few edges connecting it to other nodes. For some applications, one could argue that communities should also be well separated from other nodes, in the sense that each node has most of its links inside the set S. This leads us to the following definition, slightly generalizing one by Flake et al. [156].

Definition 3.5 *Let G be a graph, and $\alpha \in [0, 1]$. A set $S \subseteq G$ is called a* strong α-community *in G iff $d_S(v) \geq \alpha d(v)$ for all nodes $v \in S$.*

This definition captures that each individual should "belong" to the community. The value $\alpha = \frac{1}{2}$ has a particularly natural interpretation, in that it requires that each node in S have at least as many edges inside S as to nodes outside S. It is also the case considered in [156]. If we omit the value of α, we assume $\alpha = \frac{1}{2}$.

The larger α, the more "tightly knit" the community is. So the best communities are the ones with large α. However, this leads to the problem that the whole graph is the "best" community, because all of its edges are within the selected set (itself), making it a 1-community.

The entire graph is not a very interesting community. We are more interested in discovering significantly smaller communities which nevertheless have many edges inside. As an extension, we may also wish to force the inclusion of one or more nodes, and find the best community containing them.

Unfortunately, finding the smallest (or even approximately smallest to within a factor of $c \log(n)$ for some constant c) community is NP-hard, with or without the inclusion of specific sets.

Theorem 3.6

1. *It is NP-complete to decide whether a given graph G has an α-community of size at most k.*
2. *Unless P=NP, the size of the smallest α-community cannot be approximated within a factor of $O(c \log n)$ in polynomial time, for some c.*
3. *The same results hold if we require to find a community of a specified vertex v.*

Proof. Membership in NP is obvious. The NP-hardness will be implied by the approximation hardness, so we will only prove the second part of the theorem (and show the few simple extensions necessary to prove the third). To avoid notational clutter, we assume here that $\alpha = \frac{1}{2}$.

We reduce from the Set Cover problem. Let $X = \{x_1, \ldots, x_n\}$ be the set of elements, and $S_1, \ldots, S_m \subseteq X$ the sets. The goal is to find a minimum number of sets S_i whose union covers X. To do so, we construct an undirected and unweighted graph G. Let X_i for an element x_i denote the set of all indices j such that $x_i \in S_j$.

We start with a large clique B of $4mn^2 + 1$ *bulk nodes.* These will be used solely to increase the degree of other nodes and force them to include many of their more "legitimate" neighbors — we don't even need to give them individual names.

We also have a set F of n *forcer nodes*, each of which is connected to n distinct bulk nodes. Forcer nodes are not connected among each other.

The other nodes are more important, and fall into two classes: first, there is an *element node* v_i for each element x_i, and the v_i form a clique. In addition, each v_i is connected to $2n + 1 - |X_i|$ distinct bulk nodes, and to each of the n forcer nodes.

For each set S_j, there are $2n + 1$ *set nodes* $s_{j,a}$, which are connected to form a clique (for fixed j). Each $s_{j,a}$ for $2 \leq a \leq 2n + 1$ is connected to $2n$ distinct bulk nodes, whereas $s_{j,1}$ is connected to $2n + |S_j|$ distinct bulk nodes. Finally, there is an edge between $s_{j,1}$ and v_i whenever $x_i \in S_j$ in the Set Cover instance.

In the resulting graph, each element node v_i has degree exactly $4n$, each set node $s_{j,a}$ for $a \geq 2$ has degree $4n$, the nodes $s_{j,1}$ have degree $4n + 2|S_j|$, forcer nodes have degree $2n$, and the bulk nodes have degree at least $4mn^2$.

Given a set cover of size k, we can obtain a community as follows: Let C consist of all element nodes v_i, all forcer nodes, and all set nodes $s_{j,a}$ for which S_j is in the set cover. The size of C is $k(2n + 1) + 2n$. To verify that it is a community, notice that in the induced subgraph of C, each node $s_{j,a}$ for $a \geq 2$ has degree $2n$, and node $s_{j,1}$ has degree $2n + |S_j|$. Each node v_i has degree at least $2n$, because C contains all other $v_{i'}$, all forcer nodes, and a node $s_{j,1}$ for some set S_j containing x_i.

Conversely, consider a community C of size at most $k(2n+1)+2n$ for some $k < n$ (solutions with $k \geq n$ are of course uninteresting for set cover approximations). Because it is quite small, C cannot contain any bulk nodes — if it contained any one of them, it would have to contain at least $2mn^2 > m(2n+1)+2n$ of them to be a community.

Now, if C contains any set node $s_{j,a}$, it must contain all of the $s_{j,a'}$ for that particular j in order to be a community (otherwise, the degree requirement for $s_{j,a}$ would be violated, as none of its adjacent bulk nodes are included). By the size constraint, C contains the set node clique for at most k sets S_j, and we will prove that these sets S_j form a set cover.

In order to satisfy the degree requirement, all of the v_i adjacent to $s_{j,1} \in C$ must be included — in particular, any community C (of size at most $k(2n+1)+2n$) contains at least one element node v_i.

In order for the element node to have its required degree of $2n$ without bulk nodes, C must contain a forcer node, for C can contain at most n element nodes and $k \leq n-1$ nodes $s_{j,1}$. But in a community C with a forcer node and without bulk nodes, all element nodes v_i must be included to meet the forcer node's degree requirement. Thus, any community C without bulk nodes contains all element nodes.

Now, to meet an element node v_i's degree requirement of $2n$ without including bulk nodes, at least one set node adjacent to v_i must be included in C, for all forcer and element nodes would only yield degree $2n-1$. Hence, the sets S_j for which $s_{j,1} \in C$ indeed form a set cover of size at most k.

This completes the proof that the above reduction is approximation preserving, thus showing the inapproximability of the community problem, because Set Cover is known to be inapproximable within $O(c \log n)$ for some c (unless P=NP) [150, 335].

When we ask about communities including a particular node v, we can use exactly the same reduction, and fix v to be a forcer node — the result is going to be the same, since forcer nodes were just shown to be included in all sufficiently small communities. ■

Given that even approximating the smallest α-community within $O(\log n)$ is NP-hard, we will look at heuristics: approaches that may often work in practice, even if they give us no guarantees. Let $(S, \overline{S})$ be a minimum s-t cut in G. Then, S is almost a $\frac{1}{2}$-community because each $v \in S \setminus \{s,t\}$ has at least as many edges inside S as crossing $(S, \overline{S})$: otherwise moving v to the other side of the cut would make the cut cheaper. If this also held for s and t, then $(S, \overline{S})$ would be a $\frac{1}{2}$-community.

If we are looking for communities including a given node s, we can use the above heuristic approach to compute the minimum s-t cut for all t in n min-cut computations. Then, we simply take the best cut found this way.

If we are looking for just communities, without specifying a node s, then we can try all (s,t) pairs ($\Theta(n^2)$ min-cut computations). We can reduce that number of computations to $O(n)$ using Gomory/Hu trees [183]. The idea is that all min-cuts can be "encoded" in a tree.

Theorem 3.7 (Gomory-Hu Trees [183]) *Let $G = (V,E)$ be a graph. For all node pairs $i,j \in V$, let f_{ij} be the maximum flow (min-cut) between i and j. Let G' be the complete graph on V with edge costs f_{ij}. Let T be a maximum spanning tree of G'.*

For each $i,j \in V$, let P_{ij} denote the (unique) i-j path in T. Then, the tree T has the property that $f_{ij} = \min_{e \in P_{ij}} f_e$ for all i,j. If e is the edge attaining the minimum, then the two connected components of $T \setminus \{e\}$ define a minimum i-j cut in the original graph G.

Proof. We will prove that $f_{ij} = \min_{e \in P_{ij}} f_e$ by contradiction, ruling out inequality in both directions.

If $f_{ij} > \min_{e \in P_{ij}} f_e$, then inserting (i,j) into T and removing the edge $e \in P_{ij}$ minimizing f_e would create a more expensive tree T'. This contradicts the assumption that T was a maximum spanning tree of G'. So $f_{ij} \leq \min_{e \in P_{ij}} f_e$.

For the other direction, let $(S, \overline{S})$ be an i-j cut of capacity f_{ij}. Because P_{ij} is an i-j path, it must cross this cut, i.e., there is an edge $e = (u,v) \in P_{ij}$ with $u \in S, v \in \overline{S}$. So $(S, \overline{S})$ is also a u-v cut, and thus, $f_{uv} \leq f_{ij}$. But then, $\min_{e \in P_{ij}} f_e \leq f_{uv} \leq f_{ij}$, completing the proof. ■

Therefore, the s-t cuts for all $s,t \in V$ can be compactly represented. An interesting additional consequence is that there are only $n-1$ different min-cuts (and associated min-cut values) for the $n(n-1)$

different pairs of nodes. Gomory and Hu [183] also show how to compute T from G using only $n-1$ min-cut computations. Using the approach of first computing T, we can thus find the communities for all nodes with only $O(n)$ min-cut computations.

3.2.1 Other values of α

If we want to find α-communities for $\alpha \neq \frac{1}{2}$, we need to adapt the approach. The idea is to adapt some degrees so that $d_S(v) \geq \alpha d_G(v)$ holds in the original graph if and only if $d'_S(v) \geq \frac{1}{2} d'_G(v)$ holds in the new graph. We do this by adding a source and sink, and connecting them to each node v in an effort to "balance" the sides of the inequality.

If $\alpha < \frac{1}{2}$, each node v has an edge to the source with capacity $(1-2\alpha)d(v)$; otherwise, each node v is connected to the sink with capacity $(2\alpha-1)d(v)$. Nodes x whose inclusion (or exclusion) is required are connected to s (resp., t) with infinite capacity. (Notice that if no nodes are connected with infinite capacity, then the minimum s-t cut will always just separate s or t from the rest of the graph, corresponding to our above intuition that the entire graph is the best community.) The algorithm then just looks for the minimum s-t cut in the resulting graph G'.

Claim 3.8 *This approach will produce "almost α-communities," in the sense that all nodes except those whose inclusion was forced will meet the community constraint.*

Proof. Here, we give the proof for the case that $\alpha \geq \frac{1}{2}$ — the proof for $\alpha < \frac{1}{2}$ is symmetric.

As in the argument for plain min-cuts in a graph, the fact that a node u (without infinite-capacity edges) is on the s-side as opposed to the t-side implies that $d_S(u) \geq d_{\overline{S}}(u) + (2\alpha-1)d(u)$, or $d_S(u) + d(u) - d_{\overline{S}}(u) \geq 2\alpha d(u)$. Now, because $d(u) = d_S(u) + d_{\overline{S}}(u)$, this becomes $2d_S(u) \geq 2\alpha d(u)$, or $\frac{d_S(u)}{d(u)} \geq \alpha$. ■

The sets found by this approach can violate the community constraint at the nodes whose inclusion was forced. Notice that this may happen even when there are communities including/excluding specific nodes, i.e., the fact that the algorithm did not find a community does not mean that none exists.

Flake at al. [156] propose another min-cut based heuristic, which makes more of an effort to avoid the "entire graph" problem described above. For a parameter δ that will be varied, all nodes other than a specified node s to be included are connected to a new sink t with capacity δ. We then vary the parameter δ, and look for the minimum s-t cut. The cuts found this way can then be inspected manually, and interesting ones extracted. (Notice that they can be found with one parametric max-flow computation [168], as described in Theorem 3.2.)

For $\delta = 0$, the minimum cut is $(V, \{t\})$. On the other hand, for very large δ, the minimum cut is $(\{s\}, V \cup \{t\} \setminus \{s\})$. If along the way, some δ yields a non-trivial solution, that is an "almost community," in that it violates the constraint only at the node s.

3.3 Bipartite Subgraphs

The previous views of communities, both of which essentially looked for dense subgraphs, were based on the implicit assumption that nodes within a community are "likely" to link to each other. This homophily often applies in social settings. However, as we argued in Section 2.2, it is a more questionable assumption in competitive settings such as the WWW. For instance, within the "community" of car manufacturers, we would expect relatively few or no links between the most prominent members. At the time, the solution was to look (implicitly) for links in the co-citation graph, or look for bipartite graph structures of hubs and authorities.

Kumar et al. [254] propose the same approach for community identification, and argue that large and dense bipartite graphs are the "signature" of communities in the WWW.

Ideally, we would like to enumerate "all" such signatures, and expand them to communities. However, the complexity of doing so would be prohibitive. If nothing else, deciding the presence of large complete bipartite graphs is NP-hard, and as hard to approximate as the Clique problem itself. However, in this context, it is interesting to note that if the given graph is dense enough (i.e., contains enough edges), it always has a large complete bipartite subgraph.

Lemma 3.9 *If a bipartite graph has $\Omega(b^{1/3}n^{5/3})$ edges, it contains a $K_{3,b}$, i.e., a complete bipartite subgraph with 3 nodes on one side, and b nodes on the other.*

Proof. For each node v on the right side, we write $\delta(v)$ for the set of its neighbors, and $d(v) = |\delta(v)|$ for its degree. Each node v is labeled with each 3-element subset T of $\delta(v)$ (i.e., with all $T \subseteq \delta(v), |T| = 3$). Notice that each node thus has many labels, namely $\binom{d(v)}{3}$. Taken over all nodes on the right side, the total number of labels is $\sum_v \binom{d(v)}{3}$. We assumed in the statement of the lemma that $\sum_v d(v) = \Omega(b^{1/3}n^{5/3})$. The total number of labels is minimized when all $d(v)$ are equal, i.e., $d(v) = \Omega(b^{1/3}n^{2/3})$. Even then, the number of labels is

$$\sum_v \binom{d(v)}{3} \quad = \quad n\binom{\Omega(b^{1/3}n^{2/3})}{3} \quad = \quad n\Omega(bn^2) \quad = \quad \Omega(bn^3). \tag{3.4}$$

But the total number of distinct labels is only $\binom{n}{3} = O(n^3)$. Hence, by the Pigeonhole Principle, some label must appear at least b times. The b nodes on the right side sharing the label, and the three nodes on the left side who form the parts of the label, together form a $K_{3,b}$. ■

By considering a-tuples instead of triples for labels, we can obtain the generalization that any bipartite graph with $\Omega(b^{1/a}n^{2-1/a})$ edges contains a $K_{a,b}$.

While it is interesting to know that sufficiently dense graphs will contain a $K_{a,b}$, it does not necessarily help us in finding one, in particular if the graph is not dense. For large a and b, the problem is NP-hard in general, but we may still be interested in speeding up the search for smaller, and practically important, values, such as finding $K_{3,6}$ graphs. By brute force (trying all 9-tuples of nodes), this would take $\Theta(n^9)$ steps. A first and simple improvement is given by realizing that we only need to look at triples of nodes on one side. Given nodes v_1, v_2, v_3, we can take the intersection of their neighborhoods $\bigcap_i \delta(v_i)$. If the intersection contains at least b elements, then a $K_{3,b}$ has been found, else those three nodes cannot be part of a $K_{3,b}$. This reduces the running time to $\Theta(n^4)$.

The ideas underlying this improvement can be extended further. Obviously, any node of indegree less than 3 can be pruned, and similarly for outdegrees less than b. Once nodes have been pruned, we can iterate, as the degree of other nodes may have been reduced. In addition, if a node reaches indegree exactly 3 (or outdegree exactly b), it can be verified easily if it and all its neighbors form a $K_{3,b}$, after which they can either be reported (and pruned), or just pruned. These heuristics, while not useful in a worst-case scenario, help a lot in practice. They were reported, along with other heuristics, by Kumar et al. [254], and used to identify several 10,000 communities in the web graph.

Remark 3.10 Finding large complete bipartite graphs can be likened to finding dense areas of a 0-1 matrix, a task known as *association rule mining* in the data mining community [6]. A common approach there (see, e.g., [7]) is to take simple rules, and combine them into larger rules. The idea is that any subgraph of a larger complete (or dense) graph must itself be complete (or dense). Hence, looking only at combinations of small dense graphs rules out a lot of unnecessary attempts. By starting from $K_{1,1}$ graphs, extending them to $K_{1,2}$ and $K_{2,1}$, then to $K_{2,2}$, $K_{3,1}$, and $K_{1,3}$, etc., we make sure to only look at relevant data, which leads to a lot of speedup in practice (though again no theoretical guarantees in the case of dense graphs).

3.3.1 Other eigenvalues

If we follow the intuition of hubs and authorities a little further, we arrive at an alternate approach to identify community structure. Rather than looking merely for dense bipartite subgraphs (or large complete bipartite graphs), we could instead look at other eigenvectors of the adjacency matrix (or cocitation matrix). Nodes with large positive or negative entries in those eigenvectors (hub or authority weights) could then be considered as the cores of communities [173].

To gain some intuition for why this might make sense, consider the following adjacency (or cocitation) matrix B.

$$B = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

The largest eigenvalues of B are $\lambda_1 = 3$ and $\lambda_2 = 2$, and the corresponding eigenvectors are $\mathbf{e_1} = (0,0,1,1,1)$, and $\mathbf{e_2} = (1,1,0,0,0)$, respectively. The sub-communities found are the bottom three nodes (a $K_{3,3}$), and the top two nodes (a $K_{2,2}$).

Here is the intuitive reason why this works: consider a vector $\mathbf{e}$ which is an eigenvector for a large eigenvalue λ. Let i be some index such that e_i is "large". The i^{th} row of $B \cdot \mathbf{e}$ is $\sum_j b_{ij} e_j$, and must be equal to λe_i because of the eigenvector property. Because both λ and e_i were assumed to be "large," the sum is large, and most of the contribution to the sum must come from the entries j for which e_j is large. But then, the corresponding b_{ij} must also be large. In other words, the b_{ij} values are "fairly large" among the pairs (i,j) such that the e_i are large. But this just means that the nodes i with large e_i values are relatively densely connected in B, which is exactly what we are looking for in a community.

While the above provides some intuition, there are also quite a few things that can be said formally about this approach. One is the following lemma, which is a well-known basic fact about random walks on undirected graphs.

Lemma 3.11 *Let G be an undirected graph, and define the matrix A as $a_{ij} = \frac{1}{d(j)}$ if (i,j) is an edge of G, and $a_{ij} = 0$ otherwise. (A corresponds exactly to the transition probabilities for a random walk on G.)*

Then, the largest eigenvalue of A is 1, and its multiplicity k is the number of connected components of G. Furthermore, the k (right) eigenvectors corresponding to the eigenvalue 1 identify the connected components of G.

Remark 3.12 We are deliberately vague about "identifying the connected components." If there is more than one component, then the eigenvectors are not unique: any set of k orthonormal vectors spanning that space is possible. We can say the following, however: there is a set of k orthonormal eigenvectors such that the i^{th} eigenvector has non-zero entries exactly for the nodes in the i^{th} connected component. Furthermore, given *any* set of k eigenvectors, it is fairly easy to obtain instead a set with this "identifying" property.

Of course, there are easier ways to compute connected components, and furthermore, connected components are not necessarily the most exciting type of communities. However, Lemma 3.11 at least motivates the approach of Gibson et al. [173] described above.

3.4 Personalized PageRank and Communities

Much more can be said about eigenvectors of different matrices derived from the adjacency matrix of a graph, and the sense in which they identify communities. In particular, eigenvectors are useful in identifying sets with small "surface-to-volume" ratio, i.e., many edges inside the set, and few edges outside.

More precisely, let $e(S, \overline{S})$ denote the number[1] of edges between S and $\overline{S}$. The *volume* of a node set S is the sum of degrees of all its nodes, i.e., $vol(S) = \sum_{v \in S} d_v = 2e(S,S) + e(S, \overline{S})$. The *conductance* of S is then

$$\Phi(S) := \frac{e(S, \overline{S})}{\min(vol(S), vol(\overline{S}))}. \tag{3.5}$$

[1]In previous chapters, we used the same notation for the *set* of edges between S and $\overline{S}$.

The conductance of the graph G is then $\Phi(G) := \min_{S \subset V(G)} \Phi(S)$. A closely related quantity is the *edge expansion* $\frac{e(S,\overline{S})}{\min(|S|,|\overline{S}|)}$, which we will revisit in more detail in Section 7.2.

Since low-conductance sets can be construed as natural "communities," there has been a lot of work on finding them, and relating them with spectral properties. The best-known result relating the concepts is Cheeger's Inequality. To state it, we define the *normalized Laplacian matrix* of G to be $\mathcal{L}_G = I - D^{-1/2}AD^{-1/2}$, where I is the $n \times n$ identity matrix, $D = diag(d_1, d_2, \ldots, d_n)$ is the diagonal matrix of node degrees, and A is the adjacency matrix of G. Let $0 = \lambda_1(G) \leq \lambda_2(G) \leq \lambda_3(G) \leq \cdots \leq \lambda_n(G) \leq 2$ be the eigenvalues of $\mathcal{L}_G$. Then, Cheeger's Inequality states the following:

Theorem 3.13 (Cheeger's Inequality [14, 95]) $\Phi^2(G)/2 \leq \lambda_2 \leq 2\Phi(G)$.

Rearranged, Cheeger's Inequality bounds the conductance of a graph as $\sqrt{2\lambda_2} \geq \Phi(G) \geq \lambda_2/2$. The proof of Theorem 3.13 is constructive: it shows how find a low-conductance cut $(S, \overline{S})$ from the *Fiedler vector*, i.e., the eigenvector corresponding to λ_2. More recent work (e.g., [261, 256]) gives generalizations and sharpenings of Cheeger's Inequality incorporating higher eigenvalues $\lambda_3, \lambda_4, \ldots$.

3.4.1 Local Communities

While the algorithms implicit in Theorem 3.13 and its generalizations can find communities of provably low conductance, these communities are necessarily "global" in nature: they give a good partition of the graph into two (or a few) sets. There are two major drawbacks to this approach:

- For a graph at the scale of the WWW or the social network of all or most humans, a partitioning into a constant number of sets is likely not of much use. Instead, given a handful of individuals, we would typically be interested in finding their "tightly knit" community, comprising only a very small portion of the graph.
- For graphs such as the WWW or a large social network, even linear time in n (let alone polynomial time with larger exponent) may be too slow, in particular when the community we are seeking is itself small.

The second concern suggests the notion of *local computation* [356]: the running time should be (nearly) linear not in the size of the input, but in the size of the *output*. Applied to community detection in huge graphs, this means that if the community we seek is small, then the running time should be much smaller than the size of the graph.

A valuable technique for local computation of communities was pioneered by Spielman and Teng [356]. Let S be the node set whose community we would like to find. The idea is to start random walks in S and truncate them after a suitable (small) number of steps. The nodes that are reached in this way will be "close" and "well-connected" to S. Of course, in the limit (as the number of steps goes to infinity), the distribution over nodes reached by a random walk will be independent of the start node. But after a small number of steps, the random walk will be heavily biased towards the nodes "around" S. This is particularly true if there is a community of low conductance around S, since the random walk is unlikely to cross one of the few edges leaving S.

Explicitly truncating the random walk after some number of steps suffers from the sharp cutoff. Perhaps a more natural approach is for the random walk to return to S probabilistically in each step: for each step, with probability ϵ, the random walk returns to a (uniformly) random node in S; with probability $1 - \epsilon$, it takes a normal random walk step. Notice that this is exactly the behavior of topic-sensitive (also called *personalized*) PageRank from Section 2.4, when the restart vector $\mathbf{f}$ is uniform over S. This suggests that with a suitable choice of ϵ, the nodes with large topic-sensitive PageRank values should form a reasonable community around S.

In fact, this is not just a useful heuristic, but an approach that comes with provable guarantees, as shown by Andersen, Chung, and Lang [17, 16]. We will explore a simplified analysis of some of the key insights from [17, 16].

For the purpose of the discussion here, we assume that S consists of just a single node $\hat{v}$, so the reset vector is $\mathbf{f} = \mathbf{e}_{\hat{v}}$ (the vector with 1 in coordinate $\hat{v}$, and 0 in all other coordinates). Recall from Section 2.4 that the personalized PageRank vector $\mathbf{p}_{\hat{v}}$ satisfies

$$\mathbf{p}_{\hat{v}} = (1 - \epsilon) \cdot M \cdot \mathbf{p}_{\hat{v}} + \epsilon \cdot \mathbf{f}, \tag{3.6}$$

where M is the transition matrix of the random walk on the web graph G. For notational convenience, and because $\hat{v}$ will be mostly clear from the context, we will omit $\hat{v}$ from the notation unless it is needed for clarity.

In order to use personalized PageRank values for local community detection, two issues must be addressed:

1. The entries of $\mathbf{p}$ must be computed efficiently. When truncating the random walk deterministically after some number k of steps, it is clear that the support (set of non-zero entries) of $\mathbf{p}$ is at most $d_{\max}^k$. But $\mathbf{p}$ will in general have full support. One of the main contributions of [17] was to show how to compute a good approximation of $\mathbf{p}$ that has small support, and perform this computation in time depending only on the support size of the approximation, not on the graph's size n.

   For the results in [17, 16] on how to actually detect communities, it then becomes crucial to account for the approximation in $\mathbf{p}$, which is the source of a large share of the technical contribution. For our simplified exposition here, we will assume that we have access to the precise values of $\mathbf{p}$, though we will occasionally consider the difficulties that arise from approximations to $\mathbf{p}$.

2. We need to establish formally how the $\mathbf{p}$ relate to the community structure. This is the focus of the analysis here.

As we saw in Chapter 2, the stationary probabilities of a simple random walk are proportional to the degrees (for undirected graphs); thus, the degrees will also be an important part of $\mathbf{p}$, obscuring the community structure we are seeking. Therefore, for most of the analysis, we are interested in the normalized vector $\mathbf{q}$, defined as $q_i = p_i/d_i$. For simplicity of notation, we assume that nodes are sorted, so that $q_1 \geq q_2 \geq \cdots \geq q_n$. We write $S_j := \{1, \ldots, j\}$.

The key lemma of the analysis, intuitively capturing the connection between eigenvector entries and community structure, is the following:

Lemma 3.14 (Sharp Drop Lemma) *If there is a sharp drop in q_i values from index j to index k, then S_j has small conductance. More precisely, consider any j, and some $\beta \in (0, 1)$. If for all $k > j$, (at least) one of the following two holds:*

- $vol(S_k) < (1 + \beta) \cdot vol(S_j)$,
- $q_k < q_j - \frac{\epsilon}{\beta \cdot vol(S_j)}$,

then $e(S_j, \overline{S_j}) \leq 2\beta \cdot vol(S_j)$.

Proof. The high-level intuition of the proof has been outlined before: if a set S_j has large conductance, then a lot of probability mass will traverse the cut from S_j to its complement with each step of the random walk, so the q_i values should equalize quickly. The contrapositive states that if there is a sharp drop, then the conductance must be small.

Fix any j and $\beta \in (0, 1)$. We distinguish two cases.

1. If S_j is large, in the sense that $vol(S_j) > \frac{1}{1+\beta} \cdot vol(G)$, then almost all edges are inside S_j, so few edges can cross from S_j to $\overline{S_j}$. More precisely,

$$e(S_j, \overline{S_j}) \;\leq\; vol(G) \cdot (1 - \frac{1}{1 + \beta}) \;<\; \beta \cdot vol(S_j).$$

   So for large S_j, the conclusion of the lemma always holds.

2. In the case $vol(S_j) \le \frac{1}{1+\beta} \cdot vol(G)$, let k be the index such that $vol(S_{k-1}) \le (1+\beta) \cdot vol(S_j) \le vol(S_k)$, so going from $k-1$ to k is where the volume reaches $(1+\beta)$ times that of S_j. In other words, k is the first index such that S_k is "sufficiently bigger" than S_j. Assume that $e(S_j, \overline{S_j}) > 2\beta \cdot vol(S_j)$. We will show that there is no sharp drop at j for any $k' \le k-1$. Because the volume of those $S_{k'}$ is also close to the volume of S_j, this shows that the antecedent of the lemma is false, and thus completes the proof of the contrapositive.

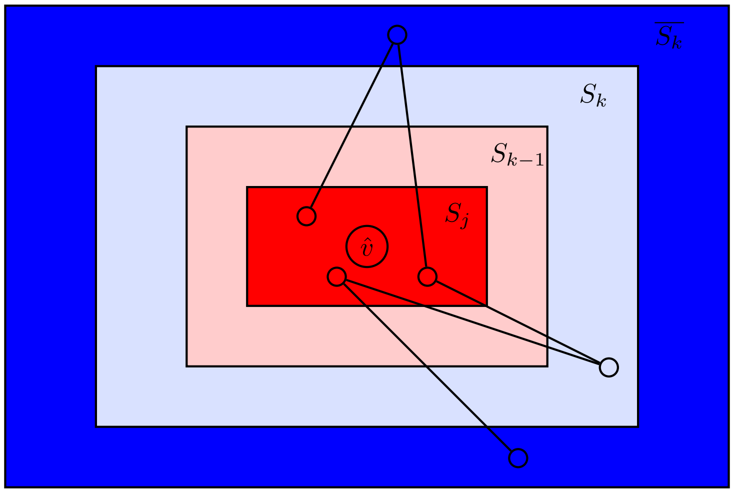


Figure 3.1: An illustration of the nested sets S_i.

First, we lower-bound the edges between S_j and $\overline{S_{k-1}}$, shown schematically in Figure 3.1 as edges between the dark red and (light or dark) blue regions. Because $e(S_j, \overline{S_j}) > 2\beta\, vol(S_j)$, we can bound

$$e(S_j, \overline{S_{k-1}}) \;\ge\; e(S_j, \overline{S_j}) - vol(S_{k-1} \setminus S_j) \;>\; 2\beta \cdot vol(S_j) - \beta\, vol(S_j) \;=\; \beta \cdot vol(S_j).$$

Next, we want to relate the q_i values with the number of crossing edges. For any set S, the probability mass inside it after one step of the random walk is the probability that was inside S before the extra step, plus the probability mass that entered S, minus the probability mass that left S. So we can write

$$\sum_{u \in S} (M \cdot \mathbf{p})_u = \sum_{u \in S} p_u + \sum_{(u,v) \in (S, \overline{S})} \frac{1}{d_v} \cdot p_v - \sum_{(u,v) \in (S, \overline{S})} \frac{1}{d_u} \cdot p_u \;=\; \sum_{u \in S} p_u - \sum_{(u,v) \in (S, \overline{S})} (q_u - q_v). \tag{3.7}$$

Rearranging Equation (3.6), we get that $M \cdot \mathbf{p} = \frac{1}{1-\epsilon} \cdot (\mathbf{p} - \epsilon \mathbf{f})$. Substituting this value, and considering only sets $S \ni \hat{v}$, we can rewrite

$$\sum_{u \in S} (M \cdot \mathbf{p})_u = \frac{1}{1-\epsilon} \sum_{u \in S} (p_u - \epsilon f_u) \;\ge\; \sum_{u \in S} (p_u - \epsilon f_u) \;=\; \sum_{u \in S} p_u - \epsilon. \tag{3.8}$$

Choosing $S = S_j$, and equating Equations (3.7) and (3.8), we obtain that $\sum_{(u,v) \in (S_j, \overline{S_j})} (q_u - q_v) \le \epsilon$. By dropping the edges from S_j to $S_{k-1} \setminus S_j$ from the sum, we can now bound

$$\sum_{(u,v) \in (S_j, \overline{S_j})} (q_u - q_v) \;\ge\; \sum_{(u,v) \in (S_j, \overline{S_{k-1}})} (q_u - q_v) \;\ge\; e(S_j, \overline{S_{k-1}}) \cdot (q_j - q_k) \;\ge\; \beta \cdot vol(S_j) \cdot (q_j - q_k).$$

The middle inequality used that q_j is the largest q value of any node in S_j, and that q_k is the smallest q value of any node outside of S_{k-1}.

Solving the inequality now gives that $q_j - q_k \leq \frac{\epsilon}{\beta \cdot vol(S_j)}$. Thus, we have shown that if the set S_j has large conductance, then from S_j to S_k, there is no steep drop, nor does S_k have roughly the same volume as S_j. This completes the proof. ■

Lemma 3.14 shows that we can read off low-conductance cuts from sharp drops in q_i values. However, for this to be valuable, we still have to show that cuts of sharp drops actually exist, or at least to examine under which conditions they do. The approach here is algorithmic — we will consider a natural candidate algorithm for finding a suitable set S_j, and understand roughly under what conditions it succeeds.

We start with an index j_0 and corresponding set S_{j_0}. Then, for each i, let j_{i+1} be smallest such that $vol(S_{j_{i+1}}) \geq (1+\beta) \cdot vol(S_{j_i})$. That way, if there is a sharp drop from j_i to j_{i+1}, the first condition holds by definition for $j_i < k < j_{i+1}$, and Lemma 3.14 immediately implies that S_{j_i} has small conductance. The following lemma shows the implications of finding no sharp drop in this sequence:

Lemma 3.15 *If the sequence $j_0, j_1, \ldots$ contains no sharp drop, then $q_{j_k} \geq q_{j_0} - \frac{\epsilon(1+\beta)}{\beta^2 \cdot vol(S_{j_0})}$.*

Proof. By definition of the sequence, $vol(S_{j_i}) \geq (1+\beta)^i \, vol(S_{j_0})$, and because no sharp drop was found in the sequence, $q_{j_{i+1}} \geq q_{j_i} - \frac{\epsilon}{\beta \cdot vol(S_{j_i})}$ for all i. Substituting both, we obtain that

$$q_{j_k} \;\geq\; q_{j_0} - \frac{\epsilon}{\beta \cdot vol(S_{j_0})} \cdot \sum_{i=0}^{k} (1+\beta)^{-i} \;=\; q_{j_0} - \frac{\epsilon}{\beta \cdot vol(S_{j_0})} \cdot \frac{1+\beta}{\beta} \;=\; q_{j_0} - \frac{\epsilon(1+\beta)}{\beta^2 \cdot vol(S_{j_0})}. \;■$$

Thus, if no low-conductance cut is found, then all q_j values are "almost as large" as q_{j_0}. Next, we show that there is a choice that ensures a large q_{j_0}.

Lemma 3.16 *For any probability vector* $\mathbf{p}$ *(in particular: the PageRank vector), there exists an index i with $q_i \geq \frac{1}{H_{2m}} \cdot \frac{1}{vol(S_i)}$. (Here, m is the number of edges, and H_j is the j^{th} Harmonic number.)*

Proof. Suppose that no such i existed. Then

$$1 \;=\; \sum_i p_i \;=\; \sum_i q_i d_i \;<\; \frac{1}{H_{2m}} \cdot \sum_i \frac{d_i}{vol(S_i)}.$$

By writing $d_i = \sum_{j=1}^{d_i} 1$, we can now write this expression as

$$\frac{1}{H_{2m}} \cdot \sum_i \sum_{j=1}^{d_i} \frac{1}{vol(S_i)} \;\leq\; \frac{1}{H_{2m}} \cdot \sum_i \sum_{j=1}^{d_i} \frac{1}{vol(S_i) - j + 1} \;=\; \frac{1}{H_{2m}} \cdot \sum_{j=1}^{2m} \frac{1}{j} \;=\; 1,$$

a contradiction. ■

We will only outline the rest of the algorithmic and analysis ideas. We would like to find a set $S \ni \hat{v}$ with $\Phi(S) \leq \beta$, and of sufficiently large volume $vol(S) \geq x$. In order for a simple heuristic to have a chance, we assume that not only does such a set S exist, but in fact, there is a set of sufficiently large volume with significantly smaller conductance. Define the restart probability to be $\epsilon = \frac{\beta^2}{C \cdot H_{2m}}$ for a suitable constant C.

Let j_0 be the largest index i satisfying $q_i \geq \frac{1}{H_{2m}} \cdot \frac{1}{vol(S_i)}$; the existence of such a j_0 is guaranteed by Lemma 3.16. Then, if no sharp drop is found, all the q_{j_i} in the sequence outlined above satisfy

$$q_{j_i} \;\geq\; q_{j_0} - \frac{1+\beta}{C \cdot H_{2m} \cdot vol(S_{j_0})} \;\geq\; \frac{1}{H_{2m}} \cdot \frac{1}{vol(S_{j_0})} - \frac{1+\beta}{C \cdot H_{2m} \cdot vol(S_{j_0})} \;\geq\; C' \cdot \frac{1}{H_{2m} \cdot vol(S_{j_0})},$$

for a suitable constant C'.

We say that the vertex v is *ϵ-central* to a set S if the vector $\mathbf{p}_v$ defined by Equation (3.6) satisfies $\sum_{u \notin S} p_{v,u} \leq \frac{2\Phi(S)}{\epsilon}$. In other words, v is ϵ-central to S if the resetting random walk for v is unlikely to leave S. The following lemma (whose proof we omit) says that central vertices make up a large fraction of sets:

Lemma 3.17 (Probability Capturing Lemma) *For any set S and any ϵ, let $S_\epsilon \subseteq S$ be the set of ϵ-central vertices of S (with respect to the random walk with reset probability ϵ). Then, $vol(S_\epsilon) \geq \frac{1}{2} vol(S)$.*

Based on Lemma 3.17, we cannot be assured that the vertex $\hat{v}$ we chose will find the good community S, but for a lot of other choices of the start vertex, we would have started with a vertex central to S. Now, the argument basically says that if the target volume x is large enough (in particular, $x \geq vol(S_{j_0})$), then the sequence of indices j_i will not include a lot of nodes outside the "ideal" community (i.e., nodes from $\overline{S}$), not contain much more than half of the volume of G, and recover something close to S of conductance at most 3β. Some of the complexity arises due to the error analysis because the personalized PageRank vectors are only computed approximately.

3.5 Modularity

We now move away from the problem of identifying just one community, and instead try to partition the graph into a collection of *disjoint* communities. In this sense, the problem of finding communities is synonymous to the problem of partitioning a graph, or of *clustering* a graph. Naturally, there are many different objectives describing the quality of such a partitioning. Here, we will focus on a measure called *modularity* proposed by Newman et al. [314, 310].

To motivate the idea behind the modularity measure, we return to an observation made earlier in the context of α-communities: high-degree nodes, by definition, tend to contribute more edges to communities. Thus, if we include many high-degree nodes in a community together, we expect to capture a lot of the edges. On the other hand, if we isolate low-degree nodes, we expect not to lose many edges. This really does not tell us much about communities, but only recovers known information about the degree distribution. The idea of the modularity measure is to capture how many more edges a partitioning explains beyond what could be predicted merely from the degree distribution. To capture what we mean by "from the degree distribution", we look at how many more edges are inside communities than would be in a random graph with the same degree distribution.

Formally, consider a graph $G = (V, E)$, and a partition $\mathcal{P} = \{S_1, \ldots, S_k\}$ of that graph. If the partition "explains" the communities in the graph, then just the partition, without any information about the edges, should allow us to "reconstruct" the edges relatively well. How well is captured by *modularity.* The actual number of edges inside S_i is exactly $|e(S_i)|$. If all we knew about G was its degree distribution, then the *expected number* of such edges can be calculated as follows: we assume that the graph is a uniformly random multi-graph of the same degree distribution, i.e., we allow self-loops and parallel edges.

Consider a vertex $v \in S_i$ of degree $d(v)$. For any vertex u, the probability that a given edge e of v has u as its other endpoint is $\frac{d(u)}{2m-1}$, where m is the total number of edges. The reason is that with m edges, there are $2m$ edge endpoints, and for any one edge, it might choose any one of these endpoints (except the ones that's already used up). By linearity of expectation, the expected number of edges between u and $v \neq u$ is therefore $\frac{d(u)d(v)}{2m-1}$. The expected number of self-loops of v is $\frac{d(v)(d(v)-1)}{2m-1}$, because one of the edge endpoints of v is already used up for the other end of the edge. Summing up over all pairs u, v in S_i now gives us that the expected number of edges with both endpoints inside S_i is $\frac{1}{2}\frac{d(S_i)(d(S_i)-1)}{2m-1}$, where we write $d(S_i) = \sum_{v \in S_i} d(v)$. The factor of $\frac{1}{2}$ here arises because each edge is considered twice, once for each of its endpoints. Ignoring the two -1 terms, and summing up over all of the partitions gives us that the expected number of edges inside partitions is (roughly) $\frac{1}{4m}\sum_i d(S_i)^2$.

We define the *modularity* $q(\mathcal{P})$ of a partition $\mathcal{P}$ as the difference between the actual number of edges inside clusters, and the expected such number under a uniformly random multigraph, normalized to a scale of $[-1, 1]$ by dividing out the total number of edges. Thus, the definition is

$$q(\mathcal{P}) \quad = \quad \frac{1}{m}\sum_i |e(S_i)| - \frac{1}{4m} d(S_i)^2. \tag{3.9}$$

Thus, we can now formally state the algorithm question as finding a partitioning $\mathcal{P}$ maximizing $q(\mathcal{P})$. Notice that the number k of clusters is not specified; avoiding the need to pre-specify the number of clusters is one of the advantages of the modularity objective. Of course, if desired, one can also consider a version in which the number of clusters is actually given.

Little is known about algorithms with provable guarantees for the problem of finding a modularity-maximizing partitioning. Only recently did Brandes et al. prove that the problem is NP-hard [66]. So far, no algorithms with provable guarantees are known. Many heuristics have been proposed, based on bottom-up greedy aggregation [310, 98], identifying edges which lie on many shortest paths [314], extremal optimization [135], simulated annealing [191], and rounding of linear and semi-definite programs [5].

One of the more interesting heuristics by Newman [311, 312] also uses the top eigenvector of a matrix termed the *modularity matrix*. The key observation is the following: if we define $a_{u,v} = 1 - \frac{d(u)d(v)}{2m}$ if there is an edge between u and v, and $a_{u,v} = -\frac{d(u)d(v)}{2m}$ otherwise, then Equation 3.9, the definition of modularity, can be rewritten as $\frac{1}{4m}\sum_i \sum_{u,v \in S_i} a_{u,v}$. Now imagine that we only want a partition into two communities, and use an indicator variable $y_v = \pm 1$ for each node v, depending on whether v is in one or the other partition. Then, the objective can be written as $\frac{1}{4m}\sum_{u,v} a_{u,v}(1 + y_u y_v)$. Because $\sum_{u,v} a_{u,v} = 0$, we can rewrite the modularity as

$$\frac{1}{4m}\mathbf{y}^\intercal A \mathbf{y}, \tag{3.10}$$

where A is the matrix of all $a_{u,v}$ entries, and $\mathbf{y}$ is a vector of ± 1 entries describing the partition.

While finding such a vector $\mathbf{y}$ is NP-hard, it ceases to be hard if we remove the restriction that all entries be ± 1. In fact, the vector maximizing $\mathbf{y}^\intercal A \mathbf{y}$ is exactly the top eigenvector of A. Thus, it appears to be a good heuristic to first find the top eigenvector of the modularity matrix A, and then round the entries of y. Different ways suggest themselves: the most obvious is to put all nodes with positive $\mathbf{y}$ entries in one partition, and all nodes with negative $\mathbf{y}$ entries in the other. But instead, one can choose any threshold τ, and put all nodes with $y_v \geq \tau$ in one partition, and $y_v \leq \tau$ in the other. In fact, it probably makes sense to try all possible τ values for this.

Akin to considering other eigenvectors in Section 3.3.1, Newman [311, 312] also suggests looking at the next eigenvectors, and the partitionings defined by them. Again, heuristically, this may lead to the discovery of interesting community structure.

Since its introduction by Newman, modularity has become a very popular measure of community structure, in particular in the physics and biology communities. However, there are also important questions to ask. For instance, at what modularity measure is an observed community structure really meaningful?

Clauset et al. [98] suggest that the community structure identified by an algorithm is significant in the context of a graph when the modularity is above about 0.3. On the other hand, Guimerà et al. [192] argue that those values will be obtained merely by random fluctuations in $G(n, p)$ random graphs. While for any *given* community partitioning, the expected number of edges inside it under a random graph will be (essentially) equal to the baseline used in the definition of modularity, this does not hold if the community structure is chosen after the random graph is generated. A more in-depth examination of the idea of "conditioning on the degree sequence" is performed by Gaertler et al. [166]. They generalize the notion to express the "significance" of a clustering as the amount of information (edges) it contains compared to any posited baseline model. A uniform random multi-graph subject to a degree distribution is one particular example, but far from the only one. Similarly, the objective functions falling into the broad class of "modularity like" are explored further by Reichardt and Bornholdt as well [336].

3.6 Correlation Clustering

In the previous section, we began considering the problem of community identification as partitioning a graph. The assumption there was that each edge expresses that two nodes are "more likely" to be in the same community. In many cases, this is a reasonable assumption. However, in particular in contexts such as the web (including blogs), we may often find edges that explicitly suggest the two endpoints might *not* be in the same community. For instance, in politics or sports, pages will link to other pages with the explicit goal of deriding the content. This can be frequently identified from anchor text and similar clues. Thus, it makes sense to consider a clustering problem where edges are annotated with ‘+’ or ‘-’, expressing whether the endpoints appear to be similar or dissimilar.

This is a clustering problem called *correlation clustering* [28]. In the setting described above — each edge of the graph is annotated with a label of ‘+’ or ‘-’ — the goal is to find a clustering that puts many ’+’ edges inside clusters, and ’-’ edges between clusters. However, these goals may be conflicting, as can be seen for a triangle with two edges labeled ’+’ and one labeled ’-’. As with the problem of maximizing modularity, we do not pre-specify the number of clusters, though of course one can also consider the variant where the target number of clusters is given.

More formally, given the graph $G = (V, E)$ on n vertices, we write $+(i, j)$ if the edge between i and j is labeled ‘+’ and similarly for $-(i, j)$. The optimization problem can now be expressed in two ways:

- Maximize the number of *agreements*, i.e., the number of ‘+’ edges inside clusters plus the number of ‘-’ edges across clusters. This problem is called MaxAgree.
- Minimize the number of *disagreements*, i.e., the number of ‘+’ edges across clusters plus the number of ‘-’ edges inside clusters. This problem is called MinDisAgree.

Clearly, the solution optimizing the first criterion is the same as the one optimizing the second. However, we will see that the two objective functions differ with respect with how well they can be approximated.

The correlation clustering problem is NP-hard, even for the complete graph (where each possible edge exists, and is labeled either ‘+’ or ‘-’). Hence, we are interested here in approximation algorithms.

3.6.1 A Simple Algorithm for Maximizing Agreements

For the maximization version, we may choose to go just after one of the two types of edges. By putting all nodes in one cluster, we get all the ‘+’ edges right — by putting each node in its own cluster, we get all the ‘-’ edges right. This suggests the following algorithm [28]:

> If the number of ‘+’ edges in the graph is larger than the number of ‘-’ edges, then put everything into one cluster, else put every node in its own cluster.

Claim 3.18 *This is a $\frac{1}{2}$-approximation.*

Proof. If the graph has m edges, then the optimal solution can have at most m agreements. Our algorithm produces a clustering that has at least $\frac{m}{2}$ agreements. Hence, it is a $\frac{1}{2}$-approximation. ∎

One consequence is that in a complete graph with assignments of edges, there must exist a clustering with at least $\frac{n(n-1)}{4}$ agreements (half of the total number of edges in the complete graph).

3.6.2 Improving the Approximation Factor

While the algorithm is a $\frac{1}{2}$-approximation, it is hardly satisfactory in a practical sense. We don’t need a new clustering model or algorithm if all it does is lump everything together, or put each node in its own cluster. So naturally, we want to know if the approximation guarantee can be improved.

In [28], the authors develop a PTAS (*Polynomial Time Approximation Scheme*) for the MaxAgree problem in a complete graph. That is, they present a class of algorithms, parametrized with some $\epsilon > 0$, such that the algorithm with parameter ϵ is a $(1 - \epsilon)$ approximation with running time $O(n^2 \mathrm{e}^{O(1/\epsilon)})$. While this grows exponentially in ϵ, for any fixed ϵ, the algorithm takes polynomial time.

3.6.3 A 4-Approximation for MinDisAgree in complete graphs

Given that MaxAgree is (at least theoretically) settled for complete graphs, we next look at the minimization version. [85] gives a 4-approximation based on rounding the solution to a Linear Program.

The idea of the Linear Program is to start with an Integer Program. For each pair i, j of nodes, we have a variable x_{ij} which is 1 if they are in different clusters, and 0 otherwise. To make these variables consistent, we have to require that if i and j are in the same cluster, and j and k are in the same cluster, so are i and k. This can be expressed by saying that $x_{ik} \leq x_{ij} + x_{jk}$. Subject to this consistency requirement, we want to minimize the number of '+' edges between clusters, plus the number of '-' edges within clusters. Thus, we obtain the following IP.

$$\begin{array}{lll} \text{Minimize} & \sum_{+(ij)} x_{ij} + \sum_{-(ij)} (1 - x_{ij}) & \\ \text{subject to} & x_{ik} \leq x_{ij} + x_{jk} & \text{for all } i, j, k \\ & x_{ij} \in \{0, 1\} & \text{for all } i, j \end{array} \tag{3.11}$$

As we know, solving IPs is itself NP-hard, so as usual, we want to relax the IP to a linear program, by replacing the last constraint with the constraint that $0 \leq x_{ij} \leq 1$ for all i, j. This LP can now be solved; however, the output will be fractional values x_{ij}. Our goal is to somehow convert these fractional values into integer ones ("round" the solution) in such a way that the objective function value of the rounded solution is not much more than that of the fractional solution. As the latter is a lower bound on the optimum integer solution, we will derive an approximation guarantee.

After obtaining the fractional x_{ij} values (in polynomial) time, we need some intuition for dealing with them. We notice that the main constraint is just the triangle inequality. Hence, if we define $x_{ii} := 0$ for all i, the x_{ij} exactly form a metric space. In other words, the optimum solution is the metric minimizing the objective function. Nodes that are close in the metric space can be considered as being "almost in the same cluster", while distant nodes are "quite separated". So we likely want our clusters to consist of nodes that are mutually close. At the same time, we need to choose the boundaries carefully, so as not to cut too many edges. It turns out that the following Algorithm 2 from [85] carefully trades off between the different types of costs:

Algorithm 2 LP-Rounding

```
1: Start with a set $S$ containing all the nodes.
2: while $S \neq \emptyset$ do
3:   Select an arbitrary node $u \in S$.
4:   Let $T := \{i \mid x_{ui} \leq \frac{1}{2}\} \setminus \{u\}$
5:   if the average distance from $u$ to the vertices in $T$ is at least $\frac{1}{4}$ then
6:     Let $C := \{u\}$ (a singleton cluster).
7:   else
8:     Let $C := T \cup \{u\}$.
9:   Remove all of $C$ from $S$.
```

Theorem 3.19 *Algorithm 2 is a 4-approximation.*

Proof. The idea of the proof is to compare the mistakes that are incurred with the above algorithm against the LP cost, by showing that whenever a cluster is formed, the number of mistakes ('+' edges across clusters plus '-' edges inside clusters) is at most four times the corresponding LP cost of the corresponding edges.

First, by simple applications of the triangle inequality and reverse triangle inequality, we obtain the following relationships between the distances (which will be used later).

Lemma 3.20 *For any nodes i, j, u:*

- *The cost of a '+' edge (i, j) incurred by the LP is at least $x_{ij} \geq x_{uj} - x_{ui}$.*
- *The cost of a '-' edge (i, j) incurred by the LP is at least* $\max(0, 1 - x_{uj} - x_{ui})$.

We will show the claim for each cluster separately. So let u be a cluster center of the cluster C. We can then sort the other nodes by increasing distance from u, i.e., whenever $x_{ui} < x_{uj}$, we say that $i < j$ (ties are thus broken arbitrarily).

We distinguish between the two cases when a singleton cluster is created, or when the cluster includes T.

1. Singleton cluster $\{u\}$:

   This case occurs when the average distance from u to the vertices in T is at least $\frac{1}{4}$. Our algorithm then pays for all the '+' edges incident with u. For each such '+' edge whose other endpoint is not in T, the LP pays at least $\frac{1}{2}$. And for the edges whose other endpoint is in T, the LP pays at least

   $$\textstyle\sum_{i\in T,+(ij)} x_{ui} + \sum_{i\in T,-(ij)}(1-x_{ui}) \quad\geq\quad \sum_{i\in T} x_{ui} \quad\geq\quad \frac{|T|}{4}.$$

   Therefore, for a singleton cluster, our algorithm pays at most four times the LP cost.

2. Non-singleton clusters

   In this case, the algorithm could be making two kinds of mistakes:

   - Allowing '-' edges inside C.
   - Cutting '+' edges between C and $S \setminus C$.

   We first analyze the costs of negative edge mistakes. If we have a '-' edge (i,j) such that both i and j are close to u, then they must be close to each other, so the fractional solution must also pay a lot for this edge. Specifically, if x_{uj} and x_{ui} are both at most $\frac{3}{8}$, then from the second part of Lemma 3.20, x_{ij} is at least $1 - x_{uj} - x_{ui} \geq \frac{1}{4}$. So the cost for any such edge incurred by our algorithm is at most 4 times that of the LP solution.

   For the remaining '-' edges, we will not be able to come up with a bound on an edge-by-edge basis. Indeed, if two nodes i and j are at distance $\frac{1}{2}$ each from u, they may be at distance 1 from each other, so the fractional solution incurs no cost. Instead, we will compare the costs node by node: specifically, we will show that for any node j, the number of '-' edges between it and nodes i that are closer to u than itself is at most four times the corresponding cost of the LP solution.

   So we fix a node j with $x_{uj} \in (\frac{3}{8}, \frac{1}{2}]$. The total cost of edges (i,j) with $i < j$ (both '+' and '-' edges) that the LP solution incurs is

   $$\sum_{i<j,+(ij)} x_{ij} + \sum_{i<j,-(ij)} (1 - x_{ij}) \quad\geq\quad \sum_{i<j,+(ij)} (x_{uj} - x_{ui}) + \sum_{i<j,-(ij)} (1 - x_{uj} - x_{ui}).$$

   By writing p_j for the number of '+' edges (i,j) with $i < j$, and n_j for the number of '-' edges (i,j) with $i < j$, we can rewrite this as

   $$p_j x_{uj} + n_j(1 - x_{uj}) - \sum_{i<j} x_{ui}.$$

   Because the algorithm chose not to have a singleton cluster, the average distance of all nodes to u is at most $\frac{1}{4}$. The last sum $\sum_{i<j} x_{ui}$ only leaves out some subset of nodes furthest away from u, so the average distance of those nodes is also at most $\frac{1}{4}$. Hence, the last sum is at most $\frac{1}{4}(p_j + n_j)$, and the entire LP cost is at least

   $$p_j x_{uj} + n_j(1 - x_{uj}) - \frac{p_j + n_j}{4} \quad=\quad p_j(x_{uj} - \frac{1}{4}) + n_j(1 - x_{uj} - \frac{1}{4}).$$

The number of negative edge mistakes the algorithm will make is at most n_j. On the other hand, because $\frac{3}{8} \le x_{uj} \le \frac{1}{2}$, the LP cost is at least $\frac{p_j}{8} + \frac{n_j}{4}$. Thus, the sum of edge costs incurred by the algorithm is at most four times that of the LP. Since this holds for any cluster, and we only account for the LP cost of any edge once, we have shown that the algorithm is a 4-approximation for '-' edges.

Next, we turn our attention to '+' edges. A '+' edge (i, j) only contributes to the cost of our solution if one endpoint, say, i, is included in the cluster that is formed, while the other, j, is not. So we are dealing with the case that $x_{ui} \le \frac{1}{2}$ and $x_{uj} \ge \frac{1}{2}$. If in fact, $x_{uj} \ge \frac{3}{4}$, then $x_{ij} \ge \frac{1}{4}$ by the triangle inequality, so the cost paid by our algorithm is within a factor of 4 of the LP-cost for any such edge. Hence, we now focus on the case of nodes j with $x_{uj} \in (\frac{1}{2}, \frac{3}{4})$. We compare the number of '+' edges cut by our algorithm to the total LP cost of all edges ('+' and '-') between j and nodes in the cluster C.

By the triangle inequality (captured in Lemma 3.20), writing p_j and n_j for the number of '+' resp. '-' edges between j and nodes from C, we obtain that

$$\begin{aligned}
\mathrm{LP}_j &= \sum_{i \in C, +(ij)} x_{ij} + \sum_{i \in C, -(ij)} (1 - x_{ij}) \\
&\ge \sum_{i \in C, +(ij)} (x_{uj} - x_{ui}) + \sum_{i \in C, -(ij)} (1 - x_{ui} - x_{uj}) \\
&= p_j x_{uj} + n_j (1 - x_{ui}) - \sum_{i \in C} x_{ui}.
\end{aligned}$$

Because the algorithm didn't form a singleton cluster, the average distance of nodes in C from u is at most $\frac{1}{4}$, so LP_j is bounded below by $p_j x_{uj} + n_j(1 - x_{uj}) - \frac{p_j + n_j}{4}$. But $\frac{3}{4} \ge x_{uj} \ge \frac{1}{2}$, so

$$\mathrm{LP}_j \;\ge\; \frac{p_j}{2} + \frac{n_j}{4} - \frac{p_j + n_j}{4} \;=\; \frac{p_j}{4}.$$

As the algorithm cuts p_j '+' edges, the total cost of edges cut by the algorithm is at most four times the LP-cost. By summing this over all nodes j and all clusters formed, we obtain that the algorithm is a 4-approximation. ■

We mentioned above that, while the optimal solution for the minimization and maximization version is the same, the approximation guarantees differ. For the minimization version on complete graphs, the algorithm from [85] we just analyzed gives a 4-approximation. On the other hand, [85] also shows that the problem is APX-hard, i.e., there is some constant such that the minimization version on complete graphs cannot be approximated to within better than that constant unless P=NP. On arbitrary graphs, the best known approximation is $O(\log n)$; however, it is open whether the problem can be approximated to within a constant.

Even though there is a constant-factor approximation for the minimization version on complete graphs, together with an APX-hardness result, we may wonder what is the best possible constant. We will show that the LP used above has an integrality gap of 2, so no rounding approach solely based on that LP can yield an algorithm with a better guarantee. The example is the "wheel" graph, in which all nodes of a k-cycle are connected to one additional center node with a '+' edge (while all edges of the cycle are labeled '-'). Then, the integral optimal solution puts all of the cycle nodes in different clusters, paying a total of $k - 1$, while the fractional optimum assigns $x_{ij} = \frac{1}{2}$ to all edges between the center and the cycle nodes, paying $\frac{k}{2}$. The ratio approaches 2 as $k \to \infty$.

The authors of [85] show that no rounding algorithm based on the same type of "region growing" will lead to an approximation guarantee of better than 3, and conjecture that in fact, no similar approach will give a better approximation than the factor of 4 obtained.

For the maximization version, there is a PTAS on complete graphs [28]. For graphs that are not complete, the problem is APX-hard. However, in this case, it is known how to approximate it to within a constant factor. The factor of 0.7664 from [28] was improved to an 0.7666 approximation via semi-definite programming by Swamy [362].

3.7 Further Reading

Since the topic of community discovery in graphs has attracted so much attention (in particular in physics and biology), there is a very large literature by now. Several good surveys exists, including ones by Fortunato and Castellano [157, 159], Danon et al. [112], and Porter et al. [329]. They share a focus on physics-style heuristics, and are short on surveying important techniques (such as approximation algorithms or rounding of linear programs) from the computer science literature. Good overviews of some of the basic techniques common in computer science are given in Part II of the book "Network Analysis" by Brandes and Erlebach [67] and a survey by Schaeffer [346].

Naturally, the identification of cohesive groups of individuals has long been an important task for sociologists as well. Some of the traditional approaches can be found in classical texts on social networks by Wasserman and Faust [378] and Scott [350]. An overview of some of the more recent work on group cohesiveness is given in the article by White and Harary [383].

The definition of Flake et al. [156] is called a strong community because of the stringent condition on each node. One can instead consider *weak communities.* In a *weak α-community S*, we only require that the *total* number of edges inside S be at least an α fraction of the sum of degrees of all nodes in S. This definition was proposed by Radicchi et al. [330], simultaneously with a paper by Hayrapetyan et al. considering the same motivation and definition [198]. As with the definition by Flake et al., the entire graph is always a weak 1-community. So again, one could like to find the *smallest* weak α-communities. Currently, no approximation algorithm with provable guarantees is known for this problem. On the hardness side, only NP-hardness is known. Another related definition, due to He et al. [199], is called (α, β)-community: a vertex set S is called an (α, β)-community if each vertex $v \in S$ connects to at least β vertices inside S, while each vertex $u \notin S$ connects to at most $\alpha < \beta$ vertices in S.

Lemma 3.9 is an example of the topics studied in the area of *extremal graph theory* [54]. Broadly, extremal graph theory studies the relationships between different graph properties. For instance, in the case of the lemma, we see that a certain number of edges implies that the graph must have bipartite cores.

Beyond Cheeger's Inequality and its generalizations, a lot more can be said about the connections between eigenvalues, eigenvectors, and graph properties. For a more detailed introduction to spectral graph theory, see for instance [95, 48, 289].

Another interesting justification for using Personalized PageRank vectors in community detection was recently provided by Kloumann et al. [248]. They consider the Stochastic Blockmodel (SBM) [206], in which nodes are divided into k disjoint partitions, and for each pair of partitions i, j, a probability $\theta_{i,j}$ is specified. For each pair u, v of nodes such that u is in partition i and v is in partition j, there is an edge independently with probability $\theta_{i,j}$. Kloumann et al. [248] show that if edges within partitions are sufficiently more likely than across partitions, and the $\theta_{i,j}$ are generally large enough that there are enough edges, then the Personalized PageRank entries for a careful choice of ϵ allow one to reconstruct the entire partition containing a given seed set S (assuming that S is entirely contained inside one partition).

More generally, the goal of recovering the blocks of a Stochastic Block Model (or partitions of a Planted Partition model) has received a lot of attention. In early pioneering work, McSherry [279] showed how to use suitable singular vectors of matrices to provably reconstruct the block structure when it is sufficiently pronounced and the graphs are dense enough. A lot of recent work (e.g., [1, 120, 277, 296, 297, 298]) has focused on the goal of recovering the partitions even when the graphs are sparse, and as the differences in $\theta_{i,j}$ values approach a lower bound beyond which reconstruction is provably impossible.

As discussed in Section 3.5, maximizing modularity has become very popular, in particular among physicists. The paper [311] contains a fairly detailed survey of much of the work that has been done on the topic. Among the more critical papers is one by Fortunato and Barthélemy [158], which proves that modularity has an intrinsic *resolution limit*, in the sense that the modularity of a proposed clustering (in a large graph) can always be improved by merging smaller communities into larger ones. In fact, Fortunato and Barthélemy show that this is the case for every clustering objective that summarizes the quality of a partitioning in a single number.

There is a direct analogy between the modularity maximization problem and correlation clustering, as observed in [5]. Consider again Equation (3.10). There is really no reason why A would have to be restricted to be the modularity matrix. For instance, if instead, we made A the matrix with +1 for '+' edges, and -1 for '-' edges, then we measure the difference between correctly and incorrectly classified edges. By adding $|E|$ and dividing by two, we thus get the number of correctly classified edges, i.e., the MaxAgree objective. Since $|E|$ is a constant, this does not change optimality of solutions. Indeed, the formulation is the basis of the approximation algorithm by Swamy [362], who adds the constraint that $y_i^2 = 1$ for all i, and then uses semi-definite programming with an appropriate rounding scheme.

One can also apply the same reasoning to the LP (3.11). Instead of having terms x_{ij} and $1 - x_{ij}$ in the objective function for edges labeled '+' resp. '-', we can write $\sum_{ij} a_{ij} x_{ij}$ for the modularity matrix entries a_{ij}, under the same constraints. While the objective function now takes a different form, we can still aim to apply the same kind of rounding techniques. Unfortunately, they do not come with approximation guarantees any more.

The previous discussion suggests a more general question: given a matrix A, find a ± 1 vector $\mathbf{y}$ maximizing $\mathbf{y}^\intercal A \mathbf{y}$. This would subsume both modularity maximization and correlation clustering. Some recent work has identified conditions on A under which provable approximation guarantees can be obtained. Nesterov [304] shows that if A is positive semi-definite, then solving the semi-definite program alluded to above and rounding appropriately gives a $2/\pi$ approximation. Charikar and Wirth [86] give an $\Omega(1/\log n)$ approximation algorithm based on semi-definite programming and rounding, under the assumption that the diagonal elements of A are all zero. While this is the case for correlation clustering, it does not apply to modularity maximization.

Chapter 4

Link-Based Classification

When looking at correlation clustering, we started from the motivation that nodes that have '+' edges between them are more likely to belong to the same cluster. In a sense, we could then consider the clusters we formed as communities, sharing perhaps some similarity in topic. If we have an a priori estimate of the topics that nodes (web pages) are about, we can use a similar approach to correct and refine our estimates. For instance, if a page about coffee points to a lot of pages about programming, then perhaps we misinterpreted the meaning of "Java" in one case, and should revise our initial estimate. Hence, we will try to optimize two conflicting goals: agreements with an a priori estimate, and agreements between nodes with edges between them.

Essentially the same problem arises frequently in the context of computer vision [64]. The goal there is to label pixels into classes (classification problem) and assign labels representing these classes as foreground or background objects (or different colors). Again, the competing constraints are that we don't want many disagreements with the a priori labels, but also not between adjacent (physically close) pixels.

4.1 Markov Random Fields and the Optimization Problem

To formalize this intuition, we can think about the following model, first proposed in the context of link-based text classification by Chakrabarti, Dom, and Indyk [80], and later also studied by Broder, Krauthgamer, and Mitzenmacher [69]. Our goal is to assign some label $f(v)$ to each node v in the graph. Focus on one node v, and consider its neighborhood $\delta(v)$. (For now, we will ignore the direction of edges.) Suppose that we knew the (correct) labels $f(u)$ of each neighbor $u \in \delta(v)$. With this knowledge, a classifier could give us a conditional probability distribution over the possible labels of v, e.g., by taking text and other content at node v into account. This is exactly the definition of a *Markov Random Field (MRF)* (see, e.g., [41, 324]). Notice that the specification of the distribution at each node v is still exponential in the size of $\delta(v)$, but if nodes have low degree, this allows us to specify probability distributions concisely.

Of course, we do not know the correct labels in neighborhoods of v, either, so what we are really looking for is a *complete labeling* maximizing the overall probability, the product over all nodes v of their conditional probability. As expected, this is still a hard problem; among others, it contains Graph Coloring or Independent Set as special cases.

We can transform this optimization problem into a more familiar looking one, as described by Kleinberg and Tardos [245]. First, by the Hammersley-Clifford Theorem [41], the probability can be written as

$$\frac{1}{Z}\exp\left(-\sum_{C\in\mathcal{C}}\Gamma_C(f|_C)\right).$$

Here, Z is a normalizing constant, $\mathcal{C}$ is the collection of all cliques in the graph, and Γ_C is a *clique potential.* The important thing here is that the clique potentials depend only on the *restriction* $f|_C$ of the labeling to C, i.e., only on the labels of the nodes in the clique.

The real observation will not necessarily be drawn from the given Markov Random Field. Instead, we assume that it was obtained by first drawing a labeling from the Markov Random Field, and then introducing independent random noise at each node, according to known distributions.

If we further assume that only pairwise interactions matter, then instead of all cliques $\mathcal{C}$, we only need to look at penalty terms Γ_e for edges. In principle, these terms could depend in arbitrary ways on the labels of the edge endpoints. However, it is natural to assume *homogeneity*: each Γ_e is of the form $w_e \cdot \Gamma$. That is, the type of penalty is the same for all edges; only the weights are different. Combining all of these assumptions, and using Bayes' Rules, we obtain the following form for the probability that the true labeling is f, given that the observed labeling is f'.

$$\frac{1}{\gamma Z}\prod_v \text{Prob}[f'(v) \mid f(v)] \cdot \exp\left(-\sum_{e=(u,v)\in E} w_e \Gamma_e(f(u), f(v))\right).$$

Here, γ is a constant depending only on f', but not on f. The terms in the first product are the probabilities of observing labels $f'(v)$ if the true underlying labels are $f(v)$, and are thus given by the description of the random noise.

By taking logarithms, we see that maximizing the probability is the same as minimizing the objective

$$\sum_v \log \frac{1}{\text{Prob}[f'(v) \mid f(v)]} + \sum_{e=(u,v)\in E} w_e \Gamma_e(f(u), f(v)).$$

This motivates the following, somewhat more general, optimization problem: Given a graph G, where each edge $e = (u, v)$ has weight w_e representing the cost or strength of the relationship between vertices u and v, we want to assign labels $f(v) \in L$ for all vertices v such that we minimize the assignment cost

$$\sum_v c(v, f(v)) + \sum_{e=(u,v)} w_e \cdot d(f(u), f(v)). \tag{4.1}$$

$c(v, f(v))$ is the cost of choosing label $f(v)$ for node v, which will be a result of deviating from the observed or otherwise determined label. d represents how "different" the two labels are, and how much penalty therefore should be assigned to adjacent nodes with these labels. Since d is a measure of dissimilarity, it makes sense to assume that it is a *metric*. From now on, we will therefore focus on the *metric Markov Random Field labeling problem*.

In fact, for the remainder of this chapter, we will focus on the special case of the *uniform labeling problem*. In that case, for two labels a, a', we have that $d(a, a') = 0$ if $a = a'$, and $d(a, a') = 1$ otherwise. Thus, labels are either identical, or simply different — there are no gradations of "different". We will discuss the general problem briefly at the end of the chapter.

Remark 4.1 In the physics literature, this type of Markov Random Field is also called the *Ising Model*. It is used to describe the distributions of states of spin systems such as magnets. In those systems, adjacent atoms tend to have the same orientation of spin in ferro-magnetic states. Depending on the temperature, the state will be more or less aligned, and physicists try to answer how likely different spin configurations will be at different temperatures.

4.2 A 2-approximation using LP-rounding

In this section, we will present and analyze a polynomial-time 2-approximation algorithm using LP-rounding. The algorithm and analysis were given by Kleinberg and Tardos [245]. Recall that we focus on uniform Markov Random Fields. That is, our goal is to find a labeling $f(v)$ of vertices v minimizing

$$\sum_v c(v, f(v)) + \sum_{e=(u,v), f(u)\neq f(v)} w_e. \tag{4.2}$$

To formulate this problem as an (integer) linear program, we define variables $x_{v,a}$, which will be equal to 1 if node v is assigned label a, and equal to 0 otherwise.

Using the $x_{v,a}$ values, we would like to define a variable y_e that is equal to 1 if the two endpoints of e have different labels, and equal to 0 otherwise. To do this, we notice that if the endpoints u and v of e have the same label, then $x_{v,a} - x_{u,a} = 0$ for all a. Otherwise, it is 0 for all but two of the a, and equal to 1 and -1 for the other two. Thus, for $e = (u, v)$,

$$y_e = \frac{1}{2} \sum_a |x_{v,a} - x_{u,a}|.$$

However, this is still no linear constraint, as $|x_{v,a} - x_{u,a}|$ is not a linear function. We will see in a moment how to deal with this problem. For now, notice that our goal is to minimize

$$\sum_{v,a} x_{v,a} \cdot c(v, a) + \sum_e w_e \cdot y_e. \tag{4.3}$$

The only constraint is that each node should obtain exactly one label, so $\sum_a x_{v,a} = 1$ for all v, and each $x_{v,a} \in \{0, 1\}$.

To return to the issue of how to express y_e via linear constraints, we first write $y_e = \frac{1}{2} \sum_a y_{e,a}$, where $y_{e,a} = |x_{v,a} - x_{u,a}|$. While we cannot directly express this as a linear constraint, we can require that $y_{e,a} \geq |x_{v,a} - x_{u,a}|$, by writing $y_{e,a} \geq x_{u,a} - x_{v,a}$ and $y_{e,a} \geq x_{v,a} - x_{u,a}$ for all $e = (u, v)$. But since the objective is to *minimize* the objective function (4.3), the optimum solution will never choose $y_{e,a}$ larger than necessary, so we will indeed have $y_{e,a} = |x_{v,a} - x_{u,a}|$. In summary, we have derived the following IP for the metric labeling problem:

$$\begin{array}{lll}
\text{Minimize} & \sum_{v,a} x_{v,a} c(v, a) + \sum_e w_e \cdot y_e & \\
\text{subject to} & \sum_a x_{v,a} = 1 & \text{for all } v \\
& y_e = \frac{1}{2} \sum_a y_{e,a} & \text{for all } e \\
& y_{e,a} \geq x_{v,a} - x_{u,a} & \text{for all } e = (u, v), a \\
& y_{e,a} \geq x_{u,a} - x_{v,a} & \text{for all } e = (u, v), a \\
& x_{v,a} \in \{0, 1\} & \text{for all } v, a.
\end{array}$$

As usual with LP rounding algorithms, we relax the constraint $x_{v,a} \in \{0, 1\}$ to $x_{v,a} \in [0, 1]$, i.e., we allow the solutions to take on fractional values. Then, we can solve the LP in polynomial time. We will investigate how to round the solution to obtain a labeling which does not perform much worse than the fractional optimum. In particular, that will show that it is not much worse than the integral optimum as well.

For the underlying idea, notice that the $x_{v,a}$, summed over all a, add up to 1 for each v. Hence, we can view them as fractions to which v is given label a, or as probabilities of v having label a. A first approach would label each node v with a with probability $x_{v,a}$. Notice that we would do very well on the labeling cost $\sum_{v,a} x_{v,a} \cdot c(v, a)$ this way, as

$$\mathbb{E}\,[\text{labeling cost of node } v] \quad = \quad \sum_a \text{Prob}[v \leftarrow a] \cdot c(v, a) \quad = \quad \sum_a x_{v,a} \cdot c(v, a),$$

Here (and later), we denote by $v \leftarrow a$ the fact that node v is assigned label a, and use the fact that $\text{Prob}[v \leftarrow a] = x_{v,a}$.

However, this does not ensure that we do well on separation costs, too. In fact, if we make the choices for different nodes v *independently*, it can happen that this rounding procedure does very badly. Suppose that the graph consists of just two vertices, v_1 and v_2, with a non-zero cost edge between them. There are two labels a and b, and all vertex labeling costs are $c \equiv 0$. Then, the fractional solution can choose any fractional assignment, so long as v_1 and v_2 have the same fractional share of a. One of the optimal solutions of the LP, which we may have to round, would thus be assigning all probabilities equal to $\frac{1}{2}$. The LP cost is then 0. But if v_1 and v_2 receive different labels, we pay a non-zero separation cost, and thus have infinitely bad approximation.

Thus, we need to make more coordinated choices between labels of nodes and their neighbors. Suppose that we label a node v with the label a. A first stab would be to label all neighbors u of a with $x_{u,a} \geq x_{v,a}$ with a as well. But then, u's label may conflict with that of one of its neighbors when it is labeled later. So we may also want to require that a be the label with the largest fractional value among u's labels. But that does not really solve the problem either. It seems that we may also have to label the neighbors of u, and so on recursively.

As a result, one thing we can do is that once we pick a node v and assign a to it, we pick $x_{v,a}$ as a threshold, and assign a to *all* nodes u having $x_{u,a} \geq x_{v,a}$. Then, when a conflict occurs between some such u and another node w which gets a different label b, at least we know that they didn't have exactly identical fractional values.

This motivates the following rounding algorithm, which picks a label a uniformly at random, as well as a random threshold, and labels all nodes with that particular label if their probability for the chosen label is above the threshold. This process is repeated until there are no more nodes left to be labeled.

Algorithm 3 LP Rounding algorithm

```
1: repeat
2:     Pick a uniformly random label $a \in L$, and a uniformly random $\alpha \in [0, 1]$.
3:     Label all $v$ with $x_{v,a} \geq \alpha$ with $a$, and remove them.
4: until no more nodes are left
```

4.2.1 Approximation Guarantee

Theorem 4.2 *Algorithm 3 is a 2-approximation.*

Proof. We prove the theorem by analyzing the assignment costs and separation costs separately. We show that for both cases, the cost incurred using the approximation algorithm is within a factor of two of the optimum fractional solution of the LP, and hence also within the same factor of the integral solution (which can be no better than the best fractional one). Let $k = |L|$ denote the number of labels.

1. To analyze the assignment cost, we first notice that in any particular iteration, v is labeled a with probability $\frac{x_{v,a}}{k}$. That is because label a is picked with probability $1/k$, and conditioned on picking label a we label v iff the random threshold is at most $x_{v,a}$.

   Because in each iteration, label a is assigned to v with probability proportional to $x_{v,a}$, the overall probability of assigning a to v is $x_{v,a}$. We can thus use the same argument as above to obtain that the expected labeling cost is $\sum_a \text{Prob}[v \leftarrow a] \cdot c(v, a) = \sum_a x_{v,a} \cdot c(v, a)$.

2. To analyze the separation cost, we first notice that we only incur cost $w_{u,v}$ for the edge $e = (u, v)$, when u and v get different labels. But that can only happen when u, v are labeled in different iterations of the algorithm. Therefore, there must have been an iteration labeling exactly one of $\{u, v\}$. We bound the probability of that happening.

   Given a label a, exactly one of $\{u, v\}$ is labeled a iff the threshold $\alpha \in (x_{u,a}, x_{v,a}]$ (assuming $x_{u,a} < x_{v,a}$). So the probability of labeling exactly one of u and v with label a is $|x_{v,a} - x_{u,a}|$. Summing over all labels a, the probability of labeling exactly one of $\{u, v\}$ with any label is $\sum_a \frac{|x_{u,a} - x_{v,a}|}{k}$.

   For any time t, we let F_t be the event that *at least* one of $\{u, v\}$ is labeled in iteration t and E_t the event that *exactly one* is labeled. We are thus interested in $\text{Prob}[E_t \mid F_t]$. First off, notice that $E_t \subseteq F_t$, so $\text{Prob}[E_t] = \text{Prob}[E_t \cap F_t] = \text{Prob}[E_t \mid F_t] \cdot \text{Prob}[F_t]$. Rearranging yields that $\text{Prob}[E_t \mid F_t] = \frac{\text{Prob}[E_t]}{\text{Prob}[F_t]}$. But we already calculated $\text{Prob}[E_t]$ above. And the probability of F_t, the event that at least one of u, v is assigned a label, is at least $1/k$, the probability that u is assigned a label. Hence, we have that

$$\text{Prob}[E_t \mid F_t] \quad \leq \quad \frac{\sum_a \frac{1}{k}|x_{u,a} - x_{v,a}|}{\frac{1}{k}} \quad = \quad \sum_a |x_{u,a} - x_{v,a}| \quad = \quad 2y_{u,v}.$$

Applying this at the time t where the first of u, v is actually assigned a label gives us that they are separated with probability at most $2y_{u,v}$. (Notice that t is a random variable, but this does not really hurt us here.)

Hence, the expected total separation cost is

$$\mathbb{E}\,[\text{separation cost}] \quad = \quad \textstyle\sum_{e=(u,v)} \text{Prob}[e \text{ separated}] \cdot w_e \quad \leq \quad \sum_{e=(u,v)} 2y_{u,v}w_e.$$

Summing up over the two cases, the total cost is at most

$$\textstyle\sum_{v,a} x_{v,a}c(v,a) + 2\sum_{e=(u,v)} w_e y_{u,v} \quad \leq \quad 2 \cdot \text{cost(LP-OPT)}.$$

Thus, the algorithm is a 2-approximation. ■

Having obtained a 2-approximation, we naturally want to know if we can do better. While this question has not been entirely resolved, we will show here that we cannot do better than a 2-approximation if the bound on the optimum uses only the LP solution. We exhibit an integrality gap of 2, i.e., we show that there are instances where the best integral solution is worse than the best fractional one by a factor arbitrarily close to 2.

The example consists of a complete graph K_n with edge weights of 1 for each edge. There are n labels, one corresponding to each vertex. The labeling cost is $c(v, v) = \infty$, and $c(v, u) = 0$ for all nodes $u \neq v$. That is, no node can be labeled with its own name, but any other labeling is free.

The best integral solution assigns each node label "1", except for node 1, which is given label "2". Then, exactly the edges incident with node 1 are cut, so the total cost is $n - 1$. On the other hand, a fractional solution can assign

$$x_{v,a} \quad = \quad \begin{cases} 0, & \text{if } a = v \\ \frac{1}{n-1}, & \text{if } a \neq v. \end{cases}$$

Then, the cost incurred by each edge (u, v) is $y_{u,v} = \frac{1}{n-1}$, so the total cost is $\frac{1}{n-1}\binom{n}{2} = \frac{n}{2}$. Hence, the integrality gap is $\frac{n-1}{\frac{n}{2}}$. As $n \to \infty$, this integrality gap approaches 2.

4.3 Uniform Labeling via Local Search

We can avoid having to solve an LP (which, while polynomial, tends to be fairly slow), by using a different 2-approximation algorithm based on Local Search where search moves use Min-Cut computations [65, 373]. In local search algorithms, we start with a solution, and repeatedly apply one of several simple moves, as long as it leads to an improvement. When no more improvement is possible with the simple moves, the algorithm terminates. It is rare that provable guarantees can be shown for efficient local search algorithms, but this is one of the cases.

Here, a local search move works as follows: We pick a label a and try if converting other vertices to that label will reduce the total cost. So we may label an arbitrary additional set of vertices with a, but no vertex gets its label changed to anything except a in one step. Among all such new labelings, we choose the best one, and then iterate over all labels. We thus obtain the following algorithm:

Notice that in this algorithm, the labeling f_a can be found using a single Min-Cut computation. Indeed, the algorithm is based on an insight by Greig et al. [190], who showed that for just two labels, the optimum solution can be found in polynomial time using a single Min-Cut computation.

Remark 4.3 Perhaps the most natural local search algorithm would only consider relabeling one vertex at a time, or a constant number. It is easy to see that any such algorithm can get stuck in highly suboptimal local minima, and no approximation guarantee can be proved. Allowing wholesale relabeling of many vertices is crucial.

Algorithm 4 Local Search

```
1: Start with an arbitrary labeling $f$ (e.g., the best individual vertex labeling $f(v) = \text{argmin}_a c(v,a)$).
2: repeat
3:     for all labels $a$ in turn (e.g., round robin) do
4:         Let $f_a$ be the best assignment with $f_a(v) = \begin{cases} a, & \text{if } f(v) = a \\ a \text{ or } f(v), & \text{if } f(v) \neq a. \end{cases}$
5:         Update $f = f_a$.
6: until no more improvement.
```

4.3.1 Approximation Guarantee

Theorem 4.4 *The algorithm produces a 2-approximation.*

Proof. The idea behind the proof is to show that the termination condition — the fact that no single-label relabeling yielded any more improvement — is enough to be close to the optimum solution. Roughly, we will do this by "decomposing" the optimum solution into the algorithm's solution.

Specifically, let f be the algorithm's solution and f^* the optimal solution. For each a, we let $S_a = \{v \mid f^*(v) = a\}$ be the set of all nodes that the optimum labeled a. We define a labeling f_a that "interpolates" between f and f^*, by saying that $f_a(v) = a$ if $f^*(v) = a$, and $f_a(v) = f(v)$ otherwise. That is, the interpolating labeling agrees with f^* whenever f^* chose a, and with the algorithm's labeling otherwise.

Because f_a was a candidate for relabeling from f, the termination of the algorithm implies that $\gamma(f) \leq \gamma(f_a)$ for all labels a. We now divide the cost of f into that incurred inside S_a, outside S_a and across the boundary. So we define

$$\gamma_S(f) \quad := \quad \sum_{v \in S} c(v, f(v)) + \sum_{(u,v) \in S \times S, f(u) \neq f(v)} w_{(u,v)}$$

for any set S. As a result, we can rewrite $\gamma(f)$ and $\gamma(f_a)$ as

$$\begin{aligned}
\gamma(f) &= \gamma_{S_a}(f) + \gamma_{\overline{S_a}}(f) + \sum_{(u,v) \in S_a \times \overline{S_a}, f(u) \neq f(v)} w_{(u,v)}, \\
\gamma(f_a) &= \gamma_{S_a}(f_a) + \gamma_{\overline{S_a}}(f_a) + \sum_{(u,v) \in S_a \times \bar{S_a}, f_a(u) \neq f_a(v)} w_{(u,v)}.
\end{aligned}$$

Above, we argued that $\gamma(f) \leq \gamma(f_a)$, and because the two labelings agree on the set $\overline{S_a}$, we have that $\gamma_{\overline{S_a}}(f) = \gamma_{\overline{S_a}}(f_a)$. On the other hand, f_a and f^* agree on S_a, so $\gamma_{S_a}(f_a) = \gamma_{S_a}(f^*)$. Taken together, this implies that

$$\gamma_{S_a}(f) + \sum_{(u,v) \in S_a \times \overline{S_a}, f(u) \neq f(v)} w_{(u,v)} \quad \leq \quad \gamma_{S_a}(f^*) + \sum_{(u,v) \in S_a \times \overline{S_a}, f_a(u) \neq f_a(v)} w_{(u,v)}.$$

But under the last sum, whenever $f_a(u) \neq f_a(v)$, then also $f^*(u) = a \neq f^*(v)$, so we can upper bound the sum by replacing $f_a(u)$ and $f_a(v)$ with $f^*(u)$ and $f^*(v)$. Thus, we have derived

$$\gamma_{S_a}(f) + \sum_{(u,v) \in S_a \times \overline{S_a}, f(u) \neq f(v)} w_{(u,v)} \quad \leq \quad \gamma_{S_a}(f^*) + \sum_{(u,v) \in S_a \times \overline{S_a}, f^*(u) \neq f^*(v)} w_{(u,v)}.$$

Because the S_a for all a form a disjoint cover of all nodes, summing up over all a now gives us that

$$\begin{aligned}
\gamma(f) &\leq \sum_a \gamma_{S_a}(f) + \sum_a \sum_{(u,v)\in S_a\times\overline{S_a}, f(u)\neq f(v)} w_{(u,v)} \\
&\leq \sum_a \gamma_{S_a}(f^*) + \sum_a \sum_{(u,v)\in S_a\times\overline{S_a}, f^*(u)\neq f^*(v)} w_{(u,v)} \\
&\leq 2\gamma(f^*),
\end{aligned}$$

because in the last sum, each edge between differently labeled nodes is counted exactly twice, while the assignment cost for each node is counted exactly once. So the algorithm is a 2-approximation. ■

4.3.2 Running Time

For a given labeling f, we let $\gamma(f) = \sum_v c(v, f(v)) + \sum_{e=(u,v), f(v)\neq f(u)} w_e$ denote its total labeling cost. The algorithm only chooses f_a over f if it is better, so whenever changes happen, γ is strictly decreasing. As a result, the algorithm cannot loop. Further, if all w_e and $c(v, a)$ are integers, then the decrease will be at least 1 in each iteration, so there can be at most $W = \sum_e w_e$ iterations. Therefore, the algorithm runs in pseudopolynomial time.

In order to make the running time actually polynomial, we can apply a standard trick (see, e.g., [22]): only perform an update step if it decreases the cost by a factor of at least $(1-\frac{\epsilon}{p(n)})$ for some polynomial p and some small constant ϵ. At the cost of an additional factor depending on ϵ in the approximation guarantee, this ensures that the number of steps is at most $\log W / \log \frac{1}{1-\epsilon/p(n)}$. (The analysis for this becomes a bit more messy, though.) At this point, it appears to be open whether this or a similar local search algorithm runs in strongly polynomial time, i.e., time not depending on W at all.

4.4 Further Reading

In the presentation of approximation algorithms, we focused on the case of uniform metric labeling. The paper by Kleinberg and Tardos [245] also gives an approximation algorithm for arbitrary metrics $d(a, a')$ on labels a, a'. The idea is to first probabilistically embed the metric into a hierarchically well-separated tree metric with low distortion. A hierarchically well-separated tree metric is defined by assigning distance labels on the edges of a tree. These distance labels must decrease exponentially from the root to the leaves. For each node pair, their distance is the length of the unique path in the tree. The distortion of such an embedding here is the largest factor by which the distance of any pair a, a' increases when going from the metric d to the tree metric. (Here, the tree metric has to be such that no distance shrinks.)

[245] shows how a modification of the LP-rounding based approach presented in Section 4.2 gives a constant-factor approximation for hierarchically well-separated tree metrics. Combining this result with the fact that each metric on k points can be probabilistically embedded into hierarchically separated tree metric with expected distortion $O(\log k \log\log k)$ [35, 34] now provides an $O(\log k \log\log k)$ approximation algorithm if there are k different labels to assign. Subsequently, Fakcharoenphol et al. [148] improved the approximation guarantee for probabilistic embeddings into hierarchically well-separated trees to $O(\log k)$, immediately implying the same approximation guarantee for the algorithm of Kleinberg and Tardos.

Several papers focus on algorithms for particular types of metrics. For example, Boykov, Veksler and Zabih [64] show that if the metric is the linear metric $d(a, a') = |a - a'|$, which applies for instance to pixel intensities, then an optimum solution can be found using minimum-cut computations. Gupta and Tardos [193] consider the variation of the truncated linear metric $d(a, a') = \min(M, |a - a'|)$, which assigns all pairs beyond a certain maximum distance the same penalty. For this version, they show that a variation of the local search algorithm presented in Section 4.3 gives a 4-approximation in polynomial time.

Chekuri et al. [87] give an interesting novel LP-formulation for the general metric problem. As special cases, they recover a 2-approximation for the uniform metric, an $O(\log k)$ approximation for arbitrary metrics, and a $2 + \sqrt{2}$ approximation for truncated linear metrics.

The objective functions discussed in this chapter are all concave. This corresponds to penalties increasing more slowly as the assigned labels grow more and more different. If the increase in penalties instead grows more steep, the “metric” d will instead be convex. In fact, this simplifies the problem significantly. Hochbaum [202] shows that if the function $d(a, a')$ is convex (in fact, even if there are different such convex functions on each edge), and each node penalty function $c(v, a)$ is convex in a, then the problem can be solved optimally in polynomial time.

The idea of using cut-based algorithms for classification has proved useful in other applications as well. In particular, Blum and Chawla [51] propose using it in the context of semi-supervised learning [83], where a small set of labeled examples for learning is augmented by many more unlabeled training examples. These unlabeled examples can be labeled by considering a graph representation of similarities. If the labels are only binary (“Yes” or “No”), then the best labeling of the additional instances can be obtained using a Min-Cut algorithm.

Classification problems of this type have been studied extensively in the literature on machine learning. For example, Taskar et al. [366] study the problem for learning the weights of edges (or, more generally, cliques) from example data. They also propose heuristics based on the popular belief-propagation algorithm [324] for the classification problem. However, these heuristics come with no provable guarantees. Subsequent to the paper by Taskar et al., several papers have proposed heuristics for dealing with sparse labels in a networked classification task. For some examples and overviews, see [169, 306, 270, 167].

Chapter 5

Rank Aggregation and Meta-Search

In previous chapters, we had discussed the problem of searching for relevant results on the WWW by exploiting the link structure. Nowadays, there are already multiple search engines giving quite good results. However, given that there are so many search engines already, employing different techniques, we may be interested in combining their positive features, and constructing a meta-search engine that uses all of their results [351, 283]. This leads very naturally to the problem of *rank aggregation*: given several orders on items (such as search results), determine a *consensus ordering*, which somehow reflects all orderings together.

This problem can be studied from different perspectives. We could treat it as a machine learning problem, by treating each search engine as an "expert", and trying to learn which engine's advice to trust, based on past performance. The goal would then be to perform no worse than the best expert, but without knowing ahead of time which one is actually the best expert.

Another approach is to consider it as an optimization problem. By defining an appropriate notion of distance between rankings, we can then look for a ranking that is close to all given rankings in a certain sense.

A third approach, and the one we begin with, exploits the close connection between rank aggregation and voting. In both settings, we have items (candidates or web pages), and orderings on them (voter preferences or search engine rankings). The goal is to find a ranking that is "agreeable" to all voters/search engines. By using this analogy, we can leverage several centuries of thought on the issue of voting and social choice.

5.1 Rank Aggregation as Social Choice

To get a feel for the difficulties in determining a consensus ordering, let us look at a simple example where a first choice is to be determined. Suppose there are three candidates, GWB, AG, and RN, and the voters' preferences are as follows:

- 49% of voters prefer the order GWB–AG–RN
- 48% of voters prefer AG–GWB–RN
- 3% of voters prefer RN–AG–GWB

Who should be considered the winner legitimately? There are different plausible answers, depending on the view we take of these preferences. We could argue that GWB was the candidate desired as winner by the largest number of voters, and hence should be the winner. This is the outcome of plurality voting. On the other hand, we could argue that a majority of voters prefers AG over GWB, and also over RN. Hence, AG wins all pairwise comparisons, and should be the winner.

Over the years, different rules have been proposed for deciding on a winner, or even a complete ranking. One of the earliest is due to Borda [116]. His method essentially uses the average position of a candidate, averaged over all voters' preferences, and sorts candidates in this order. More formally, for each voter, each candidate obtains a score equal to the number of other candidates he beat in this voter's ranking. Candidates are then ranked by the sum of their scores, summed over all voters. In our above example, this would give us AG as the winner, as the score would be $49 \cdot 1 + 48 \cdot 2 + 3 \cdot 1 = 148$, whereas the score of GWB is $49 \cdot 2 + 48 \cdot 1 = 146$ (and the score of RN is $3 \cdot 2 = 6$).

Although Borda's rule looks useful at first, a major problem with it is that even the candidate with the largest number of pairwise wins can lose. For instance, if 51% of the voters prefer the order ABC, and 49% prefer BCA, A's score is $51 \cdot 2 = 110$, while B scores $49 \cdot 2 + 51 = 149$, thus winning the election though clearly more people prefer A to B than vice versa.

Taking the latter idea further, we may wish to attain the following property, called *Condorcet Criterion* (after N. de Condorcet): If candidate A beats candidate B in pairwise comparison (i.e., more voters prefer A to B than vice versa), then A should precede B. We quickly run into a problem, already observed by Condorcet [117]: If three voters have the preference orders ABC, BCA, and CAB on three candidates A,B,C, then A should precede B, B should precede C, and C should precede A. Obviously, not all three can be accomplished simultaneously.

A relaxed version of the property, called *Extended Condorcet Criterion (XCC)* was proposed by Truchon [368], and requires the following: If X, Y are a partition of the set V of all candidates (i.e., $X \cap Y = \emptyset$ and $X \cup Y = V$), and for all $x \in X, y \in Y$, x beats y in direct comparison, then all of X should precede all of Y.

This version is much less restrictive; in particular, for the example given above, it allows us to choose an arbitrary ranking of candidates. In fact, we can show that for any input orderings, there is always an ordering satisfying the XCC.

Proposition 5.1 *For any input rankings, there is a consensus ordering satisfying the XCC.*

Proof. Consider the directed graph G on V with an edge from x to y if x beats y in pairwise comparison (we also write $x > y$ for this). Notice that this is a *tournament graph*, i.e., a graph in which for each pair of nodes, there is an edge one way or the other. We prove below that every tournament graph contains a Hamiltonian Path. Consider the candidate ordering determined by the sequence of vertices along such a Hamiltonian Path. Assume that this ordering fails the XCC. Then, there is a partition (X, Y) of V such that each $x \in X$ beats each $y \in Y$ in direct comparison, yet some $y \in Y$ precedes some $x \in X$ in this ordering. Then, there must also be an *adjacent* such pair in the ordering, i.e., one where y immediately precedes x on the Hamiltonian Path. But this is a contradiction, as there must have been an edge from y to x in the Hamiltonian Path, so y must actually beat x in direct comparison. ■

Lemma 5.2 *Each tournament graph G contains a Hamiltonian Path.*

Proof. We use induction on the number of vertices, n. For $n = 1$, the claim is trivial. For $n \geq 2$, let $x \in V$ be an arbitrary vertex. By induction hypothesis, $G[V - x]$ has a Hamiltonian path $x_1, x_2, \ldots, x_{n-1}$. If there is an edge from x to x_1, or from x_{n-1} to x, then insert x at the beginning or end of the ordering, respectively. Otherwise, there must be a k with $1 \leq k < n - 1$, such that there is an edge from x_k to x, and from x to x_{k+1} (start from x_1, and follow the path until a node has an edge from x). By inserting x between x_k and x_{k+1}, we obtain a Hamiltonian path for G. ■

Truchon [368] proves Proposition 5.1 in a different way, by showing that a Kemeny order also satisfies the XCC. A Kemeny order [229] is defined as follows: for each ordered pair (x, y) of alternatives, let $w_{x,y}$ be the number of voters who prefer x over y. A Kemeny order then minimizes the total sum of weights $w_{x,y}$ going "backwards", i.e., going from x to y with $x > y$ in the ordering. We will prove this fact in Lemma 5.7 below.

5.1.1 Formalization of Social Choice Properties

So far, we have investigated several concrete approaches for determining a consensus ordering from several given orderings. All approaches suffered from "unnatural" outcomes in some cases or others. Perhaps, it would thus make sense to axiomatize which properties a voting scheme should satisfy, and then look for voting schemes meeting these axioms. We describe here the axioms proposed by Arrow [20].

Formally, we will associate with each voter i a preference order $\prec_i$. A *social choice function* f takes all of these k orders, and outputs a consensus order

$$\prec \quad = \quad f(\prec_1, \ldots, \prec_k).$$

Such a function f should intuitively satisfy certain properties.

Monotonicity : If $a \prec b$, and $b \prec_i a$, then swapping a and b in $\prec_i$ does not result in $b \prec a$. Intuitively, this means that if a ranks above b overall, then changing another vote in a's favor does not affect the relative ranking between a and b.

Non-triviality : For each pair a and b of candidates, there is some choice of orderings $\prec_i$ such that $a \prec b$. This ensures that the relative ordering of a and b actually depends on the votes, and is not predetermined.

Independence of Irrelevant Alternatives (IIA) : Let $\prec_1, \ldots, \prec_k$ and $\prec'_1, \ldots, \prec'_k$ be two different preference orders for each voter, and $\prec = f(\prec_1, \ldots, \prec_k)$ and $\prec' = f(\prec'_1, \ldots, \prec'_k)$ the corresponding consensus orderings. If $B \subseteq V$ is a subset of candidates such that $a \prec_i b$ if and only if $a \prec'_i b$ for all $a, b \in B$ (i.e., all $\prec_i$ and the corresponding $\prec'_i$ agree on the orderings of B), then $a \prec b$ if and only if $a \prec' b$ for all $a, b \in B$. What this expresses is that if no voter changes his relative preference between any two candidates in B, then the final ordering among candidates of B does not change. In other words, changing preferences merely with regards to third candidates does not affect the order of any two other candidates.

Monotonicity and non-triviality together imply the property of *unanimity*: if $a \prec_i b$ for all i, then $a \prec b$. That is, if every voter prefers a over b, then a ends up ahead of b. To prove this fact, start with some set of orders $\prec_i$ such that $a \prec b$ (such a set exists by the non-triviality property). Then, we keep swapping the positions of a and b in all $\prec_i$ that previously had $b \prec_i a$. By monotonicity, the outcome will still be that a is ranked ahead of b, and eventually, $a \prec_i b$ for all i.

In trying to find social choice functions satisfying all these axioms, one quickly notices that this is not so easy. In particular, the IIA property is not satisfied by many schemes. However, one class of social choice functions meeting these requirements is *dictatorship*: the dictator function f_i is defined as $f_i(\prec_1, \ldots, \prec_k) = \prec_i$. That is, the output is simply the preference of just *one* voter. Dictatorship is an undesirable quality for a voting scheme. That gives us our last property:

Non-Dictatorship : $f \neq f_i$ for all i. That is, the aggregation will not disregard the opinions of all but one voter.

Unfortunately, with this additional requirement, we have ruled out all remaining social choice functions:

Theorem 5.3 (Arrow, 1951 [20]) *There is no function f satisfying all the above four properties. Hence, the only functions satisfying the first three properties are the dictatorship functions.*

Proof. To prove Arrow's theorem, we show that any social choice function satisfying monotonicity, non-triviality, and IIA is in fact a dictatorship function. We will do this by first proving the existence of a single voter who can decide the order between two candidates; then, we prove that this voter is in fact a dictator.

First, we define sets that decide the outcome between two candidates. We call a set J of voters (A, B)-*decisive* if the fact that all of J ranks A ahead of B is enough to guarantee that A will be ranked ahead of B in the output. Formally, we write $A \prec_J B$ to denote that $A \prec_i B$ for all $i \in J$. We then say that J is (A, B)-decisive iff $A \prec_J B$ implies $A \prec B$. We call a set J of voters *decisive* iff J is (A, B)-decisive for some pair (A, B).

One useful characterization of decisiveness can be derived from the monotonicity property: J is (A, B)-decisive if and only if $A \prec_J B$ and $B \prec_{\bar{J}} A$ imply that $A \prec B$. We will use this characterization later. Notice also that by the unanimity property, the set V of all voters is always (A, B)-decisive for each pair (A, B). Hence, there must be some smallest decisive set J^*. We will prove that J^* is actually a singleton.

Let $\{A, B\}$ be the alternatives such that J^* is (A, B)-decisive. Consider an arbitrary voter $i \in J^*$ from this set, and define $J' = J^* - \{i\}$. Assume that the voter i has preference order CAB, each voter in J' has order ABC, and each voter in $\bar{J}^*$ has order BCA. (Other candidates can be ignored by the IIA property). In the final order, A must precede B, because J^* is (A, B)-*decisive*, and all of J^* ranks A ahead of B. In addition, we know that C must precede A. The reason is that only J' would prefer A to precede C, so if it actually did, then J' would be (A, C)-decisive, contradicting the assumption that J^* is a smallest decisive set. Now, by transitivity, we conclude that the final output must be CAB, so C precedes B. But i is the only voter preferring C to precede B, so by our characterization above, $\{i\}$ must be (C, B)-decisive. Because $\{i\}$ is thus decisive, it cannot be smaller than J^*, so we know that $J^* = \{i\}$, and $\{i\}$ is also (A, B)-decisive.

Next, we show that $\{i\}$ is also (A, D)-decisive for all $D \neq A$ and (C, D)-decisive for all $D \neq C$. We consider the scenario where voter i has the order ABD, and everyone else has order BDA. Then, by unanimity, $B \prec D$, and because $\{i\}$ is (A, B)-decisive, we have $A \prec B$. By transitivity, the ordering is ABD. In particular, this means that A precedes D, which only voter i prefers. Thus, $\{i\}$ must be (A, D)-decisive. Replacing A by C in the previous proof shows that $\{i\}$ is also (C, D)-decisive. Also, by substituting C resp. A for D, we further conclude that $\{i\}$ must be (C, A)-decisive and (A, C)-decisive.

Next, we show that $\{i\}$ is also (D, A)-decisive for all $D \neq A$, as well as (D, C)-decisive for all $D \neq C$. Here, we assume that voter i prefers the order DCA, and everyone else prefers ADC. By unanimity, we have $D \prec C$ in the outcome, and because $\{i\}$ is (C, A)-decisive, we have $C \prec A$. So the final order will be DCA. Again, i is the only voter preferring D over A, so $\{i\}$ must be (D, A)-decisive. Since $\{i\}$ is both (A, C)-decisive and (C, A)-decisive, we can simply switch the order A and C in the previous construction, and prove that $\{i\}$ is (D, C)-decisive as well.

As a final step, we show that $\{i\}$ is in fact (D, E)-decisive for all D, E, and hence a dictator. We assume that i votes DAE, and everyone else votes EAD. Because $\{i\}$ is (D, A)-decisive and (A, E)-decisive, the final ordering must be DAE. But by the same argument as before, this means that $\{i\}$ is (D, E)-decisive.

In summary, we have proved that voter i is a dictator, completing our proof of Arrow's Theorem. ■

This result is quite disappointing. It suggests that for a very reasonable definition of what democratic decision-making is, the process is impossible. Yet, we observe frequently in practice that voting does work (reasonably) well. So one direction to pursue further is to ask: what kind of restrictions do voter preferences in practice seem to satisfy? Our construction in the proof was based on some very carefully crafted scenarios — perhaps, those don't appear in practice.

This line of thought is pursued further in several papers. Black [49, 50] suggests single-peaked preferences on the line, i.e., each voter is located on a one-dimensional space (such as the political spectrum from "left" to "right"), and ranks preferences by distance from his own location. Barberà et al. [33] extend this to higher-dimensional spaces with the L_1-norm, and Richards et al. [338] extend it further to graph structures shared by all agents, in which agents have single-peaked preferences at nodes. In these cases, consensus voting is in fact possible, i.e., Arrow's Axioms can be achieved.

5.2 Aggregation as Optimization

A different approach from the axiomatic one (which, in a sense, we just saw fail) would be to treat the determination of a consensus ordering as an optimization problem: to find a ranking that is as close as possible to the given set of rankings.

In order to phrase the problem this way, we first need to define a notion of distance between two rankings. As rankings are really nothing but permutations (so long as they are complete rankings — more about partial rankings later), we can look at known distance measures on permutations. Two well-known such measures are Spearman's Footrule and Kendall's τ.

Definition 5.4 (Spearman's Footrule) *Let $\prec_1$, $\prec_2$ be two orderings, and $\prec_1 (a)$ the position of element* *a in the ordering $\prec_1$. Then, the Footrule distance is defined as*

$$F(\prec_1, \prec_2) \quad = \quad \sum_a | \prec_1 (a) - \prec_2 (a)|$$

Notice that this is identical to the L_1 distance between the vectors of positions.

Definition 5.5 (Kendall's τ) *Kendall's τ counts the number of inversions (or Bubble Sort swaps) between the two permutations. Writing*

$$K_{a,b}(\prec_1, \prec_2) \quad = \quad \begin{cases} 1 & \text{if } a \prec_1 b \text{ and } b \prec_2 a \\ 0 & \text{otherwise,} \end{cases}$$

we define

$$\tau(\prec_1, \prec_2) \quad = \quad \sum_{a,b} K_{a,b}(\prec_1, \prec_2).$$

The maximum value that can be attained by these metrics is $n(n-1)$ for Spearman's Footrule, and $\frac{n(n-1)}{2}$ for Kendall's τ. In both cases, the maximum is attained for two orderings that are the reverse of each other.

It is easy to see that the two metrics are not the same. One obvious example is when $\prec_1 = \langle 1\ 2 \rangle$, and $\prec_2 = \langle 2\ 1 \rangle$. Then, the Spearman Footrule distance is $F = 2$, while the Kendall distance is $\tau = 1$. While the two can be different, we can prove that they are not *very* different.

Theorem 5.6 (Diaconis and Graham [124]) *For any orderings $\prec_1$ and $\prec_2$, we have*

$$\tau(\prec_1, \prec_2) \quad \le \quad F(\prec_1, \prec_2) \quad \le \quad 2\tau(\prec_1, \prec_2).$$

Proof. We first show that $F(\prec_1, \prec_2) \le 2\tau(\prec_1, \prec_2)$. We do this by induction on the value of $\tau(\prec_1, \prec_2)$. In the base case, when $\tau(\prec_1, \prec_2) = 0$, both the orderings are the same, and hence $F(\prec_1, \prec_2) = 0$.

In the induction step, we look at $\prec_1$ and $\prec_2$ such that $\tau(\prec_1, \prec_2) > 0$. Let $\prec'$ be obtained from $\prec_2$ by one switch towards $\prec_1$. Then, $\tau(\prec_1, \prec') = \tau(\prec_1, \prec_2) - 1$, and $\tau(\prec', \prec_2) = 1$, $F(\prec', \prec_2) = 2$. By the Triangle Inequality, applied to the metric F, we have that

$$F(\prec_1, \prec_2) \quad \le \quad F(\prec_1, \prec') + F(\prec', \prec_2) \quad = \quad F(\prec_1, \prec') + 2\tau(\prec', \prec_2).$$

We apply the Induction Hypothesis to $\prec_1$ and $\prec'$, obtaining that $F(\prec_1, \prec') \le 2\tau(\prec_1, \prec')$. Thus,

$$F(\prec_1, \prec_2) \quad \le \quad F(\prec_1, \prec') + 2\tau(\prec', \prec_2) \quad \le \quad 2\tau(\prec_1, \prec') + 2\tau(\prec', \prec_2) \quad = \quad 2\tau(\prec_1, \prec_2),$$

completing the inductive proof.

For the other inequality, $\tau(\prec_1, \prec_2) \le F(\prec_1, \prec_2)$, we use induction on $F(\prec_1, \prec_2)$. In the base case $F(\prec_1, \prec_2) = 0$, both the orderings are the same, so $\tau(\prec_1, \prec_2) = 0$.

For the inductive step, we have the problem that simple switches will not necessarily improve the value of F. As an example, we can look at $\prec_1 = \langle 1\ 2\ 3\ 4 \rangle$ and $\prec_2 = \langle 4\ 3\ 2\ 1 \rangle$. In this case, a switch towards $\prec_1$ will result in the ordering $\prec' = \langle 4\ 3\ 1\ 2 \rangle$, which has $F(\prec_1, \prec') = F(\prec_1, \prec_2)$. Thus, here we try to "meta-swap" two elements such that one is too far to the left of its position in $\prec_1$ and the other is too far right of its position in $\prec_1$.

Without loss of generality, we may assume that $\prec_1 = \langle 12 \ldots n \rangle$ (i.e., the elements are sorted), and $\prec_2$ has element i in position a_i. Thus $F(\prec_1, \prec_2) = \sum_i |a_i - i|$.

Let i be maximal such that $a_i \neq i$, i.e., the rightmost element that is out of position. Notice that this implies that $a_i < i$, for otherwise, the element a_i (the one that is in position a_i in the ordering $\prec_1$) would be a larger index out of place. Let $j \le a_i$ be the largest index with $a_j > a_i$ and $a_j > j$, i.e., the rightmost element to the left of i in the order $\prec_2$ which is too far right. (See Figure 5.1 for an illustration.) Notice

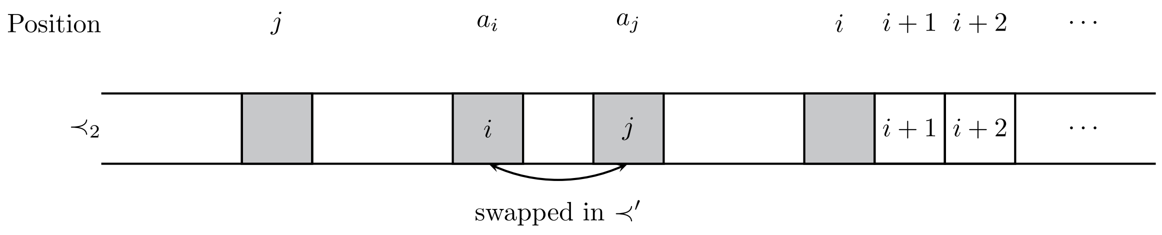


Figure 5.1: An illustration of the second half of the proof. Element i is in position $a_i < i$. Element j is in position $a_j > j, a_j > a_i$, and satisfies $j \leq a_i$. Elements i and j (in positions a_i and a_j) are swapped from $\prec_2$ to $\prec'$.

that such an element j must exist by the Pigeonhole Principle: at most $a_i - 1$ of the elements $1, \ldots, a_i$ can be in positions $1, \ldots, a_i - 1$. Also notice that $a_j \leq i$, as otherwise, the element a_j, which is out of position, would show that i is not maximal.

Let $\prec'$ be the ordering obtained by swapping i and j in the ordering $\prec_2$. Notice that in $\prec'$, neither i nor j "overshoot" their actual positions, because $j \leq a_i$ and $a_j \leq i$. Thus, both i and j move $|a_i - a_j|$ positions closer to their destination, and all other elements stay in position. We obtain that $F(\prec_1, \prec_2) = 2|a_i - a_j| + F(\prec_1, \prec')$. Applying the Induction Hypothesis, we thus have that $F(\prec_1, \prec_2) \geq 2|a_i - a_j| + \tau(\prec_1, \prec')$.

We can also calculate the Kendall τ distance between $\prec_2$ and $\prec'$ quite easily. By making $|a_i - a_j|$ switches to the right of element i, we get it into position. Then, another $|a_i - a_j| - 1$ switches to the left of element j move it to position a_i. All elements in between are moved once to the left and once to the right, and thus end up in the same position. Hence, we just proved that $\tau(\prec', \prec_2) \leq 2|a_i - a_j| - 1$. Now, by the Triangle Inequality for the metric τ, we obtain that

$$\tau(\prec_1, \prec_2) \quad \leq \quad \tau(\prec_1, \prec') + \tau(\prec', \prec_2) \quad \leq \quad \tau(\prec_1, \prec') + 2|a_i - a_j| - 1 \quad < \quad F(\prec_1, \prec_2).$$

This completes the inductive proof. ■

Now that we have metrics to measure the difference between two orderings, given a list of several orderings $\prec_1, \prec_2, \ldots, \prec_k$, we want to find an ordering $\prec$ "close to all" of them. There are multiple concrete optimization criteria we could be trying to minimize, for example:

1. The average τ distance $\frac{1}{k} \sum_i \tau(\prec, \prec_i)$.
2. The average Footrule distance $\frac{1}{k} \sum_i F(\prec, \prec_i)$.
3. The maximum τ distance $\max_i \tau(\prec, \prec_i)$.
4. The maximum Footrule distance $\max_i F(\prec, \prec_i)$.

Notice that the first definition exactly defines Kemeny orderings, as discussed in the previous section. For the sum of all Kendall τ distances is exactly the total number of disagreements between the ordering $\prec$ and all other orderings. Next, we show that a Kemeny ordering satisfies the extended Condorcet property.

Lemma 5.7 *If $\prec$ minimizes $\sum_i \tau(\prec, \prec_i)$, then it satisfies the extended Condorcet property.*

Proof. We prove the lemma by contradiction. Assume that we have an ordering $\prec$ which minimizes $\sum_i \tau(\prec, \prec_i)$, but does not satisfy the extended Condorcet property. Then, there is a partition $(S, \overline{S})$ of the alternatives $\{1, \ldots, n\}$ such that every alternative in S beats all those in $\overline{S}$ [1], yet $\prec$ ranks some $j \in \overline{S}$ ahead of some $i \in S$. Therefore, there must be such a pair such that i and j are adjacent in the ordering $\prec$. Now, swapping i and j improves $\sum_i \tau(\prec, \prec_i)$ by i's margin of victory. Since the relative orderings of no other elements are affected, the objective function actually improves, contradicting the optimality of $\prec$. ■

[1] Recall that i is said to *beat* j if there are more orderings in which i precedes j than vice versa.

Unluckily, minimizing the average τ distance is NP-hard, as proved by Dwork et al. [136] (we don't give the proof here).

Fact 5.8 *If $k \geq 4$, then minimizing $\sum_i \tau(\prec, \prec_i)$ is NP-hard.*

Instead, we may focus on the second objective function: minimizing $\sum_i F(\prec, \prec_i)$. This one can actually be minimized in polynomial time. The intuition is to look at an ordering/permutation as a matching between elements and positions. Then, we can express the total objective function value as a sum of "penalties" incurred by each element for the position it is in. Specifically, we assign a penalty $\phi_{j,p} = \sum_i |p - \prec_i (j)|$ for putting element j in position p. (Recall that $\prec_i (j)$ denotes the position in which element j appears in the order $\prec_i$). For each j, we thus need to assign a (distinct) $p(j)$ to minimize $\sum_j \phi_{j,p(j)}$. Since feasible assignments are exactly perfect matchings between elements and positions, our goal is to find the *cheapest* perfect matching in the complete bipartite graph where the edge (j, p) has cost $\phi_{j,p}$. Finding minimum-cost perfect matchings is known to be solvable in polynomial time, for instance by first computing any perfect matching, and then repeatedly eliminating negative cycles in the residual graph (see [9, 246] for discussions of polynomial algorithms for minimum cost matching).

Notice that the ordering minimizing the average Footrule distance is also a 2-approximation to the problem of minimizing the average Kendall's τ distance. For if OPT denotes the best Kendall ordering, and $\prec$ the best Footrule ordering, then

$$\textstyle\sum_i \tau(\prec, \prec_i) \quad \leq \quad \sum_i F(\prec, \prec_i) \quad \leq \quad \sum_i F(\text{OPT}, \prec_i) \quad \leq \quad 2 \sum_i \tau(\text{OPT}, \prec_i).$$

5.2.1 Partial Orderings

In the previous discussion, we assumed that the orderings were actually complete. What happens when we are not given full orderings? Obviously, this is relevant in combining search engine results, since different search engines will have different web crawls, containing different sets of pages in the search results. What techniques can we then use to compare these orderings? Fagin et al. [147] study the issue of how to compare top k lists. If the lists don't all contain all of the elements, one needs to extend (or replace) the distance notions defined above.

Fagin et al. suggest several techniques to deal with this. One is to augment all the lists such that all the elements appear in all the lists, by appending the missing elements at the end of each list (since they were clearly not considered to be in the top k by that list). This raises the question in which order the extra elements should be appended to the lists. An "optimistic" view would define the distance between lists based on the assumption that the missing elements appear in the same relative order as in the other list. Another solution is to append the elements in a random order, and define the distance as the average.

More generally, we can define an extension of Kendall's τ with a penalty p as follows. If both elements i, j are in one list, and at least one element is in the other list, then the transposition penalty for i, j is just the same as for Kendall's τ, i.e., 0 or 1 depending on whether their order is the same or different. (If only i is in the second list, then the penalty is 0 if i appears before j in the first list, and 1 otherwise.) If one list contains both i and j, and the other list contains neither, then the penalty is p. Then, $p = 0$ corresponds to the optimistic assumption, and $p = \frac{1}{2}$ to the random ordering of absent elements.

[147] shows that neither of these approaches for pairwise comparison of lists satisfies the triangle inequality, and hence, we do not obtain a metric. However, they show that the distance measures can be both upper and lower bounded by a metric.

Spearman's Footrule can also be generalized by introducing a new location ℓ, and assuming that all missing elements are at location ℓ, and calculating the normal Footrule distance. Notice that this does not really define a permutation any more. The most obvious choice would be $\ell = k + 1$, but other choices are possible. The resulting Footrule distance actually does define a metric.

5.3 Further Reading

Even though this chapter is motivated mostly by rank aggregation for search engines and similar applications, our initial example focused on elections. There, the goal is frequently different: rather than finding an aggregate ordering of *all* alternatives, the goal is solely to determine one winner. While we could design any number of simple rules (e.g., counting votes, runoff voting, etc.), there is now the issue that voters may reason that incorrectly stating their preferences might help them achieve a more desirable outcome. For instance, in our introductory example, knowing the distribution of other votes, the third type of voters (with the order RN-AG-GWB) might now declare AG as their first choice to avoid their least favorite outcome.

Hence, it becomes important to study which types of voting systems are strategy-proof against voters misrepresenting their preferences. The Gibbard-Satterthwaite Theorem [171, 345] gives an equally pessimistic characterization to Arrow's Theorem. It posits the following axioms:

1. **Fairness**: Each candidate can win if all voters unanimously prefer this candidate to all others.
2. **Strategy-Proofness**: No voter is ever better off misrepresenting his preferences.

The only voting rules satisfying these axioms (for three or more candidates) are again the dictator functions.

Again, an interesting question is what natural restrictions on the preferences would lead to non-trivial social choice functions. In this context, the survey paper of Barberà [32] gives a nice summary of results. Again, single-peaked preferences on the line and in higher dimensions with the L_1-norm (e.g., [301]) lead to non-trivial mechanisms that have voters truthfully revealing their preferences.

When we allow randomized social choice functions, the situation is not quite as dire as the one of the Gibbard-Satterthwaite Theorem. Gibbard [172] shows that a rule is strategy-proof only if it is a randomization over dictator functions and two-choice rules (i.e., rules that prune all but two candidates, and then decide on a winner among these two candidates by considering the votes). Conitzer [106] extends this result to the case when the voting rule also needs to be *anonymity-proof*, in the sense that no agent benefits by casting multiple votes. (This is necessary, for instance, for online polls, where the inherently anonymous environment cannot prevent a user from voting multiple times.) Conitzer shows that in this case, the remaining randomized social choice functions are significantly more restricted: the voting rule randomizes between picking a uniformly random alternative, or picking two alternatives, and then checking if all agents prefer one over the other. If so, that alternative is the winner; if not, then a fair coin is flipped. Thus, the outcome is — except for a few unanimous choices — essentially always random.

Dictator functions also play an important role in the Fourier Analysis of Boolean Functions [319]. Using the setup of Fourier Analysis, and a proof based on uniformly random orderings by all voters, Kalai [221] gives a very short proof of Arrow's Theorem. Kalai's survey [222] describes the connections between Fourier Analysis and Social Choice Theory in more depth, and relates them to the outcomes of elections with random choices.

Indeed, the impact of randomization already impacted Condorcet's description. He posits a model where each voter chooses between two alternatives, and makes the societally most beneficial choice with some probability $p > \frac{1}{2}$. Condorcet [117] then shows that the majority vote has the highest probability of identifying the right choice. This is a direct precursor to analysis showing that the Kemeny order, under a natural model of randomization, is most likely to be the societally preferred one. This is discussed in more detail in papers by Young [389] and Truchon [368].

Finding a Kemeny ordering is equivalent to the well-studied problem of finding a Minimum Feedback Arc Set. A set S of edges in a directed graph G is a Feedback Arc Set if and only if its removal breaks all cycles in G. Clearly, once all cycles are broken, the nodes can be ordered by any topological sort order, respecting all preferences. Thus, finding the smallest number of edges to remove in a (multi-)graph is equivalent to finding the ordering minimizing the total number of inversions.

In general, the best known approximation for Feedback Arc Set is $O(\log n \log \log n)$ [143]. However, for the case of rank aggregation, we know that the graph is a tournament graph, where an edge is directed only from the candidate who wins the pairwise comparison to the loser of that comparison. The edge weight is the difference in votes between the two relative orders. For this special case, Kenyon-Mathieu and Schudy provide a PTAS [236], improving on a previous (simpler) 3-approximation algorithm due to Ailon, Charikar, and Newman [12]. Ailon [11] shows how to extend these algorithms to the case when the rankings are only partial (as discussed in Section 5.2.1), and gives a 2-approximation, and a more complex $\frac{3}{2}$-approximation algorithm.

An issue somewhat similar in spirit to aggregating top-k rankings is the aggregation of rankings with ties. Fagin et al. [146] give extensions of the Kendall τ and Spearman Footrule measures for those cases, similar to the ones discussed for top-k lists in Section 5.2.1. Again, they generalize the result of Diaconis and Graham to show equivalence of the measures to within constant factors, and discuss the issue of aggregation.

A different approach to ranking is proposed by Hochbaum [203]. She posits that not only are we given pairwise comparisons, but for each pair (i, j) also a weight $w_{i,j}$ indicating *how much* one beat the other. These weights may be inconsistent, e.g., A beat B, B beat C, and C beat A. The goal is to "smooth" them to make sure that all numbers end up consistent, in the sense that for each pair of alternatives (i, j), all paths from i to j have the same sum of weights in the end. At the same time, the smoothed weights $w'_{i,j}$ should be "close" to the original ones. [203] shows that so long as the penalty function for deviations is convex, an optimal solution can be found in polynomial time using the dual of a minimum cost network flow problem [8].

The issue of learning orderings is addressed in more depth by Cohen et al. [102]. The idea is to use an *expert learning* [268, 269, 163] approach to finding scores for pairwise comparisons, and then find the ordering that agrees best with these scores. The learning approach is also pursued by Joachims [219], who uses Support Vector Machines to learn rankings from user preferences that are expressed in clickthrough data.

Chapter 6

Power-Law Distributions

In Section 1.2.3, we briefly remarked on the fact that the WWW graph appears to have a power-law degree distribution, i.e., the fact that the frequency of nodes of degree d decays polynomially as $d^{-\gamma}$ for some constant γ (as opposed to — say — exponentially). This observation clearly invalidated the Erdős-Rényi $G(n, p)$ model as a model for the WWW. In this chapter, we will investigate power-law distributions in much more detail, with a focus on generational models which would predict power laws. Most of the material in this chapter, and much more, is covered in the excellent survey of the topic by Mitzenmacher [287].

The existence of power law distributions has been observed empirically over the years in many natural and man-made scenarios. Both the observation of such distributions and possible generational models have received quite a lot of attention. A major surge in interest within the computer science (and physics) community resulted from the paper by Faloutsos, Faloutsos, and Faloutsos [149], which demonstrated power laws in a crawl of the Internet and WWW graphs. Some of the other contexts in which power laws had been observed previously include:

- Financial Models: price changes of securities in financial markets [274].
- Biology: lengths of protein sequences in genomes [216] or variation among plant species [353].
- Linguistics: word length frequencies [142, 395] and degrees of words in association networks [358].
- City populations or distribution of income among people [321].
- Number of papers published by scientists [307].

Formally, we define power law distributions as follows:

Definition 6.1 *A non-negative random variable X is said to have a* power law *distribution if* $\text{Prob}[X \geq x] = cx^{-\alpha}$*, for some constants $c, \alpha > 0$.*

If the variable is actually discrete, then the definition implies that $\text{Prob}[X = x] = c'x^{-\alpha'}$, for different values $c', \alpha' > 0$. We will use the definitions interchangeably.

Power law distributions fall into the class of *heavy tailed distributions*: the probability that X assumes a large value is only polynomially small, compared to the exponentially small probabilities for Gaussian, Binomial, or other common distributions.

When given an actual set of data, we can recognize it as a power law most easily in a log-log plot, i.e., in a plot where both axes scale logarithmically. For if $f(x) = c \cdot x^{-\alpha}$ denotes the frequency with which value x was observed, then $\log f(x) = \log(c) - \alpha \cdot \log(x)$. Hence, in a log-log plot, we will observe a straight line with a slope of $-\alpha$ and y-intercept of $\log(c)$. However, several important caveats should be noted:

1. In order to infer with confidence that a power law is indeed present, the straight line should extend over several orders of magnitude at least. A very short straight-line segment in a log-log plot does not indicate a power law very conclusively.

2. To determine the coefficient α, it is usually not right to fit the best straight line to the data, e.g., via regression, since the log-log scaling of the plot invalidates the stochastic assumptions behind the regression. A more in-depth discussion of this issue is given by Clauset et al. [99].

3. Indeed, a recent study [71], performing a rigorous statistical analysis on a large number of networks previously reported to possess power laws or "scale freeness," found very limited evidence that these networks were truly scale free or power law.

Power Laws in the WWW

For the WWW, the paper of Broder et al. [70] was the first to show that the distribution of in (resp. out) degree d is proportional to $d^{-\alpha}$ for some value $\alpha \in [2.1, 2.2]$ (resp. $\alpha \in [2.7, 2.8]$). Given the values of α, c, we can calculate the mean of the power law distribution as

$$\mu \quad = \quad \sum_d d \cdot c \cdot d^{-\alpha} \quad = \quad c \sum_d d^{1-\alpha}.$$

Notice that the mean is finite iff $\alpha > 2$. In particular, this implies that if the power law indegree distribution were to continue to hold as the WWW grows, the average indegree of pages would remain finite. The fact that the mean indegree of the WWW is finite is not too surprising, given that the average indegree equals the average outdegree, and we don't expect the average web page to have more than a constant number of outlinks, as each requires actual work.

However, there are natural examples of networks that exhibit power laws with $\alpha < 2$. For such networks, we would predict that if the power law persists as the size of the network grows, the mean would diverge to infinity:

- The WWW at a site level ($\alpha \approx 1.6$) [43]. Here, single sites can include large numbers of web pages, and we may expect the number of pages within a site to grow along with the web.

- Co-authorship in high0energy physics ([307], $\alpha \approx 1.2$). High-energy physics papers often have very large numbers of co-authors (in excess of 1000). It is not clear how to predict the scaling of this graph in the future, but it is not inconceivable that future papers may have even larger author lists. (Notice that by comparison, the mathematics co-authorship graph has a rather large value of $\alpha \approx 2.5$. Many mathematics papers still have a single author.)

6.1 Preferential Attachment

In trying to explain observed power laws, many models posit a "rich get richer" phenomenon: entities (such as nodes or web pages, dollars, plant genera, cities, individuals, etc.) that already have a large value (degree, number of species, populations, wealth, etc.) have a tendency to attract more of the same. If this attraction is linear in the current value, then power laws emerge.

In the case of the WWW or other graphs, such a behavior is called *Preferential Attachment*. It is posited that newly arriving nodes link to an existing node v with probability proportional to v's current degree. Thus, high-degree nodes are more likely to attract new links. The underlying assumption here is not necessarily that nodes make this choice deliberately, but rather that high-degree nodes (e.g., web pages with high indegree) are more likely to be discovered in the first place, and thus linked to. Kumar et al. [253] make this explicit by investigating a copying model, wherein newly arriving nodes randomly select another node, and copy some of that node's outlinks. A mathematically rigorous analysis of such models tends to be quite involved (and is carried out, for instance, in [253]). However, by making some non-rigorous simplifications, we can obtain qualitatively identical results.

Here, we study the following simple model: The graph starts with zero nodes. At each time $t \in \mathbb{N}$, a new node arrives and generates k outlinks; we will label the node with its arrival time t. Each outgoing link is either uniformly random, which happens with probability $1 - \alpha$, or preferential with probability α. In the latter case, the edge links to existing nodes with probabilities proportional to their degrees.

Our analysis will follow the outline by Barabási et al. [30, 31]. Let $d_i(t)$ denote the in-degree of node i at time t. Then, because node i arrives at time i with no links yet, we have that $d_i(i) = 0$. At any time t, the total number of inlinks (into all nodes) is kt, and the total number of nodes is t. Hence, the probability that node i is the endpoint of a given new link at time t is $\frac{1-\alpha}{t} + \alpha\frac{d_i(t)}{kt}$. As k new links are created[1], the expected change in the degree of node i in step t is $\frac{\alpha d_i(t)+k(1-\alpha)}{t}$.

We write $\beta = k(1-\alpha)$. The difference equation for expected degrees above can be rewritten as a differential equation (here, we are being non-rigorous in our approximation), giving us that

$$\frac{\partial d_i(t)}{\partial t} = \frac{\alpha d_i(t)+\beta}{t}$$

Rearranging and integrating we get,

$$\int \frac{\partial d_i(t)}{\alpha d_i(t)+\beta} = \int \frac{\partial t}{t},$$

which is easily seen to have solution $\frac{1}{\alpha}\ln(\alpha d_i(t)+\beta) = \ln(t) + c$. Solving for $d_i(t)$ now shows that $d_i(t) = \frac{t^\alpha \cdot e^{\alpha c} - \beta}{\alpha}$. We still have to find out the value of the constant c. To determine it, we can use the initial condition that $d_i(i) = 0$. This gives us that $\frac{i^\alpha e^{\alpha c} - \beta}{\alpha} = 0$, which by rearranging yields that $e^{\alpha c} = \beta \cdot i^{-\alpha}$. Substituting this back into the expression for $d_i(t)$ now shows that

$$d_i(t) = \frac{\beta}{\alpha}\left((t/i)^\alpha - 1\right).$$

To find the cumulative function for the number of nodes with degree greater than or equal to d, we first solve the inequality $d_i(t) \leq d$. This yields that the expected degree is at most d whenever $i \geq t \cdot (d \cdot \frac{\alpha}{\beta} + 1)^{-\frac{1}{\alpha}}$. Thus, the fraction of nodes with degree greater than d at time t is

$$\frac{t-i}{t} = \frac{t - t\cdot(d\cdot\frac{\alpha}{\beta}+1)^{-\frac{1}{\alpha}}}{t} = 1 - (d\cdot\frac{\alpha}{\beta}+1)^{-\frac{1}{\alpha}}.$$

To obtain the density function from this, we take the derivative with respect to d, which gives us density $\frac{1}{\beta}\cdot\left(d\cdot\frac{\alpha}{\beta}+1\right)^{-(1+\frac{1}{\alpha})}$.

Thus, we observe a power law with exponent $1 + \frac{1}{\alpha} > 2$. In particular, for $\alpha \approx 0.9$, we obtain the same degree distribution as for the WWW.

Notice that for $\alpha = 0$, the above analysis breaks down (and the result is meaningless). Indeed, for $\alpha = 0$, nodes never attach preferentially, but always choose uniformly at random. Older nodes will still end up with higher degree, as they are around for more rounds, and can thus receive more incoming edges. Notice that each node i, in each round t, receives an expected k/t new incoming links (as there are t competing nodes). Hence, after t rounds, node i will have expected indegree $\sum_{j=i+1}^{t} \frac{k}{j} \approx k\cdot\log(t/i)$. The indices i having degree at most d are thus $i \geq t\cdot e^{-d/k}$, and the fraction with degree at least d is $1 - e^{-d/k}$. Taking a derivative with respect to d gives us a density function of $\frac{1}{k}\cdot e^{-d/k}$, which is sharply concentrated. In particular, we will not obtain a power law. So the age advantage alone does not explain the power law distribution derived above; rather, the preferential attachment was a crucial part of the model.

Notice that the preferential attachment model can be easily extended to include deletion of nodes and edges and rewiring etc., and the same type of analysis based on differential equations can be carried out. Usually, the power law degree distribution is preserved under these modifications, although they usually result in a large number of parameters, whose relative sizes affect the exact value of the exponent.

While preferential attachment gives us the same degree distribution as was observed for the WWW, it fails to obtain several other key properties (among others, all graphs generated are acyclic). For instance, as the links are generated independently at random, the graph will likely not exhibit much community structure (such as triangles or other unusually dense subgraphs). The copying model does slightly better in this respect.

[1]we explicitly allow here the case that a node receives more than one link from the newly arriving node.

6.2 Heavy-Tailed Distributions via Optimization

In the last section, we investigated the preferential attachment model for the generation of power law degree distributions. While "rich-get-richer" models may be considered close to the truth for WWW-like graphs, they do not seem appropriate for graphs such as the Internet, as there appears no natural reason why nodes would choose to attach to high-degree nodes (or be more likely to find out about them). Hence, this class of models is not very useful for explaining the power laws observed in the Internet at the AS level [149].

Fabrikant et al. [145] argue that the reason power laws evolve in the Internet is a heuristic optimization performed by the owners of machines. Indeed, optimization as a cause for power laws has been investigated before. Mandelbrot [273] and Zipf [396] argue that power laws in word lengths are caused by the "Principle of Least Effort": languages evolve so as to optimize the average information transmitted per character or phoneme. (Notice, however, that the same result about word lengths can be obtained by assuming that characters, including the space bar, are pressed completely randomly [286], as we see in Section 6.3 below.) Carlson and Doyle [77] extend the argument for file sizes and other parameters, and Fabrikant et al. [145] apply it to Internet-like graphs.

They posit the following graph growth model. A communication tree is built as nodes arrive uniformly at random in a unit square. Let O be the first node that arrives, and assume that it arrives at the center of the square. Each node i arriving subsequently connects to a node j that had arrived earlier. The issue is which node j should i connect to. [145] argues that nodes want to be central in the network, i.e., few hops from O. At the same time, they want to have low "last mile" cost for their connection to j. The tradeoff is accomplished by considering the objective function $d_{ij} + \beta h_j$ where d_{ij} is the Euclidean distance between nodes i and j, h_j is the number of hops from j to O in the communication tree, and β is a given constant. That is, node i connects to the node j minimizing $d_{ij} + \beta h_j$, and consequently has $h_i = h_j + 1$.

Depending on the value of the parameter β, the graph evolves in very different ways.

- If β is large (e.g., $\beta \geq \frac{1}{\sqrt{2}}$), then the hop count is more important than any possible distance (all distances to O are at most $\frac{1}{\sqrt{2}}$), so the graph will evolve to be a star with node O as the center. In particular, the degree distribution will not be heavy-tailed, since all nodes but one have degree 1.

- If $\beta = 0$, then the Euclidean distances become the only criterion for minimizing the objective function. Thus, nodes connect to their closest neighbors. We will analyze this process briefly.

  If node i has two neighbors j, j', such that $d_{ij}, d_{ij'} \geq r$, then we also have that $d_{jj'} \geq r$ (else j, j' would have been neighbors instead of both connecting to i). It follows that each neighbor j of i at distance r has a circle of influence with area at least $\Omega(r^2)$ around it, which does not contain any other neighbors of i.

  Now consider the $O(\log n)$ rings around i of the form $R_k := \{j \mid 2^{-(k+1)} < d_{i,j} \leq 2^{-k}\}, k = 0, \ldots, 3 \log n$. Note that with high probability, each node other than i lies inside one of the R_k. Within each R_k, node i can have at most a constant number of neighbors, because the area of each R_k is at most $O(2^{-2k})$, while each neighbor of i in R_k has an area of at least $\Omega(2^{-2(k+1)})$ in which no other neighbor can lie. Hence, there is at most a constant number of neighbors in each such ring, and a total of at most $O(\log n)$.

  Thus, all degrees are bounded by $O(\log n)$, and the distribution of degrees is not power law, or even heavy-tailed.

- Thus, we next consider the case when $\beta > 0$ is small enough.

Theorem 6.2 *There is some $\varepsilon > 0$ (depending on β), such that there are at least $\Omega(n^{\varepsilon/6})$ nodes of degree at least $n^{1-\varepsilon}$.*

Notice that while this proves that the distribution is heavy-tailed, it does not necessarily imply a power law. Indeed, Berger et al. [39] subsequently showed that the distribution is not power-law. It has a heavy tail, but does not exhibit all intermediate degrees to the corresponding extent.

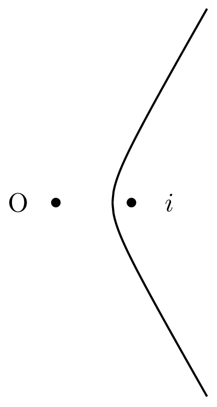
Proof. We first show that each neighbor of the first node O carves out a region of influence so that all nodes landing in the region must link to it. This gives those nodes high degrees. Then, we show that there will be enough such neighbors of O. Let S denote the set of all neighbors of O.

Consider a node i that connects to O, and has $d_{i,O} \in [\beta, 2\beta]$. Let $r_i = d_{i,O} - \beta$. At the time node i arrived, there are no other nodes from S within distance r_i of i, because i linked to the root (otherwise, it would connect to such a node). No node j arriving after i within distance $d_{i,j} \leq r_i/2$ connects to node O, because it would be cheaper to connect to i. So there are no other nodes of S within distance $r_i/2$ of i. Every node j that lands within $r_i/4$ of i links to i, because the cost to connect to node i is at most $\beta + r_i/4$, while the cost to connect to another node i' within distance $d_{i',i} \leq r_i/2$ of i is at least 2β, and the cost to connect to a node i' with distance $d_{i',i} > r_i/2$ of i is at least $\beta + d_{i',j} \geq \beta + r_i/4$ by the triangle inequality.

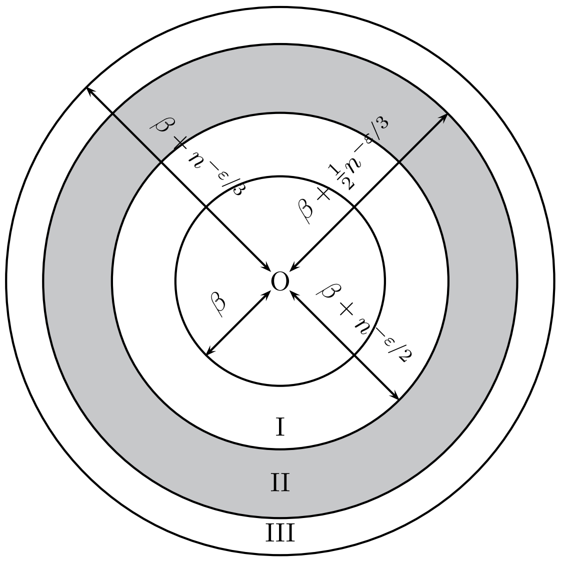
So node i has a region of influence with area $\Omega(r_i^2)$ around it in which all nodes link to it, and thus receives at least an r_i^2 fraction of all edges.[2]

So we know that any such node i has high degree (as long as r_i is large enough). We still need to show that there are enough such nodes i. Among others, we want to ensure that not too many nodes will link to i. To address this concern, consider a node j arriving after i. The line of indifference between connecting to i and connecting to O is given by $d_{j,O} = \beta + d_{j,i}$. This curve is a hyperbola (see Figure 6.1(a)), i.e., the nodes linking to i are a subset of the interior of that hyperbola.

Consider the following rings (Figure 6.1(b)) around O: Ring I has inner radius β and outer radius $\beta + n^{-\varepsilon/2}$, Ring II has inner radius $\beta + n^{-\varepsilon/2}$ and outer radius $\beta + \frac{1}{2}n^{-\varepsilon/3}$, and Ring III has inner radius $\beta + \frac{1}{2}n^{-\varepsilon/3}$ and outer radius $\beta + n^{-\varepsilon/3}$. ($\varepsilon$ is chosen small enough that $\beta + n^{-\varepsilon/3} \leq 2\beta$.) Note that the area of Ring II is a constant fraction of the total area of Rings I, II, and III.



(a)


$\beta + n^{-\varepsilon/3}$
$\beta + \frac{1}{2}n^{-\varepsilon/3}$
β
O
$\beta + n^{-\varepsilon/2}$
I
II
III

(b)

Figure 6.1: (a) The hyperbola $d(j, O) = d(j, i) + \beta$ (b) The three rings for the proof.

No node will ever link to a node $j \in S$ with $d_{j,O} < \beta$, because by triangle inequality, it would be cheaper to link to O directly. Nor will any node that falls in these rings ever connect to any node more than one hop from node O, because the cost would be at least 2β, so it would be cheaper to connect to node O directly. Similarly, nodes in Ring II will not connect to a node j outside the outer circle, because the cost is at least $\beta + \frac{1}{2}n^{-\varepsilon/3}$, while a direct connection to node O would cost at most $\beta + \frac{1}{2}n^{-\varepsilon/3}$.

[2]Note: to show that i will have degree $\Omega(r_i^2 n)$, we need to be sure that enough other nodes will arrive after i to give it high degree. We can solve this problem by only considering the degrees for nodes that are among the first $n/2$ to arrive. Then, at least $n/2$ nodes arrive subsequently, giving us the desired high degree for those nodes. We only lose a factor of 2.

In summary, each node i with $\beta + n^{-\varepsilon/2} < d_{i,O} < \beta + \frac{1}{2}n^{-\varepsilon/3}$ links either to O or to some node $j \in S$ with $\beta \le d_{j,O} \le \beta + n^{-\varepsilon/3}$. Each such node that links to O carves out a hyperbola, so that subsequent nodes that fall in the hyperbola do not link to node O.

If a new node i from Ring II arrives outside all existing hyperbolas of nodes j with $\beta \le d_{j,O} \le 2\beta$, then i links to node O. We want to find a lower bound on the number of hyperbolas defined by such i. Some analytic geometry shows that the largest angle θ of each hyperbola is $O\left(\sqrt{r_i/\beta}\right)$. Because $r_i < n^{-\varepsilon/3}$ in Ring II, and β is a constant, θ is $O\left(\sqrt{n^{-\varepsilon/3}}\right) = O(n^{-\varepsilon/6})$. So there is room for $\Omega(n^{\varepsilon/6})$ disjoint hyperbolas.

Each node i that carves out a hyperbola has degree at least $\Omega(nr_i^2)$. Because $r_i = \Omega(n^{-\varepsilon/2})$ in Ring II, the degree of these nodes is at least $\Omega(n \cdot (n^{-\varepsilon/2})^2) = \Omega(n^{1-\varepsilon})$. Among all the nodes arriving in any of the rings (and thus claiming hyperbolas), at most a constant fraction lie outside of Ring II, so at least $\Omega(n^{\varepsilon/6})$ nodes will be in Ring II, claim hyperbolas, and thus have degree at least $\Omega(n^{1-\varepsilon})$.

Notice that while we only talk about the expected number of such nodes (and their expected degree), standard occupancy bounds can be used to show concentration, i.e., that the actual outcome will be close to the expectation.

This completes the proof. ■

6.3 A Simple Model for Power Laws in Word Frequencies

While Mandelbrot [273] and Zipf [396] argue that power laws in word frequencies can be explained by optimization of transmitted content, Miller [286] proposes a much simpler model also resulting in power law distributions.

Miller posits a completely random typer (e.g., a monkey at a typewriter). The typewriter has $b > 1$ letter keys, each of which is hit with probability p/b, for some constant $p \in (0, 1)$. Word separators (such as newline or space) are hit with the remaining probability $q = 1 - p$.

In this model, the probability that any particular word of i letters is typed is $(p/b)^i \cdot q$, and all length-i words have the same probability. Thus, in this model, each length-i word is more likely than any word of length $(i + 1)$. Since there are b^i words of length i, they occupy the next b^i positions (in word frequency order) starting from position $k = \sum_{j<i} b^j = \frac{b^i-1}{b-1} = \Theta(b^{i-1})$. Thus, the words at positions $k \approx b^{i-1}, \ldots, b^i$ (roughly) each have probability $(p/b)^i \cdot q \approx p^i \cdot q \cdot k^{-1}$. Taking logs with base b, we have that the log of the probability is roughly $-\log_b k + i \log_b p + \log_b q$, which, since $i \approx \log_b k$, is $\log_b k \cdot (\log_b p - 1) + \log_b q$. We thus obtain a power law with exponent $\log_b p - 1$.

The model of "monkeys at typewriters" has been extended by Conrad and Mitzenmacher [107] to the case of non-equal probabilities for hitting different keys. They show — using a significantly more complex proof — that power laws in the word frequency distribution persist even in this case.

We can also interpret the same mathematical model differently. Suppose that we have a hierarchy of topics, organized as an (infinite) b-ary tree. Topics higher up in the tree are more general, while topics further down are more specific. We traverse this tree starting from the root; at each step, we stop with probability q, and otherwise choose a uniformly random child. Thus, the distribution of links to a particular topic is power law with exponent $\log_b p - 1$.

We can extend this to a simple model for topic-based linking in the WWW. Let the tree be truncated at a finite level now. For each of the topics (i.e., nodes of the tree), there are c pages covering that topic. When a page is created, it generates k outgoing links. Each outgoing link goes to a page on the same topic, or to a page on a more general topic, i.e., an ancestor in the tree. Say that a page links to another page h levels up with probability proportional to γ^h, for some constant γ. We calculate the in-degree distribution of a page p. Suppose that p is located h levels above the leaves. Thus, there are $b^\ell \cdot c$ pages in the descendants ℓ levels below p. Each gives rise to an expected $k/c \cdot \gamma^\ell$ links to p, so the expected number of links into p is $\sum_{\ell=0}^{h} b^\ell \cdot c \cdot \gamma^\ell \cdot k/c = k \cdot \sum_{\ell=0}^{k} (b\gamma)^\ell = \frac{k}{b\gamma-1}((b\gamma)^{h+1} - 1)$.

Thus, if $b\gamma < 1$, the nodes' degrees are nearly uniform, because $\frac{(b\gamma)^{h+1}-1}{b\gamma}$ is bounded on both sides by some constant. On the other hand, when $b\gamma > 1$, most of the contribution to any node comes from the leaves (since there are many of them, and they have high enough probability to link up). The nodes of degree at least $\frac{k}{b\gamma-1}((b\gamma)^{h+1}-1)$ are those at level h or above, which is a b^{-h} fraction of all nodes. Now, ignoring constant factors and additive terms, we can solve for h in $d=(b\gamma)^{h+1}$, giving us $h=\log_{b\gamma} d-1$. As a result, roughly $d^{\frac{-\log b}{\log b\gamma}}$ nodes will have degree d, i.e., we again observe a power law.

6.4 Further Reading

In considering merely a log-log plot, power laws are easily confused with lognormal distributions. A random variable X is lognormal if $\ln X$ has as normal (Gaussian) distribution. See the discussion in Mitzenmacher's survey [287] for more details. Lognormal distributions can for instance be obtained by processes where in each step t, each quantity X_j grows by a constant factor F_j^t (rather than a constant additive term). If the F_j^t are i.i.d., then in the limit, the X_j will be lognormal. Huberman and Adamic [210] show that combining such a multiplicative growth process with different arrival times (as we saw in Section 6.1 for the special case $\alpha=0$) also gives rise to power-law distributions. Similarly, Champernowne [82] shows that combining a multiplicative update with a hard lower bound on all sizes X_j leads to a power law.

The power-law nature of the Internet topology reported in [149] has been the subject of quite a lot of discussion. As noted by Chen et al. [89], the data set used for analysis missed a large fraction of physical connections, comprising essentially a union of BFS trees of the topology. Lakhina et al. [257] showed via experiments that for sampling based on shortest paths like the one in the data set analyzed by Faloutsos, Faloutsos, and Faloutsos, even $G(n,p)$ random graphs would exhibit a power law (though with exponent -1). This was proved by Clauset and Moore [97], and in a more general setting by Achlioptas et al. [3]. The upshot here is that even random *regular* graphs will exhibit a power law if the edges are sampled in a manner akin to the traceroute tool used for the data set of [149].

Instead of asking for "natural" models which predict the structure of known graphs with power law degree distributions accurately, we could instead start out from the assumption that the graph is uniformly random subject to its degree distribution, and investigate which properties follow merely from this degree distribution. Along these lines, Molloy and Reed [290, 291] analyze the point at which a giant component emerges in a random graph with given distribution, and how the component's size depends on the distribution. Using mean-field approximations and generating functions [385], Newman et al. [315] give an alternate derivation of the results of Molloy and Reed, and prove several other properties of random graphs with a given degree distribution.

While hard-wiring a desired degree sequence is a first step toward matching various properties of real-world observed networks, it is really only that. There are many other properties not nearly as well explained by such uniformly random graph models. One of the most prominent ones is "clustering": the fact that nodes sharing a neighbor are more likely to be connected themselves. Several recent papers have proposed simple random graph models resulting in clustering. The simplest one is that of *Random Intersection Graphs* [354, 226]. Here, we are given two types of nodes: individuals and groups. We generate a random bipartite graph between those two types of nodes. Then, we connect two individuals if they share at least one group. Thus, all individuals in the same group form a clique, and we have built in significant clustering. This model has been extended to account for different degree distributions as well (see, e.g., [309, 122]).

While this class of models can match degree distributions while building in clustering, there may be other parameters of a real-world network we would also like to match. One class of random graphs lets us match basically any features of a graph we care about. The model is called ERG (*Exponential Random Graphs*) or p^* graphs [161, 340, 379]. The high-level idea is to define several features (e.g., number of triangles, degrees, diameter, etc.), and posit that a graph G is generated at random with probability proportional to $\exp(\sum_{S\in G}\alpha_S)$, where S ranges over all features of interest, and $[S\in G]$ denotes that feature S is present in the graph G. (Notice that this approach is quite similar to the Markov Random Field analysis in Section 4.1.) If we have chosen the α_S parameters, we can sample random graphs from this distribution. A complementary problem is to find the parameter settings for α_S giving the maximum likelihood estimate of an *observed graph* G. Both problems appear to be computationally difficult in general, though there are heuristics for sampling which sometimes work well in practice.

One drawback of the basic preferential attachment model of Barabási et al. is that it produces only acyclic graphs. Hence, various papers have considered generalizations of this model. A fairly broad generalization is due to Aiello et al. [10]. In their class of models, a random number of nodes and edges can be added in each step, including edges between previous nodes. The attachment is again preferential, and nodes can arrive with initial indegree and outdegree weights. The advantages of this model claimed in [10] include the fact that it can model arbitrary power-law degree distributions on both indegrees and outdegrees, while also allowing for arbitrary density, and subsuming most past models. A fairly detailed overview of models up to that point in time, together with results on structural properties and implications of these models (again using mean-field approximations) is given by Albert and Barabási [13].

Another generalization was proposed by Leskovec et al. [266] and termed the *Forest Fire Model*. It was based on the observation that real-world networks often appear to become denser over time (i.e., the number of edges increases super-linearly in the number of nodes), and that their diameter often shrinks over time. Standard random graph models do not predict such behavior. In the Forest Fire Model, nodes arrive one at a time. When a node arrives, it chooses one *ambassador* w to link to. In addition, it selects $x \sim \text{Bin}(1/(1-p))$ of w's neighbors, links to them, and recurses on these neighbors with the same value x. p is a parameter of the model, and the model allows weighing forward and backward edges differently in determining a node's neighbors. Using simulations, Leskovec et al. show that this model matches the evolution of density and diameter in several observed real-world graphs.

While the mean-field approximations in Section 6.1 and in several of the papers cited here are not mathematically rigorous, a beautiful result by Wormald [386, 387] gives sufficient conditions under which the results of the differential equations give a precise approximation (up to lower-order terms) of the correct outcomes, with high probability. A perhaps more intuitive description of this result is given by Achlioptas [2].

Chapter 7

Small-World Graphs and Decentralized Search

In the chapters so far, we have explored the structure of information networks, and how additional information can be extracted by looking at it in graph-theoretic terms. In the process, we found that it is often useful to think about *generative models* for the network structure: if we can explain how the network formed, then we might be in a better position to label or cluster its parts, or observe other useful high-level structures.

In this and the remaining chapters, we will think of the network not so much as being the artifact to analyze and extract information from, but as a substrate that itself disseminates information. Thus, we will analyze the dynamics of information spread within social and computer networks, and try to understand what network structure would enable quick and efficient dissemination.

We begin by discussing the *small-world phenomenon*, both in its historical context and algorithmic implications. The expression refers to the anecdotal observation that any two people appear to be connected by a small chain of intermediate social acquaintances. Restated in graph-theory terms, this means that social networks (graphs whose nodes are individuals, and whose edges capture friendships, acquaintances, or other pertinent interactions) appear to have small diameter.

Stanley Milgram, in 1967, was the first to design an experiment to explore this hypothesis more scientifically [285][1]. His experiment, described in more detail in Section 7.1, led him to the conclusion that the average path length between two individuals in the social network of acquaintances within the United States is about six hops. The phenomenon has made its way into pop culture, including movies and plays, via the phrase "Six Degrees of Separation". Finding short paths in social networks is also the object of the game "Six Degrees of Kevin Bacon": the goal is to find a short path from any actor to the actor Kevin Bacon in the graph consisting of actors and their co-appearances in movies[2]. Mathematicians have a similar concept: the *Erdős number*. It captures the shortest path distance between a scientist and the famous Hungarian mathematician Paul Erdős, where edges represent co-authorships of papers[3].

7.1 Milgram's Experiment (1967)

In 1967, Stanley Milgram attempted to verify the small-world phenomenon quantitatively. His experimental setup was as follows: He selected random people from locations like Kansas or Nebraska, and asked them to forward a folder to a target in Cambridge, MA or Boston. The rules of forwarding were that the current holder of a letter could only mail it to someone they knew on a first-name basis. That person was to mail the

[1]Milgram's other famous experiment explored obedience to authority, and involved making participants believe that they were administering very dangerous electro-shocks to other participants [284].

[2]See `http://oracleofbacon.org`.

[3]See `http://www4.oakland.edu/enp/`.

folder on to another first-name acquaintance, etc. Returned tracer postcards tracked the progress of each such forwarding chain.

His first target person was the wife of a divinity student living in Cambridge, and starters were located in Wichita, Kansas. Milgram found that the very first folder reached her in just four days and took only two intermediate acquaintances. However, it appears that the completion rate of chains in this experiment was very low, so Milgram conducted a second study. Among other things, he made the folders much more impressive and official looking, which appears to have led to significantly higher participation. In the second study, the starters were located in Nebraska, and the target was a stockbroker from Sharon, MA, working in Boston. Milgram reported that "chains varied from two to ten intermediate acquaintances, with the median at five" [285, p. 65]. Subjects appeared to be able to reach the target with an average of six hops.

Graph-theoretically, Milgram was trying to find short paths in the social network of people in the US. Instead of using the "correct" approach of BFS, he had to rely on DFS, as the typical branching factor of several hundred social acquaintances would have resulted both in much work for all participating individuals, and an illegal chain letter using the US Postal System. So what Milgram found was really just an upper bound on the shortest paths.

As analyzed by Travers and Milgram [367], out of 296 chains, only 217 chains started, and 64 completed. The number of intermediate nodes varied between two and ten, with a median of 5 and a mean of 6. The frequencies that Milgram measured were monotonically increasing to chains of length 4, then dropped slightly for chains of length 5, and increased to the highest frequency for length 6, dropping again afterwards until length 10. Travers and Milgram argued that the drop in frequency at 5 is actually caused by the experimental results being a superposition of different populations.

The attrition rate (the frequency of not forwarding the letter) was about 25% at each step. Thus, it seems likely that the observed distribution is skewed to shorter chains: if a chain was naturally longer, then it has a higher probability of not finishing at all, and thus not being counted towards the average. White [384] gave a simple calculation correcting for this attrition. The argument is that if we observe a fraction of $\frac{n_j}{n}$ chains of length j, then the true fraction of such chains would have been roughly proportional to $(4/3)^j \frac{n_j}{n}$, as each chain of length j does not complete with probability $(3/4)^j$. Correcting for attrition in this way, White suggests that Milgram's data are more supportive of a median distance of 7–8 between individuals.

Additional interesting details about the Milgram experiments were that many chains had the same person as a last step: out of all completed chains, 25% went through the same last person, and 50% through one of three persons. This may be taken as an indication that society relies on "hub persons" for connectivity. Killworth and Bernard [239] analyzed the "rules" individuals used for deciding whom to forward letters to. In a separate experiment, they simply asked respondents (in West Virginia) to identify their "first hop" to a number of different targets. They found that geographic location and occupation were by far the dominant dimensions, with others used significantly less frequently.

Kleinfeld [247] revisits Milgram's experiment and finds that there were a number of loopholes in Milgram's experiment. Milgram's sampling technique to determine starting points for the chains was biased since his target was a stockbroker, and a number of his starters were stockholders, who would be more likely to know stockbrokers. He recruited people through advertisements for people who were well connected, and the experiment was generally more likely to attract economically better-off people, who tend to have more long-distance connections. In addition, out of the Kansas and Nebraska studies, only the Nebraska one, which was much more successful, was published.

7.2 Formalization, and Expander Graphs

If we want to reason about small-world networks analytically, we first need to come up with a good definition of the concept. A simple (and popular) definition is to consider a graph a small world if it has small (e.g., logarithmic) diameter. However, this ignores another characteristic of small worlds, namely that the graph exhibits a lot of clustering and regular structure, similar to lattices. Watts and Strogatz [382] propose using the notion of *clustering coefficient* C (see, e.g., [378]), defined as the number of triangles in the graph divided by the number of adjacent edge pairs. Thus, it can be interpreted as the probability that two nodes are connected, given that they share a mutual friend. For a complete graph, the clustering coefficient is 1, and for an empty graph, it is 0.

If we are interested merely in finding graphs with small diameter, our task is easy: the complete graph has diameter 1. However, the complete graph is certainly not a good model of social networks, as individuals tend to have a smaller number of contacts. So we are looking for graphs of small diameter and low degree.

One way in which we can achieve this goal is by virtue of *expanders*: A graph G is an α-expander (or α edge expander) if each vertex set $S \subseteq V$ with $|S| \leq n/2$ has at least $\alpha|S|$ edges leaving. Intuitively, this means that no two "large chunks" of the graph can be disconnected by removing few vertices or edges. Obviously, the larger α, the better an expander the graph is. Normally, people are particularly interested in families of expanders where α is constant, i.e., does not become smaller as the number of nodes n increases.

While the definition of expanders seems useful, we still have to find out whether they actually exist. It is pretty obvious that complete graphs are expanders (with $\alpha = n/2$), but again, we are looking for lower degrees. A binary tree is not a good expander, as the left half of the tree has $n/2$ nodes, but only one edge leaving. However, hypercubes are good expanders (with $\alpha = 1$), with degree $\log(n)$.

When we want to reduce the degree all the way down to constant, i.e., not increasing with n, the task becomes quite a bit more difficult. For a long time, no explicit constructions were known, and the first explicit constructions were based on Cayley graphs of rather difficult looking algebraic groups. More recently, Reingold et al. [337] presented a new way of constructing constant degree expanders via the zig-zag graph product. On the other hand, it is much easier to construct expanders randomly. For any $d \geq 3$, almost all d-regular graphs are actually expanders (notice that for $d = 2$, any d-regular graph is just a union of disjoint cycles, and thus certainly not an expander.)

It is not quite obvious how to generate a uniformly random d-regular graph. It is much easier to instead generate a d-regular *multigraph*, in which we also allow self-loops and parallel edges. For then, we can use the configuration model: each node has d edge ends sticking out, and we construct a uniformly random matching among those edge ends, by randomly picking two of the remaining edge ends and pairing them up until all edge ends are matched. For this model, whenever $d \geq 6$, a fairly straightforward analysis using standard tail bounds (Martingale or Chernoff bounds) shows that with high probability, the resulting graph is an expander. For $3 \leq d < 6$, the analysis gets a little messier, as tail bounds are not quite strong enough: one has to reason a bit more carefully about binomial coefficients and such.

7.2.1 Properties of expanders

Expander graphs, and the notion of expansion, are important for several reasons. From a network design perspective, expander graphs are resilient to node or edge failures, as a small number of failures will not disconnect large parts of the network.

Expansion also plays an important role in the analysis of random walks and their mixing time (and thus in the context of MCMC sampling). The reason is that a low expansion means that few edges connect two large components. Thus, a random walk starting in one of the components has small probability of crossing to the other component, and it will take a long time until the initial bias in the probability of being at any one node gets evened out.

For our purposes, the reason that expanders are interesting is that they have small diameter.

Lemma 7.1 *If G is an expander of maximum degree d, then G has diameter $O(\log n)$.*

Proof. Consider a BFS of G starting from any node. For each layer S of the BFS, at least $\alpha|S|$ edges are leaving the set of all nodes already reached, hence at least $\frac{\alpha}{d}|S|$ new nodes are hit, so long as $|S| \leq n/2$. Thus, at each layer, the total number of nodes in the BFS tree grows by a factor of at least $1 + \alpha/d$, and $\frac{n}{2}$ nodes are reached in

$$\log_{1+\alpha/d}(n/2) \quad = \quad O(\tfrac{\log n}{\log(1+\alpha/d)}) \quad = \quad O(\tfrac{\log n}{\alpha/d}) \quad = \quad O(\tfrac{d}{\alpha}\log n)$$

steps. Carrying out BFS from both the source s and target t, they each reach $\frac{n}{2}$ nodes. After one more step of BFS, they must therefore intersect at some node v, and the concatenation of the s-v and t-v paths gives an s-t path of length $O(\frac{d}{\alpha}\log n)$. ■

7.3 The Watts-Strogatz Model

In the past section, we began our quest for graphs that would model the empirically observed "small-world phenomenon": the fact that most (or all) pairs of people tend to be connected through short paths of mutual acquaintances. We observed that expander graphs tend to have small diameter, i.e., short paths, and that random d-regular graphs for $d \geq 3$ are with high probability expander graphs.

While this in principle will give us "small worlds", it is unsatisfactory in that real social networks do not at all look like random graphs. They exhibit a lot of "clustering" in the form of triangles or other short cycles. Indeed, two people sharing a common acquaintance are more likely to know each other as well. This observation has led to the definition of the *clustering coefficient* of node v as

$$\frac{|\{(u,u') \mid (u,u') \in E, (u,v) \in E, (u',v) \in E\}|}{\binom{d_v}{2}},$$

i.e., the fraction of actual triangles among pairs of neighbors of v. Random graphs have low clustering coefficient (roughly d/n for d-regular graphs), while real social or collaboration networks have much higher clustering.

In order to form a more realistic model giving both small diameter and high clustering coefficient, Watts and Strogatz [382] propose a simple model of small worlds. Here, we look at a slight modification (mostly for analytical simplicity), but will still refer to it as the Watts-Strogatz model. Start with a (structured) graph H (for instance, the 2-dimensional grid, or a ring), and add one or a few random edges from each node. (Alternately, "rewire" one or a few edges randomly.) The result will be that the edges of H will lead to high clustering coefficient, while the random edges lead to small diameter, as per our analysis of expander graphs. The resulting graph will still "resemble" H a lot. Intuitively, H is supposed to model the "traditional" social network, based perhaps on physical proximity or related employment, while the random edges model "random" friends.

An interesting additional outcome of Milgram's experiment [285] was not only the existence of short paths in social networks, but also the fact that individuals, who do not have a map of the graph, were able to actually *find* short paths. Their decisions were only based on local and qualitative information. Hence, an interesting question is what properties of a graph allow efficient decentralized routing based solely on local information. In addition to a better understanding of social networks, this question is also relevant for designing networks permitting simple routing.

To make this question more concrete, we can work with the Watts-Strogatz Model, and ask about the effect that different distributions of long-range links have on the ability to route with a decentralized algorithm. Watts and Strogatz considered the case of uniformly random links, i.e., each node links to one other uniformly randomly chosen one. We will show that in the case of the 2-dimensional grid of size $n \times n$, with this distribution, there is no local routing protocol reaching its destination in $o(n^{2/3})$ steps in expectation. Here, a *local protocol* is one in which a node only knows the history of the message in the network so far (including which nodes it has visited), and its own links.

For the proof, consider a ball B of radius $n^{2/3}$ around the destination node t, and assume that the source s of the message lies outside this ball. By the Principle of Deferred Decisions[4], we can assume that each node only generates its random outgoing link once it receives the message. Consider the first $\delta n^{2/3}$ steps of the protocol (for some constant $\delta \ll 1$). Because each node's outgoing link hits B with probability at most $O(\frac{(n^{2/3})^2}{n^2}) = O(n^{-2/3})$, only $O(\delta) \ll 1$ long-range links hit B in expectation, and with constant probability (by Markov's Inequality), none do. Thus, every step ending in B must have been a step on the grid. As a result, with constant probability, t was not reached in $\delta n^{2/3}$ steps, completing the proof. (Notice that the analysis is tight, in that in $O(n^{2/3})$ rounds, with constant probability, B is reached, and within at most another $O(n^{2/3})$ grid steps, the message makes it to t).

To study the effect of the distribution of long-range links on the ability to route in a decentralized manner, let Prob$[v \to w]$ denote probability that v connects to w via a long-range link. Assume that this probability

[4]The Principle of Deferred Decisions [300] states the intuitively obvious fact that we can defer coin flips or other random decisions until their outcomes are actually needed/referenced.

is a function of $d_{v,w}$, the distance between v and w. Intuitively, it seems that connections between distant individuals are less likely, so the probability should be monotone decreasing.

If $\text{Prob}[v \to w]$ is an inverse exponential function of $d_{v,w}$, then it decreases very rapidly, and long-range links are too unlikely. Hence, we use a polynomially decreasing function in $d_{v,w}$ [241], i.e., $\text{Prob}[v \to w] \sim (d_{v,w})^{-\alpha}$ for a constant $\alpha \geq 0$.

To understand this distribution well, we need to calculate its normalizing constant, i.e., the γ such that $\text{Prob}[v \to w] = \frac{1}{\gamma} d^{-\alpha}$. By noticing that for each d, there are $\Theta(d)$ nodes at distance d from any given node v, we can calculate

$$\sum_w d_{v,w}^{-\alpha} \quad \approx \quad \sum_{d=1}^n d d^{-\alpha} \quad = \quad \sum_{d=1}^n d^{1-\alpha} \quad = \quad \begin{cases} \Theta(n^{2-\alpha}) & \text{for } \alpha < 2 \\ \Theta(\log n) & \text{for } \alpha = 2 \\ \Theta(1) & \text{for } \alpha > 2 \end{cases}$$

We previously proved that for $\alpha = 0$, the case where the destination of each long-range link is a uniformly random node (independent of the distance between v and w), nodes cannot route efficiently based solely on local information. For very large α, we also expect local routing (or any routing, for that matter) to not work well, as long distance links will be exceedingly rare. Of the three cases (1) $\alpha = 2$ (2) $\alpha < 2$ and (3) $\alpha > 2$, it seems thus most promising that $\alpha = 2$ may work well for local routing. Indeed, we will show that it is the only exponent for which local routing can work in poly-logarithmic time.

The case $\alpha = 2$

Claim 7.2 *For $\alpha = 2$, we can route locally in $\Theta(\log^2(n))$ steps in expectation.*

Proof. The algorithm is the simple greedy rule of always forwarding the message to the neighbor closest to the destination.

Let s be the source node and t the destination node, at distance d from s. We will show that within an expected $O(\log n)$ steps, the distance of the message to t is halved. (Notice that the time here is independent of d.) In order to prove this, we let $B_{d/2}(t) = \{v \mid d_{v,t} \leq d/2\}$ denote all nodes in the ball of radius $d/2$ around t.

Because there are $\Theta(d^2)$ nodes in $B_{d/2}(t)$, and each is at distance at most $\Theta(d)$ from s, we can lower-bound the probability for s's long-range link to end in $B_{d/2}(t)$ as follows:

$$\frac{1}{\Theta(\log n)} \sum_{v \in B_{d/2}(t)} d_{s,v}^{-\alpha} \quad \geq \quad \Theta\left(\frac{1}{\log n} \cdot d^2 \cdot d^{-\alpha}\right) \quad = \quad \Theta\left(\frac{1}{\log n}\right)$$

Notice that this gives a notion of "uniformity of scales": the probability of halving the distance is the same independently of what the distance itself is. Similarly, if we think of circles around v of size 2^k for $k = 0, 1, \ldots$, then v has (approximately) equal probability of having its long-range link in any of the annuli between circles of radius 2^k and 2^{k+1}.

By our calculations above, any single long-distance edge reaches $B_{d/2}(t)$ with probability at least $\Theta\left(\frac{1}{\log n}\right)$. If not, the message is moved to another node, no further away, which again has at least the same probability of reaching $B_{d/2}(t)$, etc. Hence, the number of steps until a long-range edge will reach $B_{d/2}(t)$ is lower-bounded by a negative binomial random variable with parameter $\Theta\left(\frac{1}{\log n}\right)$. In particular, the expected number of steps to hit $B_{d/2}(t)$ is $\Theta(\log n)$, and the actual number is sharply concentrated around the expectation. As the distance of the message from t is halved every $\Theta(\log n)$ steps (independently of the current distance), t will be reached after $\Theta(\log n \log d) = O(\log^2 n)$ steps. ■

The case $0 \leq \alpha < 2$

We already saw that for $\alpha = 0$, local routing takes a polynomial number of steps. The problem was that links were too unstructured, and the chain was unlikely to encounter any node with long-range link into a (sufficiently small) polynomial-sized ball around the destination. Here, we will generalize the construction to show that the same problem occurs for all $\alpha < 2$.

Claim 7.3 *For $\alpha < 2$, local routing requires a polynomial number of steps in expectation.*

Proof. Let $\delta = \frac{2-\alpha}{3}$, and $B(t)$ be the ball of radius n^δ around t.

This time, we want to upper-bound the probability that a given node has a long-range link into $B(t)$. Even when two nodes are very close together, say $d = 1$, the probability of a long-range link between them is at most $\frac{1}{\gamma} = n^{\alpha-2} = n^{-3\delta}$. As there are only $\Theta\left(n^{2\delta}\right)$ nodes in $B(t)$, the probability for a given node to have a long-range link into $B(t)$ is at most $\Theta\left(n^{-\delta}\right)$. Thus, it takes n^δ steps in expectation to encounter a long-range link into $B(t)$. On the other hand, if no such long-range link is encountered, then the chain takes at least n^δ small steps (namely, all the last n^δ steps). In either case, the total number of steps is $\Omega\left(n^\delta\right)$. ■

The case $\alpha > 2$

For small α, the problem is that, while short paths exist, they cannot be found with local information alone, as the random links are "too random". For large α, the problem is that there are not many long-range links — the argument below needs to be strengthened only slightly to show that the diameter of the graph is actually polynomial in n.

Claim 7.4 *For $\alpha > 2$, the expected number of steps required to route from s to t is polynomial in n.*

Proof. Following our intuition that long-distance links are unlikely, we first calculate the probability that a link is longer than some given number m. Using again that there are $\Theta(d)$ nodes at distance d from a given node, this is at most

$$\Theta\left(\sum_{d=m}^{\infty} d d^{-\alpha}\right) = \Theta\left(\sum_{d=m}^{\infty} d^{1-\alpha}\right) = \Theta\left(m^{2-\alpha}\right).$$

Hence, the probability that a link's length is at least, say, $n^{1/3}$, is at most $\Theta(n^{\frac{2-\alpha}{3}})$. So if we only take $n^{\frac{\alpha-2}{3\alpha}}$ steps, the probability of having any of them encounter a long-range link longer than $n^{1/3}$ is polynomially small ($O(n^{\frac{2-\alpha}{3}(1-1/\alpha)})$). But if none of them encounter any longer links, then the total distance covered is at most $n^{1/3} \cdot n^{\frac{\alpha-2}{3\alpha}} = n^{\frac{\alpha-2}{\alpha}} = o(n)$, so the destination cannot be reached. (Notice that instead of $n^{1/3}$ and $n^{\frac{\alpha-2}{3\alpha}}$, we could have chosen other values $\beta > 0$ and $\gamma > 0$ such that the first n^β steps don't see a link of length greater than n^γ, so long as $\beta + \gamma < 1$.) ■

Based on simulations, it seems that the behavior is actually worse for $\alpha > 2$ than for $\alpha < 2$.

In the analysis, it is important to notice that while, for $n \to \infty$, any poly-logarithmic function is exponentially smaller than any polynomial one, this may not be so for finite, even fairly large, n. Indeed, it has been verified experimentally that the "correct" exponent α for finite n is $\alpha = 2 - f(n)$, where $f(n) \to 0$ as $n \to \infty$. Determining the exact nature of $f(n)$ is still open. This may also help in explaining the apparent gap between the short paths observed by Milgram, and the apparently much longer paths guaranteed by our proof. In addition, our analysis only used one long-range link per node. If nodes have many more long-range links, the predicted path lengths will be smaller.

While our analysis was done for a 2-dimensional grid, we did not use many properties of it. In fact, the only property we used was that there were $\Theta(d)$ nodes at distance d from a given one. The analysis extends easily to r-dimensional grids; the unique exponent for which local routing can be accomplished is then $\alpha = r$.

It also extends to hierarchical tree structures. For professions or other interests, a grid is perhaps not the right type of model. Instead, we may consider all professions to form a hierarchy ("scientific" vs. "arts" vs. "finance" etc., further subdivided into different sciences, subcategories, etc.). Each person is located at a leaf of this hierarchy, and generates $\Omega(\log^2 n)$ links to other people. The probability of linking to a given node decreases in the distance between the two nodes in the tree. It can be shown that for an appropriate choice of distribution, we still obtain local routing, and the distribution is closely related to the one studied above, in that it satisfies "uniformity over scales".

Kleinberg [242] proves a more general result for a certain class of set systems which includes metric spaces that "grow uniformly". Whether a similar result holds for arbitrary metric spaces is currently open. Watts et al. [381] suggest a model wherein multiple metric spaces (or, in their case, hierarchies) combine to create the link structure. The idea is that links could be caused by proximity in a physical space (like the grid, above), or in the space of research topics, or others. Thus, the distance between two individuals is their *minimum* distance over all of the hierarchies, and individuals know about the hierarchy structure in their routing decisions. (See also the discussion by Killworth and Bernard [239].) Unfortunately, the result is not a metric, and it is not clear whether results can be proved formally about this model; Watts et al. based their investigation on simulations.

7.4 Further Reading

Since the original work by Kleinberg [241] on the algorithmic models, many subsequent papers have studied generalizations of the models in different respects. Excellent surveys of both the results described above and much of the follow-up work are given by Kleinberg [243] and Fraigniaud [160].

One of the interesting directions is how much extra knowledge the members of the network need in order to find (nearly) shortest paths. While we showed above that the greedy algorithms finds paths of length $O(\log^2 n)$, these are not necessarily shortest for every pair of nodes. Lebhar and Schabanel [260] analyze the effects of performing some depth-first exploration of the neighborhood of the current message holder before actually forwarding the message. They show that this results in much shorter paths of length $O(\log n (\log\log n)^2)$ (for our model of one random link). Manku, Naor, and Wieder [275] show that when the average degree of nodes is $O(\log n)$, by generating each long-range link at distance d *independently* with probability $1/d$, a mere 1-step lookahead leads to paths of length $O(\log n / \log\log n)$.

Another popular direction for extensions is to change the underlying graph structure from a grid, or the precise long-range link distribution. One of the most comprehensive studies of this topic is by Nguyen and Martel [318], who analyze the diameter of the above model in D dimensions, and show that short paths exist for any exponent $\alpha < 2D$, while the diameter is polynomial for $\alpha > 2D$. (The case $\alpha = 2D$ is currently open.) They also relate these diameter bounds and greedy routing more generally to expansion properties.

The Watts-Strogatz model for small-world networks can be extended to settings beyond social networks. For example, Menczer [281] posits a model for generation of web content and links wherein the metric is obtained by textual similarity of web pages (using standard bag-of-words models), and links follow a power-law distribution in this metric. He compares the model to real-world data, and infers that it implies searchability of the Web by greedy algorithms.

Chapter 8

Diffusion of Information and Influence in Social Networks

In the first few chapters, we were studying networks mostly as "enhancing" the information stored in them. In the previous chapter, for the first time, we considered networks as active entities forwarding information between their nodes. We continue the study of such dynamics in this and the next chapter, focusing on the interactions between the structure of a network and its role in disseminating information, diseases, data, etc. As some examples of the types of "things" that spread through a network, consider the following:

Biological Networks : Diseases and mutations. Dominant mutations result in more and more affected entities from generation to generation, until almost all of the species is affected. Diseases spread through contact, while mutations spread through natural selection in reaction to evolutionary constraints.

Technological Networks : Viruses and worms, but also power failures or car traffic. Power failures in a power grid increase the load on other generating stations causing them to also break down, etc [24, 25, 380]. Congestion on freeways leads to similar behavior, spreading to nearby freeways or roads by virtue of drivers' evasive strategies. Viruses or worms directly use human or technological connections to spread from one host to another.

Social Networks : Rumors or group behavior such as riots. A rumor spreads from person to person until a large subset of people are aware of it. Similarly, peer pressure or "safety in numbers" entices people to participate in behavior shared by others, such as rioting [347, 189].

Strategies : as a special case of social networks, some trends, such as cell phone usage or car sizes, spreads as a result of "strategy optimization" [52, 137, 294, 390]. For example, to not be at a disadvantage on the road with bigger cars like SUVs around, people themselves need bigger cars. Similarly, once more others have adopted a new innovation, such as a new media format, it may become beneficial or necessary to adopt it also [211].

The common thread of all of these examples is that a new behavior or piece of information starts out with one or a few individuals, and gradually propagates through the network to (sometimes) reach a much larger number eventually.

In order to analyze these types of dynamics, we need to first define a model for the effect of the network on individuals' behavior. Here, we can draw on several decades worth of work in sociology, biology, and economics.

8.1 The Bass Model for Innovation Diffusion

While there had been many empirical studies of the diffusion of technological innovations in society (see, e.g., [105, 341], or [72] for a study of the relationship between innovation diffusion and strength of ties [188]), perhaps the first descriptive mathematical model is due to Bass [36]. It is similar to standard models for epidemic diseases [27, 125]. The presentation here is inspired by that of Jackson [213].

The population is assumed to be infinite and mixed, i.e., each individual is equally likely to interact with each other individual. Two different types of "pressure" affect individuals: mass communication such as advertising, and word-of-mouth interactions with others who have adopted the innovation already.

Specifically, assume that an individual has not yet adopted the innovation. In each "time step", he will be convinced by the "mass marketing" with probability p. In addition, he may meet another individual who has already adopted the innovation. In that case, he will adopt the innovation with probability q. (Let's assume that $q = (1-p)q'$, i.e., q accounts only for the case when the individual has not adopted the product due to mass marketing.) If at time t, an $N(t)$ fraction of the population have already adopted the innovation, then the meeting probability is $N(t)$ for any individual. Thus, the individual's probability of adopting is $p + q \cdot N(t)$. Since the fraction of individuals for which this calculation applies is $1 - N(t)$, we obtain that the increase in the fraction of adopting individuals at time t is $(1 - N(t)) \cdot (p + q \cdot N(t))$. This leads to the differential equation

$$\frac{dN(t)}{dt} = (1 - N(t)) \cdot (p + q \cdot N(t)).$$

This differential equation can then be solved, and the effects of varying p and q (as well as other assumptions) can be studied. Following the initial work of Bass [36], there has been a long line of extensions and variation. See, for instance, [271, 369].

8.2 Schelling's model for segregation

The Bass model assumes a homogeneous population, and furthermore assumes that each individual is equally likely to meet each other individual. Thus, it does not take any network structure into account. One of the first models to explicitly consider network structure in such behavior at a population scale was Schelling's model for segregation [347]. Schelling was motivated by the question: why is it that most neighborhoods are very uniform (racially, and in other respects), even though most people profess that they would prefer to live in a diverse community?

Schelling proposed the following model: Assume that roughly $\frac{n^2}{2}$ individuals live on an $n \times n$ grid. Each node wants to make sure to not be completely isolated (the odd person in a community): formally, if less than an ε fraction of v's neighbors (in a small ball around v) are of the same color as v, then v is unhappy and moves to a spot where it is not unhappy (say, the closest, or a random, such spot).

What Schelling observed was that even with a relatively small value of $\varepsilon \approx \frac{1}{3}$, neighborhoods end up mostly segregated: when the process quiesces, about $\frac{4}{5}$ of each node's neighbors are of the same color as the node itself. While this result has been verified experimentally, it appears that no result formally proves this observation as of yet. It is also not known how the result relates with the topology of the neighborhood graph.

8.3 Granovetter's threshold model

Schelling's model addresses one shortcoming of the Bass model: the assumption of a uniformly mixed population. However, it still assumes that all individuals act the same. Granovetter [189], motivated by the study of outbreaks of riots or other collective behavior, notices that cities or neighborhoods with very similar statistical demographics often tend to exhibit very different overall behavior of a riot or similar event. Thus, he concludes that an accurate description of group behavior is impossible in terms of merely high-level statistical data, and needs to take individuals' tendencies into account.

[189] therefore proposes a threshold model which has formed the basis of much subsequent work. The simplest version can be described as follows: Each individual (node) has a threshold t_v. If t_v other nodes are already active (have adopted the behavior, such as starting to riot, or using a cell phone), then v joins (becomes active).

As an example, consider the case of 100 people present in a city square in a tense environment, where the threshold of node v_i is i ($t_{v_0} = 0, t_{v_1} = 1, \ldots, t_{v_{99}} = 99$). In this case, v_0 becomes active first, followed by v_1, etc., until the entire population ends up active. On the other hand, if we make a small change and set $t_{v_1} = 2$, only node 0, and no other node, becomes active, even though 99 of the 100 nodes have the same thresholds as before. While this example may be a bit contrived, it clearly shows that the outcome of such a process can change dramatically even when the setup is almost identical.

One of the first questions we may wish to answer about this model is: given the nodes' thresholds, can we predict how many nodes will be active in the end. We can define $F(x) = |\{v \mid t_v < x\}|$. Then, the number of active nodes is the smallest x with $F(x) \leq x$. Figure 8.1 shows the plot of $F(x)$ vs. x. The vertical lines denote the number of nodes active after $0, 1, \ldots$ time steps. The number of nodes that will be active in the next time step is then $F(F(x))$, which we can determine pictorially by moving horizontally to the diagonal, and then vertically to the next intersection with $F(x)$. The process then quiesces at the first point for which the function F crosses the diagonal, as that is a fixed point.

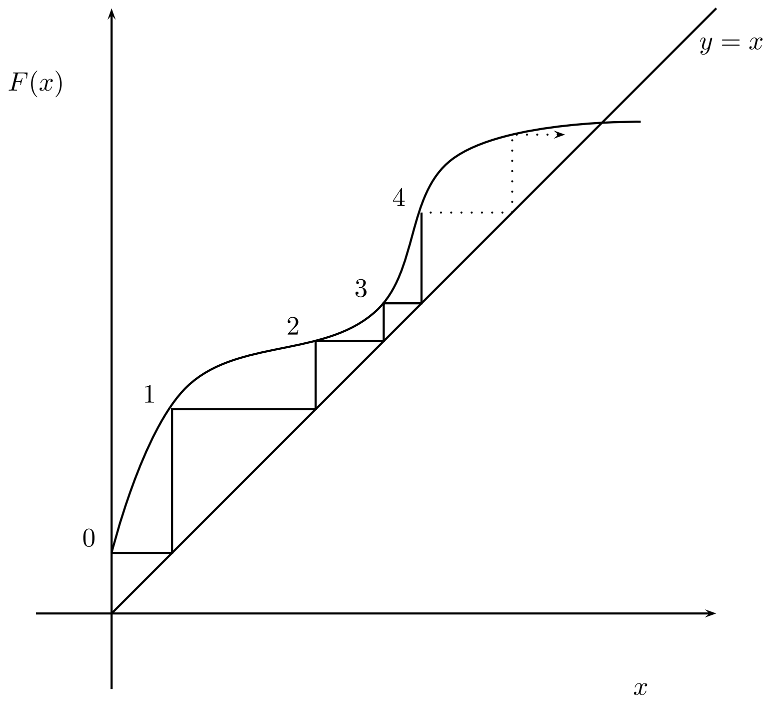


Figure 8.1: The threshold process and its dependency on $F(x)$

The threshold values we used in our introductory example were clearly somewhat contrived. Instead, we may wonder what happens when the values are drawn from a more "reasonable" distribution. For instance, we may assume that the thresholds are distributed as Gaussians, with mean, say, $\frac{n}{4}$ and variance σ^2. We could then investigate what is the result of varying σ^2, i.e., making the thresholds more or less sharply concentrated.

For small σ, almost no nodes end up active, because almost no thresholds will be 0 or close to 0. As σ is increased, there is a phase transition from almost no one active to almost everyone active. A further increase of σ leads to a slight drop, steadying around $\frac{n}{2}$.

Granovetter's threshold model is clearly very simple. We could make it more interesting and realistic in several ways:

Graphs : We assumed that all nodes affect all other nodes. In a large social network, this is certainly not true. We may consider a graph in which nodes can only affect their neighbors.

Weights : The model of graphs can be further generalized. In our model so far, each node is influenced equally by all of its neighbors. Instead, each edge in the graph could have a weight associated with it, representing how much influence one endpoint has over the other. (Edges may be directed, and have different weights in both directions.)

Probabilistic activations : In real world scenarios, some nodes may fail in activating their neighbors. This can be modeled using probabilities on activations. The resulting model may be somewhat different from Granovetter's. (A simple model is described by Goldenberg et al. [178, 177].)

Non-monotonic behavior : In many scenarios, a large rate of "participation" may discourage nodes from being active. For instance, a new fashion trend, initially spread by copying behavior, may become uninteresting to the trend-setters if many others are following it. As a result, in addition to the activation threshold $t_{1,v}$, we would have a deactivation threshold $t_{2,v} > t_{1,v}$: if more than $t_{2,v}$ nodes are active, then v becomes inactive again. In its most general version, this model subsumes cellular automata and the game of life (and hence, can simulate computation).

Deriving thresholds : So far, we assumed that the thresholds are known. Actually obtaining or estimating these data is an interesting question, and may be partially amenable to data mining techniques. (In the absence of more information, we may assume that thresholds are random, or all identical, but this is only a very crude approximation.) One interesting way to derive thresholds in the strategic settings described earlier is to cast them in terms of the underlying game played by the nodes, e.g., weighing the costs and benefits of a larger car depending on the number of other people having large vs. small cars.

Starting or preventing cascades : Understanding a model is usually only the first step towards acting on it. In the above scenarios, we may be interested in containing the spread of an epidemic, computer virus, or power failure, or promote the adoption of a product.

8.4 The Contagion model

In investigating the problem of starting or preventing cascades, a first question we may want to answer is how many nodes it takes to infect the entire graph. This leads us to a question considered by Morris [294]. His model is as follows: the graph G is assumed to be infinite, but each vertex has finite degree. All vertices have the same threshold $p \in [0,1]$: here, this means that they become active if at least a p-fraction of their neighbors is active. (Notice that the number of neighbors may be different for different nodes, but the fraction is the same).

These thresholds can be derived through a *coordination game* as follows [137]: Suppose that we have the following payoff matrix with $q \geq \frac{1}{2}$

	inactive	active
inactive	$1-q, 1-q$	a, b
active	b, a	q, q

meaning that if both players are inactive, they each get a payoff of $1-q$ (for instance, by communicating with some old technology). If both are active, they each get a payoff of $q \geq 1-q$, having switched to the new technology. When player 1 is using the old technology, and player 2 the new technology, their respective payoffs are a and b; usually, we would assume that $a < 1-q$ and $b < q$, since the players may be less able to communicate or share technology. (An important special case is $a = b = 0$, meaning that the players cannot communicate using different technology or languages.) Then, a node v with d neighbors, of whom r are active, will prefer to be active if and only if $rq + (d-r)b > ra + (d-r)(1-q)$, or $r > \frac{1-q-b}{1-(a+b)} \cdot d$. Thus, we have derived $\frac{1-q-b}{1-(a+b)}$ as an activation threshold.

A set $X \subset V$ is called *contagious*, if starting from X, all nodes are activated eventually. The *contagion threshold* $t(G)$ is the supremum of all p such that there exists a finite contagious set for threshold p.

In looking for obstacles to a set X being contagious, we notice that each node that is not infected must have at least a $1-p$ fraction of its edges to other uninfected nodes. Hence, if X^* denotes the set of all nodes reached from X, then $V \setminus X^*$ is a (strong) $(1-p)$-community in the sense studied in Section 3.2.

We define the community threshold $c(G)$ to be the infimum over all α such that every cofinite[1] set of nodes contains an infinite $(1-\alpha)$-community. Our main theorem is the following, verifying that communities are indeed the only obstacles to epidemics.

Theorem 8.1 *$c(G) = t(G)$.*

Proof. The normal way to prove an equality is to prove both inequalities $c(G) \leq t(G)$ and $c(G) \geq t(G)$. As both quantities are defined in terms of infimum/supremum, this direct approach will not work as easily. Instead, we will show for each p that $c(G) < p$ if and only if $t(G) < p$.

For any node set $S \subseteq V(G)$ and node threshold $p \in [0, 1]$, we let $f_p(S)$ be the set of nodes active after one step if exactly the set S is active previously, and all nodes have threshold p. Define $f_p^k(S) := f_p(f_p^{k-1}(S))$ inductively to be the set of nodes active after k steps (with $f_p^0(S) := S$, of course).

1. Assume that $c(G) < p$. By definition of the community threshold, this means that every cofinite set S contains an infinite $(1-p)$-community.

   Let X be an arbitrary finite set. Then, $V \setminus X$ is co-finite, and thus contains an infinite $(1-p)$-community C. Because $C \cap X = \emptyset$, induction on the number of steps k shows that no node from C is ever active. Hence, X cannot be contagious, and because X was an arbitrary finite set, we have proved that $t(G) < p$.

2. Assume that $t(G) < p$. Let S be a co-finite set, so $V \setminus S$ is finite. Therefore, by assumption, it is not contagious for parameter p. Let $Y := \bigcup_{k \geq 0} f_p^k(V \setminus S)$. Then, $\overline{Y}$ is a $(1-p)$-community, as each node of $\overline{Y}$ must have strictly less than a p fraction of its edges to nodes from Y. Moreover, $\overline{Y}$ is infinite, for otherwise, $\overline{Y} \cup \overline{S}$ would be a finite contagious set, a contradiction. Hence, every co-finite set S contains such an infinite $(1-p)$-community $\overline{Y}$.

■

Next, we would like to know how large contagion thresholds can possibly be supported by any graphs. We will show that $\frac{1}{2}$ is the limit:

Theorem 8.2 *There is no graph G with $t(G) > \frac{1}{2}$.*

Proof. For a set S, let $\phi(S) := |\delta(S)|$ be the number of edges leaving S. For $i \geq 0$, let S_i denote the set of infected nodes after i rounds. Observe that if at any point $S_i = S_{i+1}$, then clearly the infection process has terminated, as no new nodes can ever be activated.

Let $p > \frac{1}{2}$ be arbitrary, and S_0 a finite set starting out infected. Consider the sequence $\phi(S_i)$, $i \geq 0$, the number of edges leaving the infected sets in each iteration. When node v moves from $\overline{S_i}$ to S_{i+1} (i.e., v gets infected in the i^{th} step), node v has strictly more edges into S_i than into $\overline{S_i}$, because $p > \frac{1}{2}$. In the $(i+1)^{\text{st}}$ iteration, all the edges from v to S_i are not cut any more, whereas those from v to $\overline{S_{i+1}}$ are cut. Summing over all v, ϕ strictly decreases in each iteration in which nodes become infected. Because S_0 is a finite set, and each node has finite degree, $\phi(S_0)$ too must be finite. Therefore, ϕ can only decrease a finite number of times, and $S_k = S_{k+1}$ for some k. As only finitely many nodes become infected each round, only finitely many nodes will become infected eventually. This holds for any finite set S_0, and any $p > \frac{1}{2}$; therefore, the contagion threshold $t(G)$ cannot exceed $\frac{1}{2}$. ■

[1]A cofinite set is a set whose complement is finite.

In view of the above fact, a natural question to ask is which graphs have $t(G)$ close to $\frac{1}{2}$. We begin with some definitions.

A *labeling* of a graph G is a bijection $\lambda : \mathbb{N} \to V$. A labeling λ for G is *p-inductive* with parameter $k_0 \in \mathbb{N}$ if for all $k \geq k_0$, each node $\lambda(k)$ has at least a p-fraction of edges to $\lambda(0), \ldots, \lambda(k-1)$. We define $\ell(G)$ to be the supremum over all p such that G has a p-inductive labeling. This new measure is again equal to the contagion and community thresholds.

Theorem 8.3 *$\ell(G) = t(G)$.*

Proof. Start with nodes $\lambda(0), \ldots, \lambda(k_0)$ infected. Then, by definition of p-inductiveness, all subsequent nodes will become infected at threshold p. Similarly, the order in which nodes become infected can be used as a labeling, which is easily seen to be p-inductive. ∎

A *BFS-labeling* is a labeling λ for graph G consistent with the BFS traversal of G from some finite node set X. It is *δ-smooth* for parameters p and k_0 if all nodes $\lambda(k)$ for $k \geq k_0$ have between a $(p-\delta)$-fraction and a p-fraction of edges to $\lambda(0), \ldots, \lambda(k-1)$. Also, recall that a graph G has *subexponential growth* if for all $c > 1$ and all finite sets X, the ball sizes are $|B_r(X)| = o(c^r)$. (Where $B_r(X) := \{u \mid d(u, X) \leq r\}$ is the set of all nodes at distance at most r from X.)

Theorem 8.4 *If a graph G has subexponential growth and a δ-smooth BFS-labeling, then $t(G) \geq \frac{1}{2} - \delta$*

Notice that this is not an "if and only if" statement. For instance, consider the grid with sufficiently large neighborhood circles, and add one random edge per node. For purposes of infection, this graph acts very much like the grid (as most edges are grid edges). However, as we discussed in Section 7.3, this graph has exponential growth. Indeed, one might consider a definition of "small world" graphs to have both exponential growth and $t(G)$ close to $\frac{1}{2}$.

Proof. We will show that the labeling must have $p = \frac{1}{2}$. Then, because the labeling is $\frac{1}{2} - \delta$ inductive, Theorem 8.3 implies the result. We prove the statement by contraposition: if $p < \frac{1}{2}$, then the graph must have exponential growth. Consider a BFS consistent with λ. For each v, let $f(v)$ be the number of edges between v and nodes u with lower labels, and $g(v)$ the number of edges between v and nodes u with higher labels.

By definition, $f(v) \leq p(f(v) + g(v))$, so $g(v) \geq \frac{1-p}{p} \cdot f(v)$. Let $L_r = \{u \mid d(X, v) = r\}$ be the r^{th} layer of the BFS from X. We denote the edges within L_r by I_r, and the edges from L_{r-1} to L_r by F_r. Then, $\sum_{v \in L_r} f(v) = |I_r| + |F_r|$ (counting each edge towards the value of the higher endpoint), and $\sum_{v \in L_r} g(v) = |I_r| + |F_{r+1}|$ (counting edges for lower endpoints). Combining this with the inequality between $f(v)$ and $g(v)$, we obtain that $|I_r| + |F_{r+1}| \geq \frac{1-p}{p} \cdot (|I_r| + |F_r|)$, which we can rearrange to obtain

$$|F_{r+1}| \quad \geq \quad \tfrac{1-2p}{p} \cdot |I_r| + \tfrac{1-p}{p} \cdot |F_r| \quad \geq \tfrac{1-p}{p} \cdot |F_r|,$$

because $\frac{1-2p}{p} \geq 0$. Therefore, the sizes of F_r grow exponentially, and since the nodes have finite degrees, the number of vertices in the r^{th} layer must also grow exponentially. ∎

8.5 A non-monotone infection model

In the previous section, we began an investigation of models for the spread of infections on graphs. We assumed the following notion of *monotonicity*: once a node becomes infected, it stays infected. For certain types of diseases or behaviors, this model may be very accurate. For others, nodes will reevaluate their choice in every time step, based on the choices of their neighbors. A natural modification of the previous model posits that if in the previous time step, at least a p fraction of a node's neighbors are infected, then the node will be infected in the next step, and otherwise, it will not.

In thinking about this new definition, a first question is whether any infinite graphs can be infected starting from a finite set of infected nodes. It is pretty easy to see that the answer is "Yes": if we start with two adjacent infected nodes on the infinite line, and $p < \frac{1}{2}$, then the entire line will eventually become infected. In fact, for infinite graphs, Morris [294] showed the following lemma:

Lemma 8.5 *If the (infinite) graph G has a (finite) contagious set with threshold p in the monotone model, then it also has a finite contagious set with the same threshold in the non-monotone case. The starting set in the non-monotone case will be the union of the starting set of the monotone case and its neighborhood.*

Notice that our example of an infinite graph with finite infectious set crucially used that $p < \frac{1}{2}$. It seems much more difficult to come up with examples having $p \geq \frac{1}{2}$. In fact, is not even obvious how to construct arbitrarily large finite graphs that can be infected by a small set of nodes.

For the monotone case, the question with finite graphs is easy: take a star graph with threshold 1/2, and start with just the center node infected. However, in the non-monotone case, this graph will end up oscillating with period 2, between the center node infected, and all leaf nodes infected. In fact, this is far from accidental: the following is a corollary of slightly more general results of Goles and Olivos [179] and Poljak and Sura [328].

Theorem 8.6 ([179],[328]) *In the non-monotone case for a finite graph, the infection either converges, or oscillates with a period of 2.*

This result has been extended to infinite graphs by Moran [293] and Ginosar and Holzman [174]. They show that it is sufficient (and necessary) that the infinite graph have universally bounded degrees and sub-exponential growth in the sense defined above. Under these conditions, Moran [293] showed that any state that is part of a cycle of state is part of a cycle of length 2. Ginosar and Holzman [174] showed that each node in isolation will eventually enter a cycle of length 2. However, even for the infinite path in which all odd nodes and 0 have an initial state of 1, no global periodic state will ever be reached.

Returning to the question of the existence of small infectious sets for arbitrarily large graphs, Berger [38] answered it in the affirmative for $p = \frac{1}{2}$.

Theorem 8.7 (Berger [38]) *There are arbitrarily large graphs that can be infected (in the non-monotone case) with $p = \frac{1}{2}$ and a starting set S with $|S| \leq 18$.*

At this point, it is open whether such finite infectious sets exist for $p > \frac{1}{2}$. However, we have the following partial result, showing that arbitrarily large p will not work.

Lemma 8.8 *It is impossible to infect arbitrarily large graphs if $p > 3/4$.*

Proof. We denote by A_t the set of nodes active in round t, and write $I_t := A_t \cap A_{t-1}$ for the set of nodes active both in rounds t and $t-1$. Then, let $S_t := \bigcup_{t' \leq t} I_{t'}$ be the set of nodes which were active in any two consecutive rounds up to and including round t. Finally, let $\delta(S_t)$ be the set of edges leaving the set S_t, and $\sigma_t := |S_t| + |\delta(S_t)|$.

We will show that if we start with c nodes active, then $\sigma_t = O(c^2)$ for all t. For this purpose, we show that $\sigma_1 = O(c^2)$, and that σ is a non-increasing function of t.

For the case $t = 1$, notice that $S_1 = A_0 \cap A_1 \subset A_0$. Thus, $|S_1| \leq c$. Since each $v \in S_1$ is active at time $t = 1$, its degree can be at most $\frac{4}{3}c$ (for at least $\frac{3}{4}$ of its neighbors must be in A_0). Thus, the sum of all degrees in S_1 is at most $c \cdot \frac{4}{3}c$, which is $O(c^2)$.

To show that $\sigma_{t+1} \leq \sigma_t$, consider any node $v \in S_{t+1} \setminus S_t$. Due to the addition of v, the $|S_{t+1}|$ term of σ_{t+1} increases by one. On the other hand, because $v \in A_{t+1} \cap A_t$, more than $\frac{3}{4}$ of v's neighbors were active at time t, and also at time $t-1$. Thus, more than half of v's neighbors are active *both* at times t and $t-1$, i.e., they are in $A_t \cap A_{t-1} \subseteq S_t$. For each of those neighbors $u \in I_t$, there was an edge $(u, v) \in \delta(S_t)$, whereas $(u, v) \notin \delta(S_{t+1})$ any more. For each neighbor $u \notin I_t$, we introduced at most one new edge into $\delta(S_{t+1})$ (or none, if u was also added to S_{t+1}). Since there are strictly fewer neighbors not in I_t than neighbors in I_t, $|\delta(S_t)|$ decreases by at least 1 for each node $v \in S_{t+1} \setminus S_t$. Therefore, σ cannot increase overall. ■

8.6 Causing a large spread of an infection

So far, the results we studied were existential. We showed that there are large graphs that can be infected (in various models) by a small set of initial nodes. From a practical point of view, a much more relevant question is how to reach as many nodes as possible in a *given graph.* Naturally, this question has applications in using network effects for the purpose of marketing. More formally, we can look at the following question: Given a number k, which set S with $|S| \leq k$ will eventually infect as many nodes of G as possible?

As an optimization problem motivated by viral marketing, this question was first suggested by Domingos and Richardson [129, 339], who studied it first [129] for a very general model in terms of Markov Random Fields (see Section 4.1), and then for a much more restricted linear model [339]. Unluckily, even in the Morris Contagion model from Section 8.4, this problem not only is NP-hard, but also hard to approximate.

Lemma 8.9 *Unless $P = NP$, the above problem cannot be approximated to within $O(n^{1-\epsilon})$ for any $\epsilon > 0$.*

Proof. We prove this with a reduction from SET COVER: Given sets $S_1, \ldots, S_m$, each a subset of $\{1, \ldots, n\}$, and a number $k \in \mathbb{N}$, are there k sets S_i whose union is the entire set?

For the reduction, we create a directed graph as follows: Let $\{s_1, s_2, \ldots, s_m\}$ be nodes corresponding to the m subsets and $\{u_1, u_2, \ldots, u_n\}$ be nodes corresponding to the n elements. Our construction will ensure that u_i becomes active when at least one of the nodes corresponding to sets containing u_i is active. We achieve this by connecting each s_k to the u_i's in it, and setting a threshold of $1/m$ for each u_i. Next, for a large constant c, we add $N = n^c$ more nodes $\{x_1, x_2, \ldots, x_N\}$. Each x_j is connected to all of the nodes u_i, and it becomes active only when *all* of the u_i are (i.e., the x_j have a threshold of 1). The construction is depicted graphically in Figure 8.2.

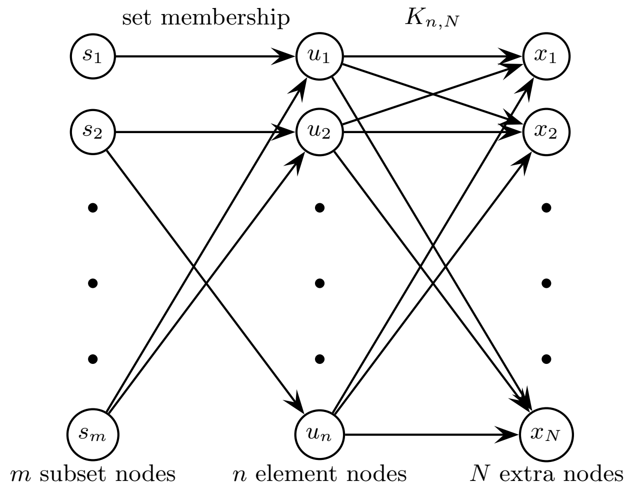


Figure 8.2: Construction for proving hardness of approximation

If there are at most k sets that cover all elements, then activating the nodes corresponding to these k sets will activate all of the nodes u_i, and thus also all of the x_j. In total, at least $N + n + k$ nodes will be active. Conversely, if there is no set cover of size k, then no targeted set will activate all of the u_i, and hence none of the x_j will become active (unless targeted). (Here, we are using implicitly that $k < n$. If $k \geq n$, then it is trivial to decide if all sets can be activated, for a Greedy Algorithm that adds sets to activate at least one more node in each iteration will work.) In particular, fewer than $n + k$ nodes are active in the end. If an algorithm could approximate the problem within $n^{1-\epsilon}$ for any ϵ, it could distinguish between the cases where $N + n + k$ nodes are active in the end, and where fewer than $n + k$ are. But this would solve the underlying instance of SET COVER, and therefore is impossible assuming $P \neq NP$. ■

Notice that this proof can be modified to deal with uniform thresholds, by adding more nodes.

In the Morris model, the problem thus turns out to be completely intractable. Yet, we may be interested in finding models that are more amenable to approximation. One small modification that turns out to make a significant difference is to assume that the thresholds are uniformly (and independently) random instead of deterministic. Thus, we obtain a model where each edge has a weight $w_e \geq 0$ such that $\sum_{e \text{ into } v} w_e \leq 1$. Each node v independently chooses a threshold $\theta_v \in [0,1]$ uniformly at random. The goal is to choose a set S with $|S| \leq k$ reaching as large a set as possible in expectation (over the random choices of θ_v). We define $f(S)$ to be objective function, i.e., the expected size of the finally infected set if we start with set S infected. The following theorem is the main result for this section:

Theorem 8.10 *There is a $1 - \frac{1}{e}$ approximation algorithm for the problem of selecting the set S maximizing $f(S)$.*

In fact, a simple greedy algorithm will give us the desired guarantees:

Algorithm 5 The simple greedy algorithm

```
1: Start with $S = \emptyset$.
2: for $k$ iterations do
3:     Add to $S$ the node $v$ maximizing $f(S + v) - f(S)$.
```

The proof of the performance guarantee consists of three parts, captured by the following lemmas.

Lemma 8.11 *A node v approximately maximizing $f(S + v) - f(S)$ can be found in polynomial time.*

Lemma 8.12 *f is a non-negative, monotone and submodular function of S.*

Recall that a function on sets is *monotone* if $f(S') \geq f(S)$ whenever $S' \supseteq S$, and *submodular* (having diminishing returns) if $f(S' \cup \{x\}) - f(S') \leq f(S \cup \{x\}) - f(S)$ whenever $S' \supseteq S$. Equivalently, submodularity is characterized by the condition that $f(S) + f(T) \geq f(S \cap T) + f(S \cup T)$ for all sets S, T. Also notice that the non-negativity and monotonicity of f are obvious.

Lemma 8.13 (Nemhauser/Wolsey/Fischer [111, 303]) *If f is a non-negative, monotone and submodular function, then the greedy algorithm is a $(1 - \frac{1}{e})$-approximation for the problem of maximizing $f(S)$ subject to the constraint that $|S| = k$.*

8.6.1 Proof of Lemma 8.13

We begin by proving the last lemma, which is naturally useful for other problems as well (such as Set Cover), and has developed into somewhat of a workhorse for many optimization tasks in machine learning.

Proof of Lemma 8.13. Let $v_1, v_2, \ldots, v_k$ be the nodes selected by the greedy algorithm (in the order in which they were selected), and denote $S_i = \{v_1, v_2, \ldots v_i\}$. Then, the marginal benefit derived from the addition of element v_i is $\delta_i = f(S_i) - f(S_{i-1})$. Let T be the optimal solution, and $W_i = T \cup S_i$.

First, the monotonicity of f implies that $f(T) \leq f(W_i)$ for all i. Because the algorithm chooses to add the best available node in the $(i+1)^{\text{st}}$ iteration, and the benefit of any elements added later cannot be greater by submodularity, the total objective value for the set W_i is at most $f(W_i) \leq f(S_i) + k\delta_{i+1}$, and thus also $f(T) \leq f(S_i) + k\delta_{i+1}$.

Solving this for δ_{i+1}, and using that $f(S_{i+1}) = f(S_i) + \delta_{i+1}$ now shows that $f(S_{i+1}) \geq f(S_i) + \frac{1}{k} \cdot (f(T) - f(S_i))$. We will prove by induction that $f(S_i) \geq (1 - (1 - \frac{1}{k})^i) \cdot f(T)$. The base case $i = 0$ is trivial.

For the inductive step from i to $i+1$, we use the above inequality to write

$$\begin{aligned}
f(S_{i+1}) &\geq f(S_i) + \frac{1}{k} \cdot (f(T) - f(S_i)) \\
&= (1 - \frac{1}{k}) f(S_i) + \frac{1}{k} \cdot f(T) \\
&\overset{\text{IH}}{\geq} (1 - \frac{1}{k})(1 - (1 - \frac{1}{k})^i) \cdot f(T) + \frac{1}{k} \cdot f(T) \\
&= (1 - \frac{1}{k} - (1 - \frac{1}{k})^{i+1} + \frac{1}{k}) \cdot f(T) \\
&= (1 - (1 - \frac{1}{k})^{i+1}) \cdot f(T),
\end{aligned}$$

completing the inductive proof. Using the fact that $(1-\frac{1}{k})^i \geq 1/\text{e}$ for all $i \leq k$, we obtain that the algorithm is a $(1-1/\text{e})$-approximation. ■

8.6.2 Proof of Lemma 8.12

In order to prove Lemma 8.12, we first observe that monotonicity of f is trivial. We then focus on a simpler model (proposed by Goldenberg et al. [178, 177] in the context of viral marketing) to illustrate the concepts of the proof.

Definition 8.14 (Independent Cascade Model) *Given a complete directed graph G with edge probabilities p_e, we consider an infection model where once a node u becomes active, it infects a neighboring node v with probability $p_{(u,v)}$. If the attempt succeeds, v becomes active; u, however, does not get to try infecting v again.*

We can observe that for each edge e, we can decide ahead of time (randomly) if the activation attempt will succeed when/if it happens, with probability p_e. We observe that, in the graph G of "successful edges", the nodes active in the end are exactly the ones reachable from S.

As a first step, we will show that the number of reachable nodes in a given graph G is a submodular function. We define $f_G(S)$ to be the number of nodes reachable from S in G, and prove

Claim 8.15 *f_G is submodular for all G.*

Proof. We need to show that $f_G(S \cup \{x\}) - f_G(S) \geq f_G(S' \cup \{x\}) - f_G(S')$ whenever $S \subseteq S'$. We write $R_G(S)$ for the set of all nodes reachable from S in G. Then, $f_G(S) = |R_G(S)|$, and because any node reachable from $S \cup \{x\}$, but not from S, must be reachable from x, we can observe that

$$|R_G(S \cup \{x\})| - |R_G(S)| \quad = \quad |R_G(S \cup \{x\}) \setminus R_G(S)| \quad = \quad |R_G(\{x\}) \setminus R_G(S)|.$$

By monotonicity of R_G, we have that $R_G(\{x\}) \setminus R_G(S) \supseteq R_G(\{x\}) \setminus R_G(S')$, so $|R_G(\{x\}) \setminus R_G(S)| \geq |R_G(\{x\}) \setminus R_G(S')|$, which proves submodularity of f_G. ■

From the submodularity for any fixed graph, we would like to derive that the same holds for the function f, over randomly chosen graphs. To that end, we use the following useful way of rewriting a random variable's expectation $\mathbb{E}[X]$: If $\{\mathcal{E}_1, \mathcal{E}_2, \ldots, \mathcal{E}_m\}$ is a partition of the probability space Ω, and X is a random variable on Ω, then:

$$\mathbb{E}[X] \quad = \quad \textstyle\sum_i \text{Prob}[\mathcal{E}_i] \cdot \mathbb{E}[X \mid \mathcal{E}_i].$$

In our case, the random variable X is the number of nodes reached from S, and $f(S) = \mathbb{E}[X]$ is its expectation. Let $G_1, \ldots, G_m$ be an enumeration of all graphs on n nodes (note that m is large). Notice that the events $\mathcal{E}_i = [G = G_i]$ form a partition of the probability space we are considering. Thus, the above identity implies that $f(S) = \sum_i \text{Prob}[G = G_i] \cdot f_{G_i}(S)$. Because all the coefficients $\text{Prob}[G = G_i]$ are non-negative, and each f_{G_i} is submodular, the submodularity of f now follows from the following

Fact 8.16 *A non-negative linear combination of submodular functions is itself submodular.*

Proof. Let $\alpha_i \geq 0$ for all i, and $f = \sum_i \alpha_i f_i$. If each f_i is submodular, then for all $S \subseteq S'$,

$$\begin{aligned} f(S \cup \{x\}) - f(S) &= \sum_i \alpha_i f_i(S \cup \{x\}) - \sum_i \alpha_i f_i(S) \\ &= \sum_i \alpha_i (f_i(S \cup \{x\}) - f_i(S)) \\ &\geq \sum_i \alpha_i (f_i(S' \cup \{x\}) - f_i(S')) \\ &= f(S' \cup \{x\}) - f(S'). \end{aligned}$$

■

Notice that we did not use a lot of properties of the Independent Cascade Process. In fact, we proved the following stronger lemma:

Lemma 8.17 *If the activation process is such that we can define a distribution on graphs $\{G_i\}$ such that $f(S)$ equals the expected number of nodes reachable from S under the distribution, then f is submodular.*

We would like to use this lemma to prove submodularity for the Linear Threshold Model that we started out with. In order to do that, we need to come up with a distribution over graphs such that the requirements of the lemma are met. In this case, it is not as obvious which distribution to chose, but we will be able to show that the following model works:

Definition 8.18 (Random Graph Model) *Each node v has at most one incoming edge which emanates from u with probability $w_{(u,v)}$, the weight of the influence of u on v. (Recall that $\sum_u w_{(u,v)} \leq 1$ for all v). With probability $1 - \sum_u w_{(u,v)}$, v has no incoming edge.*

Claim 8.19 *This model is equivalent to the Threshold Model, in the sense that the expected number of nodes reached is the same.*

Proof. We will prove by induction on t that for each time step $t \geq 0$, and any node sets $T \subseteq T'$, the probability that exactly T is active at time t and T' at time $t+1$ is the same in both the threshold and the random graph processes.

In the base case $t = 0$, the probability is 1 for the pair $(\emptyset, S)$ for both processes, and 0 for all other pairs, because both processes start with only the selected set S active.

For the inductive step, assume that $T \subseteq T'$ are the active sets at time $t-1$ and t, and consider some node $v \notin T'$. We investigate the probability that v becomes active at time $t+1$ in either process.

In the threshold model, because v was not active at time t, we know that $\theta_v \geq \sum_{u \in T} w_{(u,v)}$. However, subject to that, θ_v is still uniformly random, so by the Principle of Deferred Decisions, we can re-choose θ_v uniformly at random from the interval $(\sum_{u \in T} w_{(u,v)}, 1]$.

v becomes active at time $t+1$ iff $\theta_v \leq \sum_{u \in T'} w_{(u,v)}$. This happens with probability, $\frac{\sum_{v \in T' \setminus T} w_{(u,v)}}{1 - \sum_{u \in T} w_{(u,v)}}$.

Now, let us look at the probability that $v \notin T'$ becomes active at time $t+1$ in the random graph process. Because v is not active at time t, we know that v's incoming edge, if any, does not come from T. Thus, v becomes active at time $t+1$ iff the edge comes from $T' \setminus T$.

The probability that v's edge does not come from T is $1 - \sum_{u \in T} w_{(u,v)}$, and the probability that it comes from $T' \setminus T$ is $\sum_{u \in T' \setminus T} w_{(u,v)}$. Hence, the conditional probability of v becoming active is $\frac{\sum_{u \in T' \setminus T} w_{(u,v)}}{1 - \sum_{u \in T} w_{(u,v)}}$.

So for any individual nodes, the probability of activation at time $t+1$ is the same under both processes. As these decisions are independent in both processes (thresholds resp. edges are chosen independently), the probability that exactly all nodes from W become active is the same under both processes for any set W. So

$$\begin{aligned}&\text{Prob}[\langle T', T''\rangle \text{ active at time } \langle t, t+1\rangle]\\ &\quad= \sum_T \text{Prob}[\langle T, T'\rangle \text{ active at time } \langle t-1, t\rangle] \cdot \text{Prob}[\text{exactly } T'' \setminus T' \text{ becomes active} \mid T' \text{ active at time } t]\end{aligned}$$

is the same for both. Hence, by induction, the processes behave the same, and in particular, the final expected number of activated nodes is the same. ■

Notice that combining all of the above, we have established Lemma 8.12, i.e., the fact that f is submodular.

8.6.3 Proof of Lemma 8.11

It remains to show how to find, in polynomial time, a node giving largest marginal gain at any point. In fact, finding the node of largest marginal gain is too much to hope for — as shown in [377] (Independent Cascade Model) and [91] (Linear Threshold Model), it is #P-hard to evaluate the objective function, and thus also to compute the node with largest marginal gain. However, we can establish a slightly weaker fact, which is sufficient for our purposes. As stated in Lemma 8.11, we can find, in polynomial time, a $(1-\epsilon)$-approximately best node to add.

Proof of Lemma 8.11. First, we describe an algorithm for estimating the marginal gain of adding a node. If we can get sufficiently accurate estimates for each node, then a choice based on these estimates will give us an approximately best node.

The algorithm is simply to simulate the random process multiple times, and take the average of the number of additional nodes reached over all these simulations. This average will certainly in the limit converge to the expectation, but the question is how quickly, i.e., how often is "multiple" times? If some "gain values" had very low probability of occurring, but were extremely high, then we may need a large number of simulations to get a good estimate. Fortunately, in our case, the values we sample have a natural upper bound of n, the number of nodes. We will see that the number of iterations will not need to be very high as a result.

Formally, we will use the Chernoff-Hoeffding Bound, as given by the following theorem.

Theorem 8.20 (Chernoff-Hoeffding Bound [204, 278]) *If $X_1, \ldots, X_m$ are independent random variables with $0 \leq X_i \leq b_i$ for all i, and $X = \sum_i X_i$, and $\mu = \mathbb{E}[X]$, then for all $\Delta \geq 0$,*

$$\text{Prob}[|X - \mu| \geq \Delta] \quad \leq \quad 2\mathrm{e}^{\frac{-\Delta^2}{\sum_i b_i^2}}.$$

In our case, we let X_i be the outcome of the i^{th} simulation. Thus, $0 \leq X_i \leq n$ for all i, so $b_i = n$. Also, because at least one node is active in each outcome (the start node itself), we have that $X_i \geq 1$, and thus $\mu \geq m$. Choosing $\Delta = \epsilon\mu$, we obtain that

$$\text{Prob}[|X-\mu| \geq \epsilon\mu] \quad \leq \quad \text{Prob}[|X-\mu| \geq \epsilon m] \quad \leq \quad 2\mathrm{e}^{\frac{-(\epsilon m)^2}{mn^2}} \quad = \quad 2\mathrm{e}^{\frac{-\epsilon^2 m}{n^2}}.$$

Thus, if we are aiming for a $(1 \pm \epsilon)$-approximation with probability at least $1-\delta$, it is sufficient to require that $2\mathrm{e}^{\frac{-\epsilon^2 m}{n^2}} < \delta$. Solving for m gives that $m > \frac{n^2}{\epsilon^2} \log \frac{2}{\delta}$ simulations are sufficient. By a Union Bound over all (at most n) iterations of the greedy algorithm, and all n nodes in each iteration, all simulations have error at most ϵ with probability at least $1 - n^2\delta$.

If we want, for instance, success probability at least $1 - \frac{1}{n^2}$, we can choose $\delta = \frac{1}{n^4}$, and then, we need to run $O(\frac{n^2}{\epsilon^2} \log n)$ iterations to get a $(1 \pm \epsilon)$ accurate estimate.

It still remains to verify that we actually obtain a close to best node when picking a node based on ϵ-estimates. In the worst case, the picked node's gain is over-estimated by a factor of $(1+\epsilon)$, while the true best node's gain is under-estimated by a factor of $(1-\epsilon)$. But because the picked node appeared to be better, its gain must have been within a factor of $\frac{(1-\epsilon)}{(1+\epsilon)} \geq 1-2\epsilon$ of the best node's. So we have a $(1-2\epsilon)$-approximate best node in each iteration, in polynomial time. ■

By essentially mimicking our earlier proof of the Nemhauser-Wolsey-Fischer theorem, we can show the following stronger version:

Theorem 8.21 *If we choose a $(1-\epsilon)$-approximate best node in each iteration of the greedy algorithm, then we get a $1-\frac{1}{e}-\epsilon'$ approximation algorithm, where $\epsilon' \to 0$ as $\epsilon \to 0$, polynomially fast.*

8.6.4 More General Diffusion Models

We just showed that two natural models for influence propagation — the Independent Cascade Model of Goldenberg et al. [178, 177] and a variant of the Linear Threshold Model of Granovetter [189] — lead to submodular overall influence functions, and thus a $1-1/e$ approximation algorithm. A natural question is then which other models permit such approximations.

Initial progress was made on this question in [233]. There, it is shown that a generalization of the cascade model still leads to submodularity. The probability of activation success $p_{u,v}(A)$ along an edge (u,v) may depend on which other nodes A have previously tried (and failed) to activate v. However, the probability $p_{u,v}(A)$ must be non-increasing in A, i.e., the marginal probability of success decreases. [233] shows that this condition is sufficient to yield submodularity. Interestingly, it is also shown that for this model, there may be *no* graph distribution yielding the same expected number of active nodes. Hence, [233] uses a different technique for proving submodularity.

The result from [233] is further generalized in a beautiful result of Mossel and Roch [299]. They consider a generalization of the threshold model, where each node v has a local threshold function $f_v(A)$, and becomes active when $f_v(A) \geq \theta_v$, where A is the set of previously active nodes. Thus, the linear threshold model is the special case where $f_v(A) = \sum_{u \in A} w_{u,v}$. Mossel and Roch resolve the following conjecture of [234]:

Theorem 8.22 ([299]) *If each f_v is non-negative, monotone non-decreasing, and submodular, then so is the overall activation function.*

The proof is by a more intricate coupling argument in the style of our proof of Claim 8.19. It uses the following alternative characterization of submodularity: a function f is submodular if and only if for all sets A, B:

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B). \tag{8.1}$$

The first key lemma states that initial activations can be deferred without changing the outcome of the process. More formally, this is expressed by the following lemma from [233, 299]:

Lemma 8.23 *Let $S = S_1 \cup S_2 \cup \cdots \cup S_r$ be a partition of the seed set. Let the random variable T be the set of nodes activated at the end of the generalized threshold process if S is the initial active set of nodes. Define the random variable T' by the following process: first activate S_1 and let the process quiesce (i.e., no new activations occur); then activate S_2 and let the process quiesce again; do this for all S_i in order. Let the random variable T' be the final active set after the process has quiesced.*

Then, T and T' have the same distribution.

Proof. The lemma should be "intuitively clear," because the threshold process does not depend on any order, and all initial seeds will be added eventually. To prove it more formally, we can use the obvious coupling between the two random processes. Consider the original process $T_0, T_1, T_2, \ldots, T$, starting with $T_0 = S$ active, and the "piecemeal" process $T'_0, T'_1, T'_2, \ldots, T'$ which adds sets S_i one by one at some times t along the way. For both processes, consider the same thresholds θ_v at all nodes.

First, because all activation functions are monotone, and the S-process starts with a superset of nodes (and any nodes added by the piecemeal process have been added by the S-process before), for every time step t, we have $T_t \supseteq T'_t$, so at the end, we must have $T \supseteq T'$. On the other hand, at the step $\hat{t}$ when S_r is added, the piecemeal process has added all of S as well as possibly additional nodes, so $T'_{\hat{t}} \supseteq T_0$. By monotonicity, for all subsequent steps $\hat{t}+t$, we have $T'_{\hat{t}+t} \supseteq T_t$. In particular, at the end, $T' \supseteq T$. Thus, we have shown that $T' = T$; in particular, the two processes have the same distribution of final outcomes. ■

Proof of Theorem 8.22. In order to show that the objective function satisfies Inequality (8.1), we consider four separate processes, adding different sets of nodes and running until quiescence for each. Because the process quiesces when no new nodes are activated for one round, quiescence must be reached after at most n steps. We therefore consider times that are multiples of n as times to add new nodes.

- The process C_t starts with $C_0 = A \cap B$, and runs until quiescence; let the final active set be $\hat{C} = C_n = C_{3n}$.
- The process A_t starts with $A_0 = A \cap B$. Then, at time n, it adds the set $A \setminus B$ and runs until quiescence; let the final active set be $\hat{A} = A_{2n} = A_{3n}$.
- The process B_t starts with $B_0 = A \cap B$. Then, at time $2n$, it adds the set $B \setminus A$ and runs until quiescence; let the final active set be $\hat{B} = B_{2n} = B_{3n}$.
- The process D_t starts with $D_0 = A \cap B$. At time n, it adds the set $A \setminus B$, and at time $2n$, it adds $B \setminus A$. It runs until quiescence; the final active set is $\hat{D} = D_{3n}$.

By Lemma 8.23, the final active sets $\hat{A}, \hat{B}, \hat{C}, \hat{D}$ have the same distribution as the active sets at the end if we simply run the process starting from $A, B, A \cap B, A \cup B$, respectively. We will show how to couple the processes so that $\hat{C} \subseteq \hat{A} \cap \hat{B}$ and $\hat{D} \subseteq \hat{A} \cup \hat{B}$. In particular, this implies that

$$|\hat{C}| + |\hat{D}| \;\leq\; |\hat{A} \cap \hat{B}| + |\hat{A} \cup \hat{B}| \;=\; |\hat{A}| + |\hat{B}|.$$

(Notice that even if we were not interested in the cardinality of the final set, but rather in some arbitrary monotone and submodular function g applied to the final set of active nodes, this proof would imply submodularity.)

First, notice that because C_t does not grow after $t = n$, while A_t and B_t do (and until time n, they are the same), we have that $C_t \subseteq A_t$ and $C_t \subseteq B_t$ for all time steps t, and in particular $\hat{C} \subseteq \hat{A} \cap \hat{B}$.

The more involved part is showing that $\hat{D} \subseteq \hat{A} \cup \hat{B}$. Consider a coupling in which all nodes have the same threshold for the processes A_t, B_t, D_t. (However, we will interpret thresholds differently momentarily.) Then, $D_{2n} = A_{2n} = A_{3n}$ and $B_{2n} \subseteq D_{2n}$. We will show that $B_{3n} \setminus B_{2n} \supseteq D_{3n} \setminus D_{2n}$. Then,

$$\hat{D} = D_{3n} \;\subseteq\; D_{2n} \cup (B_{3n} \setminus B_{2n}) \;\subseteq\; A_{2n} \cup B_{3n} \;=\; \hat{A} \cup \hat{B}.$$

Because we add the same set of nodes $B \setminus A$ to both B_{2n+1} and D_{2n+1}, we have that $B_{2n+1} \subseteq D_{2n+1}$. For all nodes v that are not active in B_{2n+1}, the threshold θ_v must be uniformly random from $(f_v(B_{2n}), 1]$; similarly, for all nodes v not active in D_{2n+1}, θ_v is uniformly random in $(f_v(D_{2n}), 1]$.

In the threshold process as described before, a node v which is not active at time $2n+1$ becomes active at time $t+1$ iff $\theta_v \leq f_v(D_t)$, which happens with (conditional) probability $\frac{f(D_t)-f(D_{2n})}{1-f(D_{2n})}$. But if our goal is to achieve this particular activation probability, we can also do it by having node v become active at time $t+1$ iff $\theta_v \geq 1 - (f_v(D_t) - f_v(D_{2n}))$. Since the length of the interval $(1 - f_v(D_t) - f_v(D_{2n}), 1]$ is the same as that of $(f_v(D_{2n}), f_v(D_t)]$, and both are contained in $(f_v(D_{2n}), 1]$, the distribution of outcomes of the processes are the same. So for time steps $t = 2n+1, \ldots, 3n$, the processes B_t and D_t both use this alternative *antisense coupling* interpretation of the thresholds. See Figure 8.3 for an illustration.

Now, we can use an induction proof to show that $B_t \setminus B_{2n} \supseteq D_t \setminus D_{2n}$ for all $t \geq 2n+1$. For the base case $t = 2n+1$,

$$B_{2n+1} \setminus B_{2n} \;=\; A \setminus B_{2n} \;\supseteq A \setminus D_{2n} \;=\; D_{2n+1} \setminus D_{2n}.$$

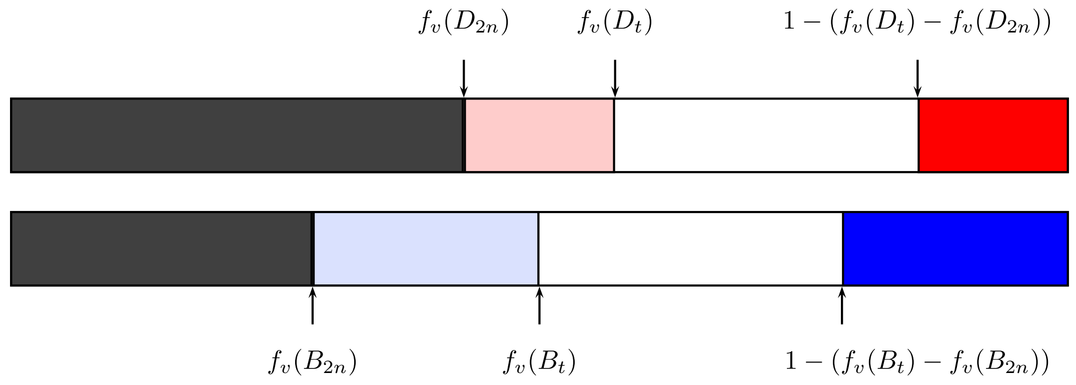


Figure 8.3: An illustration of thresholds in the antisense coupling.

For any later time step t, consider a node $v \notin D_t$; in particular, $\theta_v \geq f(D_{2n})$. If v becomes active at time $t+1$ in the D_t process, then by definition of the antisense coupling,

$$\theta_v \;\geq\; 1-(f(D_t)-f(D_{2n})) \;\overset{(*)}{\geq}\; 1-(f(B_t)-f(B_{2n})).$$

We will show the step (*) in a moment. Therefore, v also becomes active in the B_t process. Since this holds for all candidate nodes v, we have shown that $B_{t+1} \setminus B_{2n} \supseteq D_{t+1} \setminus D_{2n}$, completing the inductive proof.

Finally, to prove the step labeled (*), in the following, we use that f is submodular and $D_{2n} \supseteq B_{2n}$ (for the first inequality), and that f is monotone and $D_t \setminus D_{2n} \subseteq B_t \setminus B_{2n}$ by induction hypothesis (for the second inequality):

$$\begin{aligned} f(D_t)-f(D_{2n}) \;&=\; f(D_{2n}\cup(D_t\setminus D_{2n}))-f(D_{2n}) \;\leq\; f(B_{2n}\cup(D_t\setminus D_{2n}))-f(B_{2n}) \\ &\leq\; f(B_{2n}\cup(B_t\setminus B_{2n}))-f(B_{2n}) \;=\; f(B_t)-f(B_{2n}). \end{aligned}$$

This completes the proof of the theorem. ■

8.7 Further Reading

Characterizing when Equivalent Graph Distributions Exist

In the context of Theorem 8.22, we mentioned that there exist natural models of influence (Decreasing Cascade Model, Submodular Threshold Model) that have submodular objective functions, but for which there is no equivalent distribution of graphs to prove submodularity; in particular, Lemma 8.17 cannot be applied. This raises the question of which influence models *do* allow a proof of submodularity via an equivalent distribution of graphs. This question is answered in a paper by Salek et al. [344]. For a function g defined on sets, let g_v denote the "discrete derivative" $g_v(S) := g(S\cup\{v\})-g(S)$, and inductively define $g_{T\cup\{v\}}(S) := g_T(S\cup\{v\})-g_T(S)$. (It is not hard to see that this definition does not depend on which element of a set is chosen first.) [344] shows that there is an equivalent distribution over graphs if and only if the following three conditions hold simultaneously:

1. g itself is non-negative.

2. All discrete derivatives g_T with $|T|$ odd are non-negative.

3. All discrete derivatives g_T with $|T|>0$ even are non-positive.

Speeding up the Optimization

The greedy algorithm for influence maximization as described here scales up to a few 10,000 nodes. A huge amount of subsequent work has aimed to speed up the optimization so as to scale to millions or billions of nodes. Most of the algorithms are heuristics that sacrificed the approximation guarantees for efficiency, while a few are heuristics that keep the $1 - 1/e$ approximation guarantee while "typically" speeding up the execution, but not guaranteeing to do so. This literature is much too extensive to review here, but interested readers are pointed to the survey book by Chen, Lakshmanan, and Castillo [90].

A noticeable new idea was put forth by Borgs et al. [56], and further improved and implemented by Tang et al. [364, 365]. The idea in these papers is to consider influence in reverse, i.e., determining which nodes v would have influenced a particular target u. By sampling enough targets, choosing nodes that appear in many sets of "potential influencers of u" will give similar provable approximation guarantees, since it still solves a coverage problem. With a careful choice of expanding influence search and the right data structures, this can bring down the required running time to near-linear.

Some Different Influence Models

A somewhat different influence model was considered by Even-Dar and Shapira [144]. They consider the *voter model* [100, 207]: in each iteration, each node v simultaneously chooses a random neighbor according to some distribution (e.g., uniformly), and copies the state of that neighbor. In the limit, for a connected graph, this will result in all individuals adopting the same behavior. By showing equivalence to a Markov Chain, [144] shows that the influences of nodes factor, and therefore, a simple greedy algorithm is optimal. Even if different nodes have different activation costs, there is an FPTAS for the problem, based on the FPTAS for KNAPSACK [372].

As discussed above, when thresholds at nodes are fixed, the problem becomes harder to approximate. In fact, even if all nodes have the same constant threshold $\tau \geq 2$, the problem of finding a minimum-size set of nodes that eventually activates the entire graph is NP-complete, a fact shown for $\tau \geq 3$ by Dreyer and Roberts [220] and for $\tau = 2$ (as well as majority and some other rules) by Chen [88]. In fact, [88] showed approximation hardness results for this version of the problem.

Competition between Cascades

A direction which has received a fair share of attention is multiple competing influences. The motivation here is viral marketing, when different companies want to use word-of-mouth in the same social network to promote competing products. Each node can only adopt (and recommend) one product, and the main question is how to deal with the possible timing issues and create a well-defined and natural model. Probably the first paper offering explicit analysis of such games is one by Dubey et al. [134], which studies equilibria of such competition in the simple linear model of [339]. A similar model is studied by Grabisch et al. [186], who derive various properties of the equilibrium outcomes of the game between the companies.

Several other papers considered the aspect of competition. Bharathi et al. [45] augmented the cascade model with a timing component, which leads to a simple tie-breaking for choosing which product a node adapts. One of the main results of [45] is that the benefit of a set for the last player is still a submodular function; using a general result of Vetta [374], this also implies that the "Price of Competition" is at most a factor of 2, i.e., regardless of how many companies are competing, the expected reach of these influences combined is at least half as much as if they all pooled and coordinated their resources. Carnes et al. [78] also consider the problem of the last player in this setting, but without a timing component. Instead, they propose two tie breaking rules: one distance-based (with an additional distance notion introduced for the problem), and one by uniform random choice from among all active neighbors.

Goyal and Kearns [185] proposed a variant of a threshold model that is amenable to analysis, and showed that the Price of Competition for two companies is at most 4. As pointed out by He and Kempe [200], the model they define also satisfies the necessary conditions to apply Vetta's result [374], so that the Price of Competition can in fact be bounded by 2 for any number of companies. In general, defining competitive threshold models appears more difficult than cascade models: [57] investigate many natural ways of defining competitive threshold models, and show that they have surprising non-monotonicity or non-submodularity properties.

Tipping Points and Phase Transitions

The extent of diffusions (or epidemic diseases) through a social network has also received attention in the physics and economics communities (e.g., [214, 215, 308, 322, 323, 391]). There, the assumption is usually (implicitly) that the social network is uniformly random subject to its degree distribution, and that the initial set of infected individuals is also random. The interest in this line of work is then in "tipping point" or "phase transition" behaviors, i.e., at what settings of parameters does the reach of the influence go from nearly negligible to significant.

Equilibrium States and Convergence

As we saw implicitly in Section 8.4, there can be multiple "equilibrium" states of the system. In particular, it is possible for some of the nodes to end up active, and others inactive. In terms of technology, this means that some are still using the old (inferior) technology, while others have switched to the new and superior one. This coexistence disappears if we modify the model as follows: with small probability, nodes choose the wrong response to their neighbors' choices. Thus, sometimes, a node will become active even though it would be superior (in terms of the payoffs defined in Section 8.4) to remain inactive, or vice versa. In this case, the system always converges to the state in which all nodes use the superior technology [223]. An interesting question is then how quickly this convergence happens. Following up on work of Ellison [137], which analyzed the special case of complete graphs and k-nearest neighbor graphs on the line or cycle (for $n \gg k$), Montanari and Saberi [292] showed that the convergence time is asymptotically characterized by the *tilted cutwidth* of the graph, a certain isoperimetric property. In particular, graphs that expand well tend to have *slow* convergence, while globally poorly expanding graph have fast convergence. Note that this stands in stark contrast to the spread of diseases or models like the cascade model, where expansion tends to imply fast spread.

Maximizing Profits from Sales

In our optimization problem so far, we have assumed that the sole goal is to maximize the number of individuals adopting the innovation. In a more realistic viral marketing setting, these adoptions will correspond to *sales* of an item, and the company will have the ability to set different prices at different times, or even for different consumers. This version has been studied by Hartline et al. [196]. They show that the following "Influence and Exploit" strategies are within a factor $\frac{1}{4}$ of optimal: first, give the product for free to a most influential subset; then, choose prices for the remaining bidders in random order to maximize revenue, ignoring their network effects. As a second step, they show that the resulting problem of selecting the most influential set leads to a submodular objective function, though not necessarily a monotone one. Using an 0.4-approximation algorithm of Feige et al. [152], they thus obtain a constant-factor approximation algorithm. A slightly modified model was studied by Arthur et al. [21]: in their model, the seller cannot choose arbitrarily whom to offer the product to. Instead, the product spreads by recommendations, but the seller can decide on a price for each individual once a recommendation occurs. In this model, Arthur et al. prove competitive guarantees based on the type of buyer model.

Adopting Multiple Technologies

In our derivation of activation thresholds in Section 8.4, we assumed implicitly that each node adopted only one technology. Instead, a node v may have the choice of adopting both the old and new technologies, at an extra cost of $d_v \phi$, for some parameter ϕ (where d_v is v's degree). In return, v gets communication benefits of $1 - q$ or q for all neighbors. For instance, a person could learn multiple languages, maintain multiple messaging softwares, or multiple data formats, such as CDs and MP3s. Immorlica et al. [211] considered this question. In their most basic model, they assume $a = b = 0$, i.e., no communication is possible with differing technologies. They investigate the regions for the parameters ϕ, q which allow the new technology to spread through the network. Interestingly, this region is not always convex. They then consider the generalization to partial compatibility, i.e., values $a, b > 0$, and its impact on the diffusion of the new innovation, again noticing non-monotonic behavior.

Submodular Optimization

The technique we used in the proof of Theorem 8.10 is quite general: prove that the function to be optimized is non-negative, monotone and submodular; then, the greedy algorithm will be a $1 - 1/\mathrm{e}$ approximation. This technique has been applied in a number of scenarios such as selecting sensors, variables to observe, or samples to draw [113, 249, 250, 251, 267]. In different scenarios, we may be interested in somewhat different objectives or constraints. For instance, we saw above that when sales prices are taken into account, the natural objective function is submodular, but not monotone. As another examples, Krause et al. [252] show that a sensor placement and scheduling problem for coverage can be expressed as maximizing the minimum of multiple submodular functions $f_i(A_i)$, where the sets A_i must be disjoint, and their union satisfies a size constraint. [252], among others, gives a 1/6-approximation algorithm for this problem.

The frequency with which submodular objective functions appear in many settings has led to a lot of work on approximating them with various constraints. In a classic paper, Iwata et al. [212] show that for any submodular function f, finding a set S *minimizing* $f(S)$ can be achieved in polynomial time. (Notice, however, that we did not include any size constraints on S.) For the maximization problem, there has been a lot of recent interest in generalizing the result of Nemhauser et al. [111, 303]. Sviridenko [361] shows how to obtain a $1 - 1/\mathrm{e}$ approximation for monotone submodular functions even when different elements have different inclusion costs, and a total budget is given. Calinescu et al. [76, 375] give a beautiful $1 - 1/\mathrm{e}$ approximation algorithm for the case when the solution is constrained to be from a given matroid[2]. If the solution is required to belong to k different matroids, then a recent result of Lee et al. [263] shows how to approximate the best solution to within a factor of $1/(k + \epsilon)$ for monotone functions.

For non-monotone submodular functions, we previously mentioned the 0.4-approximation algorithm of Feige et al. [152]. Lee et al. [262] consider the more general constraint of restricting solutions to k different matroids, and give a $\frac{1}{k+2+1/k+\epsilon}$ approximation for this problem, as well as a $1/5 - \epsilon$ approximation when solutions must meet k Knapsack constraints. Vondrák [376] provides some lower bounds on how well these problems can be approximated.

Learning the Influence Model

In order to maximize the diffusion of an innovation in a network, it is necessary to first learn the influence that individuals have over each other. So far, we have assumed that these data are given precisely, which is certainly unrealistic. Indeed, they must be learned first, presumably from past observed data. This question has recently been investigated by many papers, with different objectives and results (see, e.g., [18, 132, 133, 181, 182, 184, 302, 305, 343, 388, 394] for a sample). Typically, the approach is to formulate the likelihood of the observed cascades under some modeling assumption, and then infer model parameters (such as influence probabilities or rates) so as to maximize the likelihood or log-likelihood. Depending on the model, the objective may be convex (and can thus be optimized), or heuristics are employed. A notable difference is [132], which does not aim to explicitly infer the parameters of the model, but rather uses the graph distribution representation (in the vein of Lemma 8.17) of the process, and aims to infer a sparse distribution over graphs.

[2]A matroid is a downward-closed set system with the following *exchange property*: If S, T are two sets in the set system with $|S| < |T|$, then there is an element $u \in T \setminus S$ such that $S \cup \{u\}$ is also in the matroid. In other words, elements of sets can be exchanged one by one.

Chapter 9

Epidemic Algorithms

In the previous chapter, we considered examples of processes that naturally spread through networks in an epidemic fashion, such as information or behaviors, or — of course — epidemic diseases. One intuitive observation about such spreading processes is that they are very *resilient*: even if a few individuals are immune or removed from the network, the epidemic tends to reach a similarly large share of the network. This resilience, while undesirable for actual epidemics, is very desirable for communication and computation in computer networks: even if a few nodes are faulty, we would like for information to diffuse to the rest of the network, and for computation to succeed. The first to propose simulating the behavior of epidemics as a principle for systems design were Demers et al. [123], in the context of maintaining replicated data bases. Their idea was that updates to the data bases would not be distributed directly to all other copies, but passed around like a rumor, with nodes randomly choosing others to exchange updates with.

Since the appearance of [123], there has been a large body of work on epidemic algorithms as a primitive for computer networks. Among the first to propose a system based on these approaches were van Renesse et al. with the design of Astrolabe [370, 371]. Here, we will investigate some of the analysis underlying two key primitives of distributed computation: spreading a single message to everyone, and computing the average of the values held by all nodes in the network. These lie at the core of many more complex network tasks.

9.1 Spreading One Message

To get a feel for the benefits of epidemic algorithms, let us begin by considering an obvious alternative, also used in many systems. To disseminate a message to all nodes, the nodes could build (a priori) a tree, and then forward the message along the edges of the tree. So long as the height of the tree is logarithmic, and the degrees bounded by a constant (e.g., for a complete binary tree), the message will reach all nodes in $O(\log n)$ steps. The best completion time (time by which the last node receives the message) is achieved by a binomial tree. Notice that $\Omega(\log n)$ is also an obvious lower bound: in each round, the number of nodes having the information can at most double.

The problem with any tree is that it is not fault-tolerant. If one node fails, then none of the nodes in the subtree rooted at it will receive the message. In order to achieve more fault-tolerance, we will need more redundancy in messages sent. One could make the graph 2-connected, or more highly connected. But in order to achieve higher fault-tolerance, the graph structures will have to become more complicated, and thus difficult to compute and maintain. Instead, we will see that simple randomized protocols based on the behavior of epidemics achieve essentially the same time bounds, and are naturally fault-tolerant. Such algorithms are usually called *epidemic algorithms* or *gossip algorithms*, and are based on information exchanges between random node pairs.

For the time being, we assume that every pair of nodes can in principle communicate, i.e., that the communication graph is complete. (We will discuss some pointers to literature avoiding this assumption at the end of the chapter.) In all versions of gossip protocols we consider here, we assume that there are synchronous rounds, and in each round, each node calls a random other node and exchanges information. These exchanges can be divided into three categories:

1. In *push gossip*, each (informed) sender randomly picks a partner and sends a message.
2. In *pull gossip*, each node picks a random partner, and receives a message from the partner (if it has one).
3. In *push & pull gossip*, each node picks a random partner, and both forwards and receives a message.

Naturally, push & pull gossip makes most sense when multiple messages are being exchanged, as otherwise, no informed node ever needs to pull, and no uninformed node can ever push. For instance, exchanging information in both directions is useful in synchronizing a replicated distributed database periodically, the original application of [123].

In deciding whether to use gossip as a primitive, an important consideration is its speed: how quickly do all nodes receive the message? We expect the time to be roughly $O(\log n)$, as the number of nodes having the message roughly doubles until most nodes already have the message.

Proposition 9.1 *With high probability, it takes $O(\log n)$ rounds to get a message to everyone.*

Proof. We first analyze the behavior until more than $n/3$ nodes have the message. So consider a time t when at most $m \leq \frac{n}{3}$ nodes have the message. Then, each of the m messages sent has probability at least 2/3 of reaching a previously uninformed node, so in expectation, at least $\frac{2m}{3}$ sent messages reach previously uniformed nodes.

Unluckily, this does not quite guarantee that $2m/3$ new nodes will be informed, as some of these messages will reach the same nodes. We want to upper-bound how frequently this happens. For any given pair of messages, the probability that they both go to the same uninformed node is $O(\frac{1}{n-m})$. As there are at most $\binom{m}{2}$ such pairs, the expected number of such collisions is at most $\frac{m^2}{2(n-m)} \leq \frac{m^2}{2} \cdot \frac{1}{2m} = \frac{m}{4}$. Thus, the expected number of newly infected nodes is at least $\frac{2m}{3} - \frac{m}{4} = \frac{5m}{12}$.

Let the random variable X_t be the number of informed nodes at time t. By our arguments above, $X_0 = 1$, and $\mathbb{E}\left[X_{t+1} \mid X_t = m \leq \frac{n}{3}\right] \geq \frac{17}{12}m$. By induction, $E[X_t] \geq (\frac{17}{12})^t$. We can now apply Markov's Inequality to show that with high probability, after $O(\log n)$ rounds, at least $\frac{n}{3}$ nodes are active.

Once $\frac{n}{3}$ nodes are active, any other node v is *not* called by an informed node with probability at most $(1 - \frac{1}{n})^{\frac{n}{3}} \leq \mathrm{e}^{\frac{-1}{n} \cdot \frac{n}{3}} = \mathrm{e}^{-1/3}$. So after $t = 6 \log n$ independent rounds of this process, the probability that a particular node v is still uninformed is at most $(\mathrm{e}^{-1/3})^{6 \log n} = n^{-2}$. By a Union Bound over all n nodes, the probability that all nodes get the message in $O(\log n)$ rounds is then at least $1 - \frac{1}{n}$. By increasing the constant in the $O(\log n)$ time bound, we can improve the probability to $1 - 1/n^c$, for any c. ■

A more careful analysis is performed in [164, 326], where the precise constants and lower-order terms are worked out to guarantee a high-probability bound.

In the proof of Proposition 9.1, notice the following: From the point at which $n/3 = \Omega(n)$ nodes have the message until the point when *all* nodes have the message, there are $\Omega(\log n)$ rounds, and in each such round, each informed node sends the message to some other node, for a total of $\Omega(n \log n)$ messages. On the other hand, using any tree would only require sending $O(n)$ messages. Thus, gossip achieves its fault-tolerance with high *redundancy*: by using $\Omega(\log n)$ time as many messages as necessary.

Naturally, one wonders whether such an amount of redundancy is inherent in any fault-tolerant approach, or in any gossip-based approach. Karp et al. [227] give a partial answer to this question. They show that any protocol based on uniform gossip must send $\omega(n)$ messages, and, under additional restrictions on the protocol, $\Omega(n \log \log n)$ messages. Conversely, they analyze more carefully the "push-pull" version of gossip, and show that if the terminating point is chosen carefully, only $O(n \log \log n)$ messages are sent.

Theorem 9.2 ([227]) *If push-pull is run for $\log_3 n + O(\log \log n)$ rounds, then, w.h.p., all nodes have the message, and, in addition, the total number of messages sent is $O(n \log \log n)$.*

Proof. Solely for the purposes of analysis, we divide the algorithm into four phases.

1. **Startup phase:** This phase lasts until $\log^4 n$ nodes have the message with high probability (probability at least $1-n^{-c}$ for some c). We observe that within $O(1)$ rounds, the number of informed nodes doubles with high probability (because each informed node will have called a previously uninformed one), and so the entire phase takes $O(\log\log n)$ rounds for the goal of achieving $\log^4 n$ informed nodes.

2. **Exponential growth phase:** This phase lasts until $\frac{n}{\log n}$ nodes are informed. We write S_t for the number of nodes having the message at time $t-1$, and m_t the number of messages that were sent in round t. Because in expectation, each informed node calls one node, and is called by one, we have that $\mathbb{E}[m_t] = 2S_{t-1}$.

   In fact, m_t is sharply concentrated, i.e., unlikely to deviate far from its expectation. To see this, we let

   $$X_{uv} \quad = \quad \begin{cases} 1 & \text{if node } u \text{ calls } v \text{ in round } t \\ 0 & \text{otherwise.} \end{cases}$$

   Then, $m_t = \sum_{u \in S_{t-1}} \sum_v (X_{uv} + X_{vu})$. Because m_t is a sum of negatively dependent $\{0,1\}$ variables, (a different version of) Chernoff Bounds shows that m_t is sharply concentrated around its mean.

   Theorem 9.3 (Chernoff Bounds [92, 300]) *If $X_1, X_2, \ldots, X_n$ are independent (or negatively correlated) $\{0,1\}$ random variables, with $X = \sum_i X_i$, $\mu \geq \mathbb{E}[X] = \sum_i \text{Prob}[X_i = 1]$, and $\delta > 0$, then*

   $$\text{Prob}[X > (1+\delta)\mu] \quad < \quad \left(\frac{e^\delta}{(1+\delta)^{(1+\delta)}}\right)^\mu.$$

   Similarly, if $\mu \leq \mathbb{E}[X]$, then

   $$\text{Prob}[X < (1-\delta)\mu] \quad < \quad e^{-\frac{\delta^2}{2}\mu}$$

   Some of these m_t messages will not succeed in informing new nodes. The reasons could be two-fold: either they reach an already informed node, or multiple messages reach the same uninformed node. The probability that a given message reaches an already informed node is S_{t-1}/n, and the probability of a collision with any other message is at most m_t/n. Hence, the expected number of informed nodes after t iterations is at least

   $$\begin{aligned} \mathbb{E}[S_t] &\geq S_{t-1} + m_t(1 - m_t/n - S_{t-1}/n) \\ &= S_{t-1}(1 + (2 \pm o(1/\log n))(1 - O(1/\log n))) \\ &= S_{t-1}(3 \pm O(1/\log n)). \end{aligned}$$

   In the second step here, we used the fact that $S_{t-1} \leq n/\log n$. Again, we can write the number of successful messages (those that inform a new node) as a sum of $\{0,1\}$ random variables, and apply a Chernoff Bound, showing that the value of S_t will be sharply concentrated (to within a multiplicative $1 \pm 1/\log n$) around its expectation. When all iterations have outcomes close to the expectation (which happens with high probability by a union bound over all these iterations), we reach $n/\log n$ nodes in $\log_3 n + O(\log\log n)$ rounds.

3. **Quadratic Shrinking Phase**: Once $n/\log n$ nodes are reached, each uninformed node has sufficiently high probability of reaching an informed node in its Pull attempt, so we will ignore Push transmissions, and only use Pull in the analysis. We write U_t for the number of uninformed nodes after t iterations. Then, for any node that is uninformed in iteration $t-1$ and makes a uniformly random Pull attempt, the probability that it stays uninformed is U_{t-1}/n. As a result, the expected fraction of uninformed nodes after the t^{th} iteration is $\mathbb{E}[U_t/n] \leq (U_{t-1}/n)^2$, leading to quadratic shrinking. Again, the calls for different nodes are independent, so we can write U_t as a sum of independent 0-1 random variables, and so long as the number of uninformed nodes U_{t-1} is large enough ($U_{t-1} \geq \sqrt{n}\log^4 n$), quadratic shrinking will occur with high probability by Chernoff Bounds.

Because the number of uninformed nodes thus shrinks doubly exponentially, within $O(\log\log n)$ rounds, the number of remaining uninformed nodes is at most $\sqrt{n}\log^4 n$, with high probability.

4. **Finishing**: Finally, once all but $\sqrt{n}\log^4 n$ of the nodes are informed, each node has probability $1-\frac{\log^4 n}{\sqrt{n}}$ of being informed during any one Pull call it makes, and successive rounds are independent. Hence, after three rounds, the probability of any one node not being informed is at most $\frac{\log^{12} n}{n^{3/2}}$, and by a union bound, all nodes are informed after the additional three rounds with probability at least $\frac{\log^{16} n}{n}$.

Taking all this together, we have proved that with high probability, all nodes will learn the message within $\log_3 n + O(\log\log n)$ rounds. To analyze the total number of messages sent, we notice that the startup, quadratic shrinking, and finishing phases take only $O(\log\log n)$ rounds total, so the total number of messages sent in those rounds is $O(n\log\log n)$. During the exponential growth phase, at most $n/\log n$ nodes are active, so no more than $n/\log n$ messages are sent each round, thus no more than $O(n)$ total. As a result, the total message complexity is $O(n\log\log n)$. ■

We should notice that the protocol, as presented here, is very vulnerable to faults. It needs to be run for exactly $\log_3 n + O(\log\log n)$ rounds. If the number of rounds is $\Omega((1+\epsilon)\log_3 n)$, then $\Omega(\epsilon n\log n)$ messages are sent during those extra $\epsilon\log_3 n$ rounds, violating the goal of $O(n\log\log n)$ messages. Conversely, if the number of rounds is $O((1-\epsilon)\log_3 n)$, then with high probability, some nodes will remain uninformed.

Karp et al. [227] show how a more sophisticated "Median Counter" algorithm achieves the same kind of guarantee while being more resilient to failures.

A second question is how low we can push the number of messages sent. Of course, if we do away with gossip, and build a tree instead, then $n-1$ messages suffice to inform all nodes. Even using uniform gossip, we can get away with $n-1$ messages if we are willing to sacrifice the completion time. A protocol achieving this would be one where the initial owner of the message is the only one to transmit it, and does so whenever he calls (or is called by) an uninformed node. This turns the problem into a *coupon collector* problem: the message has reached everyone if everyone has called (or been called by) the initial message holder. This takes $\Theta(n\log n)$ rounds with high probability [300].

Obviously, this type of protocol goes completely against the idea of using the network in spreading a message. Karp et al. show that under "reasonable" restrictions on the protocol, $\omega(n)$ messages are necessary. The first result concerns *address-oblivious* protocols: in each round, a node must decide whether to transmit to its communication partner(s) without knowing their identity. In particular, nodes do not know if the communication partner already has the message. Notice that the Push-Pull protocol we analyzed above is address-oblivious.

Theorem 9.4 ([227]) *Any address-oblivious protocol has to send at least $\Omega(n\log\log n)$ messages to inform all nodes.*

Even without the restriction, lower bounds can be proved, trading off the completion time and message complexity:

Theorem 9.5 ([227]) *Any protocol reaching all but $O(1)$ of the nodes in $O(\log n)$ rounds, and using uniform gossip, must send at least $\omega(n)$ messages.*

An interesting question along the lines of reducing the number of messages sent is how many nodes are informed, and how many rounds it takes, when each node forwards the message k times after receiving it, for some constant k (such as $k = 2$).

9.2 Spreading Multiple Messages

So far, we have seen an analysis for spreading a single message through a network. In reality, this is a simplistic abstraction: typically, there will be many messages originating at different nodes of the network, a fact already discussed by Demers et al. [123].

When each node has a message that is to be shared with all other nodes, we could consider a protocol in which each node always forwards all messages it holds. In that case, we are essentially "super-imposing" the simple gossip protocol for all source messages, so all nodes will learn all messages in $O(\log n)$ rounds with high probability. Unfortunately, the message complexity will be $\Omega(n^2 \log\log n)$. Here and in the next section, we will explore different assumptions/approaches for spreading multiple messages more efficiently.

We first consider the question of how how to efficiently spread k messages to all nodes, assuming that each packet sent can be no larger than one message.

Fernandess and Malkhi [155] gave a protocol carefully choosing which message to forward in each time step. By doing so, they achieve an optimal bound of $O(k + \log n)$ rounds to spread k messages.

An alternative approach was used by Deb, Médard, and Choute [119]: they proposed a protocol based on *network coding* and uniform gossip, where messages are "aggregated" by forming random linear combinations. From these linear combinations, it is then possible to reconstruct the initial messages. The advantage is that nodes can communicate just one aggregated message at a time, and still eventually collect all the necessary information. The analysis of Deb et al. gave an upper bound of $O(k + \sqrt{k}\log(k)\log(n))$ on the number of rounds. This bound was subsequently improved by Häupler [194] to the optimal $O(k + \log n)$ number of rounds. The analysis, which we will cover in this section, is very elegant and general, applying to a wide range of graph models and communication protocols (rather than just uniform gossip).

9.2.1 Network Coding

We begin with a brief introduction of network coding. The idea behind network coding is to combine (or "encode") messages and transmit the encoded messages over the network in a way that the intended recipient can decode them, and the required bandwidth is smaller than it would be for sending each message explicitly. Consider the classic example for network coding in Figure 9.1.

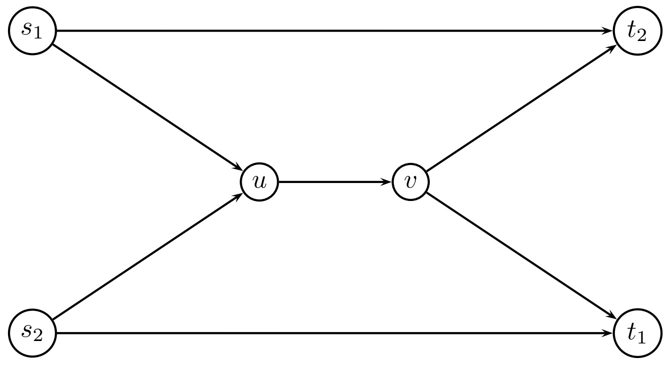


Figure 9.1: A classical example where network coding helps reduce required bandwidth.

The node s_1 is trying to send a message m_1 to t_1, and s_2 is trying to send a message m_2 to t_2. The bottleneck is the edge (u, v), which is required for the transmission of both messages. Initially, it seems as though sending the messages would require two rounds as a result. However, a solution is to compute the bitwise XOR $m_1 \oplus m_2$ at u, and send that to v (and from there to t_1 and t_2). In addition, s_1 can send m_1 to t_2, and s_2 can send m_2 to t_1. These messages appear useless by themselves, but they allow the sinks t_1 and t_2 to decode their intended messages: t_1 takes the XOR of m_2 and $m_1 \oplus m_2$ to obtain m_1; similarly, t_2 computes $m_1 \oplus (m_1 \oplus m_2)$.

9.2.2 Random Linear Network Coding Gossip

The idea of Deb et al. [119] is to combine such linear combinations with uniform gossip. More specifically, nodes compute and forward *random* linear combinations of messages in each round.

To keep the analysis more general, we consider each messages as a string over $\mathcal{F}_q = \{0, 1, \dots, q-1\}$, with addition and multiplication modulo q. q is a prime to ensure that the operations form a field. For intuition, think of $q = 2$, which gives binary strings.

There are k messages $\vec{m}_1, \vec{m}_2, \dots, \vec{m}_k$. Each message $\vec{m}_i$ is a string of ℓ elements of $\mathcal{F}_q$, and can therefore be regarded as a vector $\vec{m}_i \in \mathcal{F}_q^\ell$. (Note that $\mathcal{F}_q^\ell$ is a vector space because $\mathcal{F}_q$ is a field.) In particular, if $\mu_i \in \mathcal{F}_q$ are coefficients, the linear combination $\sum_i \mu_i \vec{m}_i$ is also in $\mathcal{F}_q^\ell$.

Each node that starts out with a message $\vec{m}_i$ knows *which* message it starts out with, i.e., it knows the index i for its message. Each packet sent from one node to another consists of two things: a message $\vec{m} \in \mathcal{F}_q^\ell$, and a coefficient vector $\vec{\mu} \in \mathcal{F}_q^k$. The meaning of the coefficient vector is that $\vec{m} = \sum_i \mu_i \vec{m}_i$.

Now consider a node v. The set of all messages that v has received up to time t together span a subspace X_v (itself a vector space) of the $(k + \ell)$-dimensional vector space $\mathcal{F}_q^{k+\ell}$. For the analysis, we will mostly focus on the space Y_v spanned by only the $\vec{\mu}$ components of the messages that v has received. Initially, the node holding the message $\vec{m}_i$ has $Y_{v,0}$ as the vector space spanned by $\vec{e}_i$ (the vector with 1 in coordinate i and 0 in all other coordinates); all other nodes have $Y_{v,0} = \{\vec{0}\}$.

In each round, node v can use Gram-Schmidt to compute a basis $\vec{b}_1, \vec{b}_2, \dots, \vec{b}_j$ of its space Y_v; along with associated messages $\vec{m'}_1, \dots, \vec{m'}_j$, the $(\vec{b}_i, \vec{m'}_i)$ form a basis of X_v as well. Node v then chooses j i.i.d. uniform elements $\lambda_1, \lambda_2, \dots, \lambda_j$ from $\mathcal{F}_q$, and sends the packet $\sum_i \lambda_i (\vec{b}_i, \vec{m'}_i)$ to another node. (For uniform gossip, this will be a uniform other node; for other communication protocols, it may be chosen from some other distribution.)

After nodes receive packets, they add the messages to their X_v and Y_v spaces for the next round $t + 1$. Once Y_v has full rank k, node v can specifically choose the basis $\vec{b}_1 = \vec{e}_1, \vec{b}_2 = \vec{e}_2, \dots, \vec{b}_k = \vec{e}_k$ and recover the associated messages $\vec{m'}_i = \vec{m}_i$. The key question is then how long it takes until each node's vector space Y_v has full rank k.

9.2.3 Analysis

The analysis of Deb et al.[119] was based on explicitly keeping track of the dimensionality of Y_v, showing that it had to increase with sufficiently high probability in each round. However, this analysis requires dealing with a lot of correlations regarding which specific messages are held by which nodes; it also does not easily generalize to communication protocols other than uniform gossip. The key idea in the improved analysis of Häupler [194] is to focus on the orthogonal complement of Y_v instead of Y_v itself; specifically, to show that the orthogonal complement shrinks sufficiently fast. The key definition is the following:

Definition 9.6 *For a vector $\vec{\mu} \in \mathcal{F}_q^k$, v* knows *about $\vec{\mu}$ iff $\vec{\mu}$ is not orthogonal to Y_v, i.e., if there exists a $\vec{\nu} \in Y_v$ such that $\vec{\mu} \cdot \vec{\nu} \neq 0$.*

Notice that "knowing about" $\vec{\mu}$ does not mean having received a message containing $\vec{\mu}$, or being able to decode a message associated with $\vec{\mu}$. Also, *not knowing* $\vec{\mu}$ does not mean *not* having received $\vec{\mu}$. For instance, if $Y_v = \{(0, 0), (1, 1)\} \in \mathcal{F}_2^2$, and $\vec{\mu} = (1, 1)$, then v actually has explicitly received $\vec{\mu}$, yet the inner product $(1, 1) \cdot (1, 1) = 0$ (and $(0, 0) \cdot (1, 1) = 0$, of course). While the notion of "knowing about" $\vec{\mu}$ is thus not directly interpretable in an immediate sense, it is useful in the following sense:

Lemma 9.7 *If v knows about* every *$\vec{\mu}$, then v can decode all messages.*

Proof. We prove the lemma's contrapositive. If v cannot decode all messages, then Y_v does not have full rank. Using Gram-Schmidt, we can construct a basis of Y_v comprising at most $k - 1$ vectors $\vec{b}_1, \vec{b}_2, \dots, \vec{b}_j$; furthermore, we can complete this basis to a basis of $\mathcal{F}_q^k$. The last vector $\vec{b}_k$ of this basis is by definition orthogonal to Y_v, so v does not know about it. ■

Next, we show that knowledge of vectors $\vec{\mu}$ spreads fast.

Lemma 9.8 *If v knows $\vec{\mu}$ and sends a message to u, then with probability at least $1 - \frac{1}{q}$, u afterwards knows $\vec{mu}$ as well.*

Proof. Let $\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_j$ be an arbitrary basis of Y_v. So v sends u a packet with $\vec{\mu'} = \sum_{i=1}^{j} \lambda_i \vec{b}_i$, where the λ_i are i.i.d. uniform from $\mathcal{F}_q$. Because v knows $\vec{\mu}$, $\vec{\mu}$ is not orthogonal to Y_v; therefore, the inner product $\vec{\mu} \cdot \vec{b}_i \neq 0$ for at least one of the basis vectors $\vec{b}_i$. Without loss of generality, assume that $\vec{\mu} \cdot \vec{b}_j \neq 0$.

Suppose that the λ_i are chosen such that λ_j is chosen last, and let $\alpha = \sum_{i=1}^{j-1} \lambda_i \cdot (\vec{b}_i \cdot \vec{\mu})$. Because $\vec{\mu} \cdot \vec{b}_j \neq 0$, the equation $\alpha + \lambda_j(\vec{\mu} \cdot \vec{b}_j) = 0$ has exactly one solution for λ_j, and this particular value of λ_j is picked with probability $1/q$. ■

Lemma 9.8 implies that knowledge of vectors $\vec{\mu}$ spreads like a message under faulty gossip: with probability $1/q$, the message is not transmitted, but otherwise, the recipient learns it. There are q^k messages that need to be "spread" in this way, and once all of them have reached all nodes, all nodes can decode all messages.

We will take a union bound over all of these q^k messages. In order to succeed in t rounds with probability at least $1-\delta$, we then need that the underlying communication protocol (e.g., uniform gossip) spreads a single message to all nodes in T rounds with probability at least $1 - \delta \cdot q^{-k}$. This is a relatively primitive property that can often be easily analyzed for different communication protocols: for most, the failure probability decreases exponentially over time.

Specifically for uniform gossip, the $1/q$ transmission failure probability in each round slows down the protocol by a factor $\frac{q}{q-1}$. We could redo our earlier analysis for uniform gossip and a single message more carefully, and indeed prove using similar techniques that the failure probability decreases exponentially in the number of rounds. In particular, in order to obtain a failure probability of at most δq^{-k} for a single message then requires $O(\log n + k \log q + \log(1/\delta))$ rounds. Since q is a constant, this gives us the desired bound of $O(k + \log n)$ for spreading k messages.

9.3 Averaging and Sampling using Gossip

A different approach can be used when it is not the individual messages that are of interest to nodes, but some aggregate quantity that the network wants to compute from the messages and spread to all nodes. Then, instead of forwarding each message individually (or using coding), we could perform aggregation within the network. For instance, assume that each node i just holds one number x_i (the number of files, or available memory, or a temperature measurement), and the network wants to compute the average $\bar{x} = \frac{1}{n} \sum_i x_i$ of these numbers. Using a tree, values could be added up (and nodes counted) on the way to the root; the root could then compute the average, and distribute it back to all nodes using the tree. As we discussed above, using a tree is not a very fault-tolerant approach, so we want to use gossip instead.

Our gossip-based algorithm is very simple. Each node i maintains only two values, a *sum* s_i, and a *weight* w_i. The sum is initialized to $s_i := x_i$, and the weight to $w_i := 1$. Then, in each round, each node executes the following protocol Push-Sum [230]:

Algorithm 6 Push-Sum

```
1: Send the pair $(s_i/2, w_i/2)$ to yourself and a uniformly randomly chosen other node.
2: Let $J_i$ be the set of all nodes that have sent a message to $i$ in this round.
3: The new values are $s'_i := s_i/2 + \sum_{j \in J_i} s_j/2$ and $w'_i := w_i/2 + \sum_{j \in J_i} w_j/2$.
4: Keep track of $s_i/w_i$ as approximation of the average.
```

9.3.1 Analysis of Averaging using Gossip

One useful fact that we can notice right away about this protocol (and prove easily using induction) is its property of *mass conservation*: at any point of the execution, we always have that $\sum_i s_i = \sum_i x_i$, and $\sum_i w_i = n$. The sums and weights get redistributed, but not truly changed. We will prove the following theorem about the convergence of Push-Sum to the true average.

Theorem 9.9 *If all x_i are non-negative, then within $O(\log n + \log \frac{1}{\delta} + \log \frac{1}{\epsilon})$ rounds, all estimates s_i/w_i are within $(1 \pm \epsilon)$ of the true average $\bar{x} = \frac{1}{n}\sum_i x_i$, with probability at least $1-\delta$.*

In order to analyze this protocol, we will track what "share" of each node j's number x_j is currently contributing to each node's sum s_i. So we look at a vector $\vec{v}_i$, in which the j^{th} component $v_{i,j}$ denotes the fraction of node j's value that currently is part of node i's sum. This means that initially, we have $v_{i,i} = 1$ for all i, and $v_{i,j} = 0$ for all $i \neq j$. In this view, our protocol can be expressed as follows:

Algorithm 7 Push-Vector

```
1: Send the vector $\frac{1}{2}\vec{v}_i$ to yourself and a uniformly randomly chosen other node.
2: Let $J_i$ be the set of all nodes that have sent a message to $i$ in this round.
3: The new vector is $\vec{v}_i' := \frac{1}{2}\vec{v}_i + \sum_{j \in J_i} \frac{1}{2}\vec{v}_j$.
```

This new version of the protocol traces the old version in the following sense (as is easy to observe, and can be proved by induction on the time steps of the protocol):

Fact 9.10 *At any time during the execution of the protocol, and for any node i, we have that $s_i = \sum_j v_{i,j}x_j$, and $w_i = \sum_j v_{i,j}$.*

Thus, the estimate that node i has is $\frac{\sum_j v_{i,j}x_j}{\sum_j v_{i,j}}$, and we want to show that this quantity converges exponentially fast to $\bar{x}$. One way that we could guarantee convergence would be if all $v_{i,j}$ were equal, and thus equal to $1/n$. In that case, we would have exactly the true average. However, this is clearly too much to hope for: by standard Balls-in-Bins analysis [300], even if they were all equal at some point, one node would be likely to receive many calls (up to $\Omega(\log n)$), and others no call at all, resulting in new values $\Omega(\frac{\log n}{n})$ vs. $\frac{1}{2n}$. However, upon closer inspection, we notice that we do not need quite such a strong condition. It would be enough if, for a fixed i, all $v_{i,j}$ were the same. So we do not need a node's value to be equally distributed among all other nodes; all we need is that a node has equally sized shares of everyone else's values.

This motivates studying how fast the vectors $\vec{v}_i$ converge to multiples of the all-ones vector $\vec{1}$. In order to talk about this convergence, we use $v_{t,i,j}, \vec{v}_{t,i}, s_{t,i}, w_{t,i}$ etc. to denote the values after t iterations of Push-Vector. For ease of notation, we will use the fact that $w_{t,i} = \sum_j v_{t,i,j}$. We then measure the convergence in terms of the error $\Delta_{t,i} = \max_j |\frac{v_{t,i,j}}{w_{t,i}} - \frac{1}{n}|$. We will prove the following two parts, giving the theorem together:

Lemma 9.11 *1. The $\Delta_{t,i}$ converge to 0 exponentially fast.*

2. When the $\Delta_{t,i}$ are small, the estimate of the average is good.

Proof. We first prove the (easier) second part of the lemma. Assume that at some point in time t, the errors $\Delta_{t,i}$ for all i are at most ϵ/n. Then, for each i,

$$\begin{aligned}
\frac{|\frac{\sum_j v_{t,i,j}x_j}{w_{t,i}} - \bar{x}|}{|\bar{x}|} &= n \cdot \frac{|\sum_j (\frac{v_{t,i,j}}{w_{t,i}} - \frac{1}{n})x_j|}{|\sum_j x_j|} \\
&\leq \frac{n}{|\sum_j x_j|} \cdot (\max_j |\frac{v_{t,i,j}}{w_{t,i}} - \frac{1}{n}|) \cdot \sum_j |x_j| \\
&\leq \epsilon,
\end{aligned}$$

where we used the triangle inequality in the numerator for the second step, and the bound on $\Delta_{t,i}$ for the expression in parentheses in the third step. (We also were allowed to cancel the sums over x_j values because all x_j were assumed to be non-negative.) Notice that once we prove exponential convergence below, the time it takes to converge to error ϵ/n is only by an additive $O(\log n)$ larger than to converge to ϵ, so we were free to choose a value of ϵ/n here.

To prove the first part of the lemma, we study a *potential function*, which measures how "close" to converged the system is. We have seen such functions before, for instance in the proof of Theorem 8.7. Here, our potential function will be the sum of variances of the vectors $\vec{v}_{t,i}$. Formally, we define

$$\Phi_t = \sum_{i,j} (v_{t,i,j} - \frac{w_{i,t}}{n})^2.$$

We will show that this potential function decreases exponentially fast, and that a small value for it implies good convergence.

Lemma 9.12 *The conditional expectation of* Φ_t *satisfies* $\mathbb{E}[\Phi_{t+1} \mid \Phi_t = \phi] = (\frac{1}{2} - \frac{1}{2n})\phi$.

Proof. Consider the values $v_{i,j}, w_i$, etc. at time t, and let $f(i)$ denote the random node called by node i in round t. Then, with all random choices known, node i's new vector $\vec{v}_i'$ and weight w_i' are

$$\begin{aligned} \vec{v}_i' &= \frac{1}{2}\vec{v}_i + \frac{1}{2}\sum_{k:f(k)=i} \vec{v}_k, \\ w_i' &= \frac{1}{2}w_i + \frac{1}{2}\sum_{k:f(k)=i} w_k. \end{aligned}$$

Plugging these values into the new potential Φ_{t+1}, we obtain that

$$\begin{aligned} \Phi_{t+1} &= \sum_{i,j} \big(\frac{1}{2}(v_{i,j} - \frac{w_i}{n}) + \frac{1}{2}\sum_{k:f(k)=i}(v_{k,j} - \frac{w_k}{n})\big)^2 \\ &= \frac{1}{4}\sum_{i,j}(v_{i,j} - \frac{w_i}{n})^2 + \frac{1}{4}\sum_{i,j}\sum_{k:f(k)=i}(v_{k,j} - \frac{w_k}{n})^2 + \frac{1}{2}\sum_{i,j}\sum_{k:f(k)=i}(v_{i,j} - \frac{w_i}{n})(v_{k,j} - \frac{w_k}{n}) \\ &\quad + \frac{1}{2}\sum_{i,j}\sum_{k,k':k\neq k', f(k)=f(k')=i}(v_{k,j} - \frac{w_k}{n})(v_{k',j} - \frac{w_{k'}}{n}). \end{aligned}$$

Noticing that in the second sum, the term $(v_{k,j} - \frac{k_i}{n})^2$ appears exactly once for each k (namely for the particular i with $f(k) = i$), we see that the first two sums are precisely equal to $\frac{1}{2}\Phi_t$. Similarly, we can simplify the fourth sum by noticing that each pair k, k' with $f(k) = f(k')$ will appear for exactly one i. So we simplify

$$\begin{aligned} \Phi_{t+1} &= \frac{1}{2}\Phi_t + \frac{1}{2}\sum_{i,j,k}(v_{i,j} - \frac{w_i}{n})(v_{k,j} - \frac{w_k}{n}) \cdot [f(k) = i] \\ &\quad + \frac{1}{2}\sum_{j,k,k':k\neq k'}(v_{k,j} - \frac{w_k}{n})(v_{k',j} - \frac{w_{k'}}{n}) \cdot [f(k) = f(k')]. \end{aligned}$$

Here, we are using Iverson's convention [187]: $[f(k) = i] := \begin{cases} 1 & \text{if } f(k) = i \\ 0 & \text{otherwise} \end{cases}$.

In this form, the expectation of Φ_{t+1} is not too difficult to evaluate: we can use linearity of expectation, and notice that the only terms actually depending on the random choices are $[f(k) = i]$ and $[f(k) = f(k')]$. As they are $\{0, 1\}$ random variables, their expectation is exactly equal to the probability of being 1, which can be easily seen to be $1/n$ for both. (For the first one, this is obvious; for the second one, notice that for any choice of $f(k)$, the probability that $f(k') = f(k)$ is $1/n$, so the same holds overall.) Substituting all of these, we obtain that

$$\begin{aligned}
\mathbb{E}\left[\Phi_{t+1} \mid \Phi_t = \phi\right] &= \frac{1}{2}\phi + \frac{1}{2}\sum_{i,j,k}(v_{i,j} - \frac{w_i}{n})(v_{k,j} - \frac{w_k}{n}) \cdot \text{Prob}[f(k) = i] \\
&\quad + \frac{1}{2}\sum_{j,k,k':k\neq k'}(v_{k,j} - \frac{w_k}{n})(v_{k',j} - \frac{w_{k'}}{n}) \cdot \text{Prob}[f(k) = f(k')] \\
&= \frac{1}{2}\phi + \frac{1}{2n}\sum_{i,j,k}(v_{i,j} - \frac{w_i}{n})(v_{k,j} - \frac{w_k}{n}) + \frac{1}{2n}\sum_{j,k,k':k\neq k'}(v_{k,j} - \frac{w_k}{n})(v_{k',j} - \frac{w_{k'}}{n}) \\
&= \frac{1}{2}\phi + \frac{1}{n}\sum_{i,j,k}(v_{i,j} - \frac{w_i}{n})(v_{k,j} - \frac{w_k}{n}) - \frac{1}{2n}\sum_{j,k}(v_{k,j} - \frac{w_k}{n})^2 \\
&= (\frac{1}{2} - \frac{1}{2n})\phi + \frac{1}{n}\sum_{j}(\sum_{i} v_{i,j} - \sum_{i}\frac{w_i}{n})(\sum_{k} v_{k,j} - \sum_{k}\frac{w_k}{n}) \\
&= (\frac{1}{2} - \frac{1}{2n})\phi.
\end{aligned}$$

In the last step, we used mass conservation, which implied that $\sum_i v_{i,j} = 1$, and $\sum_i w_i = n$, so that the second term actually became 0. Two steps earlier, we made the last sum run over all pairs k, k', and subtracted out the ones we had added in. We also notice that at that point, both sums are equal, so we added them up to form the first one. In summary, this proves the lemma about the conditional expectation. ■

By applying the lemma repeatedly, and using that $\Phi_0 \leq n$, we obtain that $\mathbb{E}\left[\Phi_t\right] \leq n \cdot 2^{-t}$. Thus, after $t = \log n + \log \frac{1}{\hat{\epsilon}}$ rounds, the expected potential is $\mathbb{E}\left[\Phi_t\right] \leq \hat{\epsilon}$. Markov's Inequality [300] states that for any non-negative random variable X and any value a, we have $\text{Prob}[X \geq a] \leq \frac{\mathbb{E}[X]}{a}$. Applying it to Φ_t, and choosing $\hat{\epsilon} = \epsilon^2 \cdot \delta/2 \cdot 2^{-2\tau}$ thus guarantees that with probability at least $1 - \delta/2$, we have $\Phi_t \leq \epsilon^2 \cdot 2^{-2\tau}$. In particular, this bound applies to each term of the sum that constitutes Φ_t, so $|v_{t,i,j} - \frac{w_{t,i}}{n}| \leq \epsilon \cdot 2^{-\tau}$ for all i and j.

At this point, we have almost finished the proof; however, the quantity that we have just proven to be small is not quite the error measure $\Delta_{t,i}$ we are interested in. We still need to divide by $w_{t,i}$ to get exactly our error measure, and $w_{t,i}$ could potentially be quite small. This happens for instance when node i has not received a message from anyone in a while (unlikely, but possible), or when it did receive a message, it was from another node that had not received a message in a while. At this point, we can leverage the earlier analysis of the dissemination of a single message.

Look at all nodes at time $t - \tau$. (Notice that our choice of t implies that $t \geq \tau$, so we are allowed to do that.) At that point, at least one node $\hat{\imath}$ had weight at least 1. Consider the experiment in which this node has a "message", and look at when each node i receives the message. By our previous analysis, and, more specifically, a theorem by Frieze and Grimmett [164], after $\tau = 4\log n + \log \frac{2}{\delta}$ steps, all nodes have received the message after τ steps with probability at least $1 - \delta/2$. Because the weight in a message is at worst halved in each round, and similarly while a node simply "holds" the message, we know that at time t, each node must have weight at least $2^{-\tau}$ with probability at least $1 - \delta/2$. Taking a union bound over both events considered, and dividing the earlier bound by $w_{i,t} \geq 2^{-\tau}$, we obtain that at time t, with probability at least $1 - \delta$, the errors are at most $\Delta_{i,t} = |\frac{v_{t,i,j}}{w_{t,i}} - \frac{1}{n}| \leq \epsilon$.

Finally, a simple inductive proof shows that once the error at all nodes drops below ϵ, it will stay below ϵ at all nodes deterministically, so the quality of the estimate never gets worse.

To complete the proof, we need to verify how large our t is exactly. Substituting the values of $\hat{\epsilon}$ and τ into t, we see that it is $O(\log n + \log \frac{1}{\epsilon} + \log \frac{1}{\delta})$. This completes the proof. ■

9.3.2 Sampling using Gossip

The analysis of the averaging protocol can be easily repurposed to solve another fundamental aggregation problem: random sampling. Each node i holds a set M_i of m_i elements, and the goal is to draw a (nearly) uniform random sample from $M := \bigcup_i M_i$. Here, we assume that all the M_i are disjoint, or alternatively, we consider M as a multiset, and would like elements of higher multiplicity to be drawn with proportionally higher probability. We write $m = \sum_i m_i$.

The protocol Push-Sample is as simple as Push-Sum. Each node i maintains a *sample* q_i (an element of M) and a *weight* w_i. Initially, $w_i = m_i$, and q_i is drawn uniformly at random from M_i. Then, in each round t, each node executes the following:

Algorithm 8 Push-Sample

```
1: Send the pair $(q_i, w_i/2)$ to yourself and a uniformly randomly chosen other node.
2: Let $J_i$ be the set of all nodes that have sent a message to $i$ in this round.
3: Let $w'_i := w_i/2 + \sum_{j \in J_i} w_j/2$.
4: Choose $q'_i$ randomly from the set $\{q_j \mid j \in J_i\}$; specifically, chose $q_j$ with probability $\frac{w_j/2}{w'_i}$.
   (That is, choose the sample from $j$ with probability proportional to $j$'s weight.)
5: Keep track of $q_i$ as the sample.
```

Theorem 9.13 *With probability at least $1-\delta$, within $O(\log n + \log \frac{1}{\delta} + \log \frac{1}{\epsilon}$ rounds, each node i has a sample q_i that is drawn from M with probabilities between $\frac{1-\epsilon}{m}$ and $\frac{1+\epsilon}{m}$.*

Proof. Let $q_{t,i}$ and $w_{t,i}$ denote the sample and weight of node i at step t. We first show by induction that $\text{Prob}[q_{t,i} = q_{0,j}] = \frac{v_{t,i,j} \cdot m_j}{w_{t,i}}$, where $v_{t,i,j}$ are the entries from the Push-Vector protocol in Section 9.3. At time 0, this holds because each node always holds its own sample, and $w_{0,i} = m_i$. (We define $0/0 = 1$.) For the induction step,

$$\text{Prob}[q_{t+1,i} = q_{0,j}] \overset{\text{IH}}{=} \sum_{k \in J_{t,i}} \frac{w_{t,k}/2}{w_{t+1,i}} \cdot \frac{v_{t,k,j} m_j}{w_{t,k}} = \frac{m_j}{w_{t+1,i}} \cdot \sum_{k \in J_{t,i}} \frac{1}{2} v_{t,k,j} = \frac{m_j}{w_{t+1,i}} \cdot v_{t+1,i,j}.$$

Applying Fact 9.10 (with the weights in Push-Sample playing the role of the sums in Push-Sum), we have that $w_{t,i} = \sum_k v_{t,i,k} m_k$. Thus, the probability that node i holds the sample from node j after t rounds is exactly $\frac{v_{t,i,j} m_j}{\sum_k v_{t,i,k} m_k}$. Because the random choice made initially by node j is independent of the execution of the rest of the protocol, and j chooses any particular element from M_j with probability $1/m_j$, the probability for i to hold a particular element from M is exactly $\frac{v_{t,i,j}}{\sum_k v_{t,i,k} m_k}$

We will use the analysis from Section 9.3 to show that this ratio converges to $\frac{1}{m}$ exponentially fast.

By the first part of Lemma 9.11, applied with $\epsilon' = \frac{\epsilon}{(2+\epsilon)n}$, for $t = O(\log n + \log \frac{1}{\epsilon} + \log \frac{1}{\delta})$, with probability at least $1-\delta$, we have $|\frac{v_{t,i,j}}{w_{t,i}} - \frac{1}{n}| \leq \epsilon'$ for all i, j. We assume the high-probability event.

By using the lower bound in the numerator and the upper bound in the denominator, we obtain a lower bound on the probability; similarly, by using the upper bound in the numerator and the lower bound in the denominator, we obtain an upper bound. Thus, we obtain the bounds

$$\frac{1/n - \epsilon'}{m(1/n + \epsilon')} \leq \text{Prob}[q_{t+1,i} = q_{0,j}] \leq \frac{1/n + \epsilon'}{m(1/n - \epsilon')}.$$

The choice of ϵ' guarantees that $1 - \epsilon \leq \frac{1/n - \epsilon'}{1/n + \epsilon'}$ and $\frac{1/n + \epsilon'}{1/n - \epsilon'} \leq 1 + \epsilon$, completing the proof. ■

9.4 Further Reading

A survey covering some of the extensive work on gossip was written by Shah [352].

In our analysis of the time to spread a single message to all nodes, we assumed that the communication graph was complete, i.e., that each node could communicate with each other node. Several papers have worked on generalizing the analysis to the case when there is a given graph structure restricting which node pairs can communicate.

For the simplest uniform gossip protocol (where each informed node simply calls and informs a uniformly random neighbor), Elsässer and Sauerwald [139] give degree-dependent lower bounds of $\Omega(\log n)$ (where the base of the logarithm depends on the degree), and an upper bound of $O(n \log n)$. This matches a lower bound for the star graph obtained via the coupon collector problem, and is thus tight in the worst case.

Naturally, for specific classes of graph, much better bounds than $\Theta(n \log n)$ are possible, even for the simple randomized broadcast protocol. For instance, Feige et al. [153] give tighter bounds for random graphs and hypercubes. Chierichetti et al. [93] use expansion-based analysis techniques to prove an upper bound of $O(\log^2 n)$ for graphs obtained from preferential attachment (see Section 6.1). Mosk-Aoyama and Shah [295] prove an upper bound of essentially $O(\frac{\log n}{\Phi})$ for the time to spread a single message in an arbitrary graph; here, Φ is the conductance of the matrix M determining the call probabilities. For their result, the authors assume that M is double stochastic. Chierichetti et al. [94] extend this result to uniform gossip (which leads to matrices that are not doubly stochastic), and obtain bounds polynomial in $\frac{\log n}{\Phi}$. This bound is based on a breakthrough result on graph sampling by Spielman and Teng [355].

For specific classes of graphs, the lower bounds of Karp et al. [227] can also be strengthened. If the protocol is address oblivious, then for a star, it is easy to see that a message needs at least $\Omega(n \log n)$ rounds to reach all nodes. For $G(n, p)$ random graphs with p sufficiently large, Elsässer [138] shows a lower bound of $\Omega(n \log n / \log \log n)$ on the number of messages required to inform all nodes. Interestingly, address obliviousness is crucial here; if the nodes have just enough memory to remember the last three other nodes they called, Elsässer and Sauerwald [140] show that the total number of messages can be reduced to $O(n \log \log n)$.

So far, our main goal in this context was to spread the message(s) to all nodes quickly. In many natural settings, nodes are embedded geographically, and it is important that nodes close to the source obtain the message first. This may be relevant if the message is an alarm or warning, or if it contains information more useful locally. The problem of spreading a message to nodes at distance d within time poly-logarithmic in d (independently of n) is studied in [232]. There, it is shown that if nodes call each other with non-uniform probability, decreasing polynomially in the distance, then such a poly-logarithmic spreading time is achieved. Both the distribution and analysis bear some similarity to the analysis of greedy routing in Small-World Networks in Section 7.3.

As shown in [232], inversely polynomial gossip distributions can be used as a substrate for more complex types of in-network computation, such as nearest network resources. The problem of resource discovery in networks has also received attention in the context of Peer-to-Peer networks; many different algorithms have been proposed (e.g., [195, 255]). Further protocols that can be implemented with inverse polynomial distributions are discussed in [231], which also proves lower bounds on how well uniform gossip could approximate the same problems.

Analyzing the speed of the diffusion of a message through a network under inversely polynomial distributions in the distance is also closely related to a problem studied in mathematics and physics under the name *long-range percolation* [349]. Here, a graph is generated by embedding the nodes in a metric space (usually, the D-dimensional grid or torus), and generating a certain number of edges per node, with probability d^{-r} of reaching distance d. Thus, the model is essentially identical to the small-world model discussed in Section 7.3. Using techniques similar to [232], Benjamini and Berger [37] prove poly-logarithmic bounds on the diameter of such graphs. Some of these bounds were subsequently improved by Coppersmith et al. [109]. See also the paper by Nguyen and Martel [318] for a discussion.

In our discussions of gossip-based averaging in Section 9.3, we assumed that the communication graph is complete. The problem of averaging (or more general aggregation of data) is also important in networks with communication restricted to neighbors. Depending on the degree of nodes, we could continue to assume that nodes can only communicate with one neighbor, or relax the condition and let nodes communicate with all neighbors simultaneously. For the latter case, [230] shows that by averaging appropriately weighted combinations of the values of all neighbors, a variant of the Push-Sum protocol computes the average, in time equal to the mixing time of a Markov Chain on the graph. Thus, the fastest convergence is achieved by choosing the multipliers so as to have the Markov Chain mix as quickly as possible. This problem has been studied by Boyd et al. [61], who show how to use convex programming to find the fastest mixing Markov Chain for a given graph.

While [230] shows that the mixing time of the fastest Markov Chain on a graph is an upper bound on the time to average using gossip, it does not rule out that much faster protocols could exist. A lower bound is provided by Boyd et al. [63]. Under a slightly more restrictive definition of "gossip algorithm," they show that the mixing time of the fastest Markov Chain on the graph essentially provides a lower bound on any gossip algorithm as well. Consequently, in [62], they study the mixing time of the fastest Markov Chain on random geometric graphs (which are good models for sensor networks), and show a lower bound of $\Theta(1/r^2)$ for nodes with communication range r embedded in a unit square.

Since this bound is fairly slow, and inefficient in terms of requiring $\Omega(n^2)$ rounds of communication, Dimakis et al. [128] propose a protocol called *geographic gossip*. In that protocol, nodes choose a uniformly random point to route to, and then use greedy multi-hop routing to exchange messages with the closest node to that point. Messages are rejected with certain probabilities to avoid oversampling nodes with few nearby other nodes. For this protocol, [128] proves a bound of $O((n \log n)^{1.5})$ on the number of transmissions required. Assuming parallelism and an appropriate communication radius r, this corresponds to $O(1/r^{1.5})$ rounds in the previous model.

Gossip-based algorithms can be used as a substrate for more complex computations beyond message dissemination and averaging. These include computations of spanning trees [231], various sketches of data stored at nodes [230] and eigenvectors and eigenvalues of a matrix held in a distributed way at the nodes [235]. In addition to Astrolabe [370, 371], several system approaches have been proposed using gossip (see, e.g., [218, 217]).

Finally, while we motivated the gossip problem with an eye on its fault-tolerance via randomization, it is an interesting question how quickly a message could be spread over a given arbitrary network, under the constraint that each current message holder can only forward the message to one neighbor in any given round. This *Optimal Broadcast* problem is NP-complete. Ravi [334] gave an $O(\log^2 n)$ approximation, which was subsequently improved by Bar-Noy et al. [29] to $O(\log n)$.

Chapter 10

Peer-to-Peer Systems and Distributed Hashing

In Chapter 7, we saw how to search for a node in a network when there is metric information guiding the search, but the network connectivity is constrained. A similar task arises when the network is used to deliberately store files or other data, as in the case of Peer-to-Peer (P2P) systems. Peer-to-Peer systems have become popular for sharing videos, music, and other types of files; they have also led to interesting research insights combining notions of hashing with network design and search.

Typically, for a variety of reasons (including fault tolerance, legal reasons, scalability, and load balancing), the files should be distributed over the nodes of the network, and those nodes may join or leave the network. The key question is then how to organize the link structure of the network and the location of files so that files are easy and fast to find.

Some of the real-world systems that implemented some of the ideas we will discuss are Freenet [162], Gnutella [175] and BitTorrent [101]. Our focus will be primarily on research approaches with provable guarantees, including Chord [359], CAN [333], Pastry [342], Tapestry [393], and Viceroy [272].

In order to divide items between nodes of the network such that (1) the load is roughly balanced across nodes, and (2) items can be easily retrieved, the key idea is to consider node names and item names as embedded in a common space, and have nodes store items whose ID is close to their own. Since it is a priori very plausible that both types of ID could be highly clustered (defeating the goal of load balancing), instead of the actual node and item IDs, we use hashes. In keeping with the type of analysis typically performed for hashing, we treat the output of these hash functions as independently random. (Details about the effects of limited independence can typically be worked out with additional calculations.) As such, the general approach is based on, and shares significant commonalities with, the idea of *consistent hashing* [224].

In addition to storing the *primary* copy, for fault tolerance reason, P2P systems typically store additional copies at other nodes. Depending on the context and the importance for theoretical properties, we will sometimes discuss those in the sequel as well.

In order to *find* a file stored at a node, the node that is looking for a file generates the hash of the ID[1] it is looking for, and routes it through the network. To facilitate this routing, in addition to the physical network, the P2P network contains an *overlay network* of edges along which file requests are forwarded. The exact nature of this overlay network is different in different architectures, and will be the central part of our analysis below.

In the rest of this chapter, we let N denote the number nodes, and M (an upper bound on) the number of files.

[1]The assumption here is that a list of files that are available in the network is easily accessible (e.g., because the list is much shorter than the actual files — which could be, e.g., movies — or that a suitable ID can be easily generated from the information that the searching node *does* have about the file it needs.

10.1 Chord

The first — and perhaps conceptually cleanest — architecture we will analyze is the Chord system [359]. In the Chord system, we think of all IDs (hashes of file and node names) as m-bit strings, which we can interpret as numbers base 2. Here, $m = \Omega(\max(\log N, \log M))$. We then consider a cycle that embeds these m-bit strings in increasing order, with wraparound $\mod 2^m$. Each node is in charge of storing all files whose IDs lie below its own ID and that of the next node on the cycle in clockwise order. For example, in Figure 10.1, node A (with a coordinate of 00000) is in charge of storing all objects with IDs 00000, 00001, and 00010.

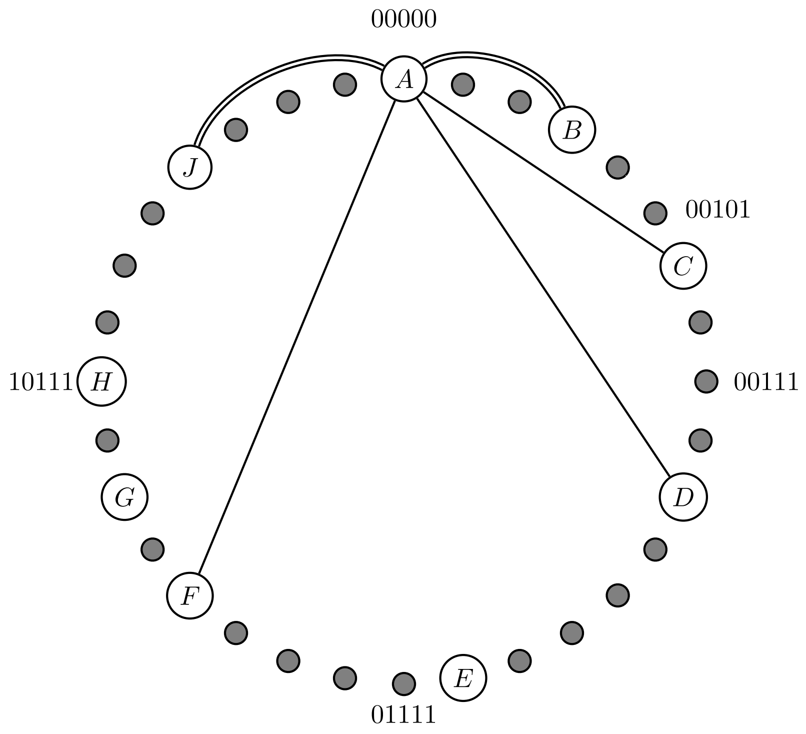


Figure 10.1: An illustration of the Chord architecture.

Each node also knows the ID of the next clockwise node (its *successor node*) for the purpose of routing file requests, and the ID of its *predecessor node* (next node counterclockwise); in Figure 10.1, these edges are shown (for node A) as double lines Based on the successor pointers, all requests could be routed, but the number of hops would be linear in the number of nodes. For that reason, in addition to the successor and predecessor pointers, each node also stores $\Theta(m)$ *shortcut links.* For a node with ID x, the shortcut link at *level* $\ell = 1, 2, \ldots, m$ goes to the next node after $(x + 2^\ell) \mod 2^m$ on the cycle. We call this node the *level-ℓ neighbor of* x, and denote it by $N_\ell(x)$. In Figure 10.1, these edges are shown for node A as single lines (to node C, D, F).

The routing protocol is now quite simple. If node x wants to route to the node that holds file y, it forwards the request along the longest shortcut link that does not overshoot y. In other words, it routes along the level ℓ link such that $N_\ell(x) \leq y < N_{\ell+1}(x)$, where inequalities are taken to be suitably modulo 2^m.

To see why this approach can with high probability route in $O(m)$ steps, consider some ID y that node x is trying to route to. Let ℓ be such that $x + 2^{\ell-1} \leq y < x + 2^\ell$. If $N/2^{m-\ell} = \Omega(m)$ (i.e., there are enough nodes such that an interval of length 2^ℓ contains $\Omega(m)$ nodes in expectation), then by standard occupancy bounds, with sufficiently high probability, there is at least one node in the interval $[\frac{x+y}{2}, y]$, meaning that the routing process will halve the distance to y, which can happen at most m times. On the other hand, if $N/2^{m-\ell} = O(m)$, then in expectation, only $O(m)$ nodes should lie between x and the destination. By Chernoff Bounds, with high probability, the *actual* number of nodes between x and y will be $O(m)$ as well, so even without using any shortcut links at higher levels, the routing will succeed in $O(m)$ hops.

So Chord is a construction of distributed hash tables using routing tables of size $O(m)$ and routing in $O(m)$ hops.

To insert a new node into the system, the node first determines its (hash) ID x, and then tries to route to a "file" with ID x. By doing so, it will find its predecessor x' node on the cycle, who can inform it of its successor. It can thus easily insert itself into the cycle, and obtain files from x' that it will henceforth host.

Notice that the approach here is quite similar to the maintenance of a randomized skip list data structure.

10.2 Viceroy

If we want to avoid the logarithmic degrees of $\Theta(m)$ in the Chord construction, we can use some insights we gained in Chapter 7 on small world graphs, combined with ideas underlying Butterfly Graphs and skips lists. We will see how the Viceroy [272] approach reduces the degree to $O(1)$. To see why small world models can be useful to guide our intuition, notice that the graphs we studied in Chapter 7 had constant degree, and long-range links were equally likely (probability $\Omega(1/\log n)$) to be at any "scale," in the following sense: with probability $\Omega(1/\log n)$, the link was such that following it (at least) halved the distance to the destination. This is very similar to how our analysis sketch for Chord in Section 10.1 worked.

The second motivation for the Viceroy approach is the Butterfly Graph architecture [264]. Butterfly graphs are layered graphs with n nodes in each layer, $\log n$ layers, and constant node degrees. In layer i, the edges to layer $i+1$ correspond to flipping bit i in the binary representation of the node ID. (See Figure 10.2.) Butterfly graphs are a useful topology for routing and have been used in the design of parallel algorithms and architectures. Notice that by contracting all layers into one, one exactly obtains a hypercube.

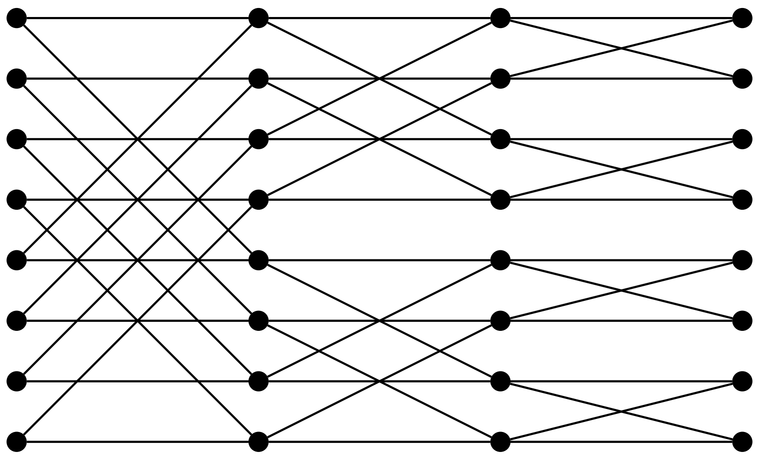


Figure 10.2: The butterfly graph on $n = 8$ nodes.

To take advantage of the insights/motivations discussed above, the main added feature of Viceroy compared to Chord is that each node x chooses a uniformly random *level* (or *label*) $\ell_x \in \{0, 1, \ldots, m-1\}$. Then, instead of having long-distance links at each level, it will only have one link at level ℓ_x. More specifically, this level-ℓ_x edge will go to the next node y at level $\ell_y = \ell_x - 1$ after position $(x + 2^{\ell_x}) \mod 2^m$ on the cycle. In addition to this one level-ℓ_x edge, node x also maintains edges to the closest (clockwise) level-$(\ell_x - 1)$ and level-$(\ell_x + 1)$ nodes (if they exist), as well as its predecessor and successor. We call the first two of these the *level change edges*. Thus, each node has outdegree (and thus routing table size) at most 5. The key useful property of this construction is that the next node at any level ℓ can be found fast:

Lemma 10.1 *Let x be any node, and ℓ any level. Finding the next (clockwise) level-ℓ node x' from x takes $O(m)$ steps with high probability.*

Proof. Because levels are i.i.d. uniform from $\{0, \ldots, m-1\}$, simply following successor pointers, the waiting time to find a node of level ℓ is distributed as a geometric random variable with mean m, and thus sharply concentrated. ■

To search for a key y starting from node x, the first step is to find the closest level-$(m-1)$ node x' clockwise after x. By Lemma 10.1, this takes $O(m)$ steps with high probability. Then, for each step, when at some node x, follow the level-ℓ_x edge, unless it would overshoot y by more than $O(m)$, in which case follow the level change edge to level $\ell_x - 1$. If even the level change edge would overshoot y, then simply follow successor edges until y is found. Notice that in either the level-ℓ_x edge case or the level change case, the next step always takes us to a level-$(\ell_x - 1)$ node.

We show that this routing approach maintains the invariant that when the routing process is at a level-ℓ node x, then $y - x < 2^{\ell+1}$. The proof is by induction, the base case being trivial. First, if even the level change edge would overshoot y, that means that there is no node of level $\ell - 1$ between x and y. Then, with high probability, y is within $O(m)$ steps of x, so using successor edges takes $O(m)$ steps. Similarly, if the level-ℓ edge overshoots y, but only by $O(m)$, then after following the edge, it takes at most $O(m)$ steps along predecessor edges to find y. Otherwise, we consider two cases.

- If $y - x \geq 2^\ell$, then by Lemma 10.1, with high probability, the level-ℓ edge does not overshoot y by more than $O(m)$. The case of overshooting by $O(m)$ was covered above; otherwise, taking the level-ℓ edge reduces the distance by at least 2^ℓ, and the induction hypothesis holds.
- If $y - x < 2^\ell$, then the induction hypothesis holds even without taking an edge, and since taking an edge further reduces the distance, it holds after taking the level change edge.

10.3 CAN

The CAN (Content Addressable Network) architecture was proposed in [333]. At a high level, it uses a similar idea to Chord. However, rather than thinking of the nodes as forming essentially a hypercube of dimension $m = \Omega(\log M)$ (see the discussion of contracting the Butterfly Graph), CAN embeds nodes into a more low-dimensional space. This allows for faster routing, at the cost of larger routing tables. The main argument in favor of CAN is that each hop of routing in a large decentralized system will take significant time, while routing tables are stored locally, so their size is secondary. In particular, instead of the $(O(\log M), O(\log M))$ tradeoff of Chord (or the $(O(1), O(\log M))$ tradeoff of Viceroy) for (routing table, path length), CAN argues for a tradeoff of $(O(\text{poly}(M), O(1)))$, so long as $\text{poly}(M)$ grows sufficiently slowly.

As mentioned above, the space into which all IDs are hashed is the d-dimensional space $\{0, 1, \ldots, M^{1/d}\}^d$. Each node is again assigned a range of items, which is a hyper-rectangle in this grid. For a simple illustration in $d = 2$ dimensions, see Figure 10.3. In this figure, for example, node A is in charge of the square $[0, 3]^2$ of document IDs, while G is in charge of the rectangle $[4, 8] \times [6, 8]$. Note that some IDs (such as $(4, 4)$ or $(0, 7)$) are stored at multiple nodes.

Each node maintains a routing table with all addresses that differ from its own coordinates in exactly one coordinate. For instance, in Figure 10.3, the routing table for node A contains the node B (for IDs of the form $[0, 3] \times [4, 7]$), the node C (for IDs of the form $[0, 3] \times [7, 8]$), the node D (for IDs of the form $[4, 6] \times [0, 3]$), the node E (for IDs of the form $[7, 8] \times [0, 3]$), and the node F (for IDs $(4, 3), (5, 3), (6, 3)$). In general, the routing table has size at most $O(d \cdot M^{2/d})$ (because for each of d dimensions, a node has at most $M^{1/d}$ coordinates in its range; and for each of those, there are at most $M^{1/d}$ other coordinate entries for which a routing entry needs to be stored. Typically, the size will be more like $O(d \cdot M^{1/d})$.

To route a request for a file with a given d-dimensional ID, nodes correct the coordinates one by one from their own ID(s). For example, in Figure 10.3, suppose that node A were receiving a request for $(6, 6)$. It might route it to node B (since B is in charge of — say — $(0, 6)$ or $(2, 6)$), to node D (because D is in

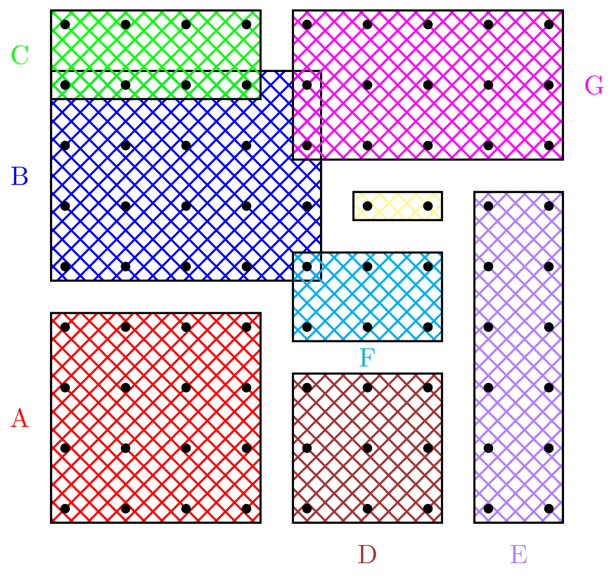


Figure 10.3: An illustration of the CAN architecture with $d = 2$ dimensions.

charge of $(6, 0)$, or $(6, 2)$), or to node F (because F is in charge of $(6, 3)$). Those nodes would then all have G in their routing tables for the coordinate $(6, 6)$, and be able to forward the request directly to G. Generally, in d dimensions, it takes at most d steps to fix all digits, meaning that routing takes $O(d)$ steps.

As mentioned above, the approach advocated by CAN is to choose d small (say, $d = 2$ or $d = 3$), and incur larger routing tables to be able to route in few steps. For instance, for $d = 3$, even a network with 10^{12} nodes only needs routing tables of size about 30000, while allowing routing in at most 3 hops.

10.4 Finding local copies

So far, the goal of the constructions we analyzed was to achieve a good tradeoff between the degrees of nodes (size of the routing tables) and the number of hope that it would take to find a particular object. Beyond the *number* of hops, an additional consideration is the total *length* of the path that the request for an object traverses. This objective assumes that the nodes live in some metric space, and is motivated if we associate routing latency with distance in the metric space. Roughly speaking, the goal is as follows: if a node v has a copy of the object it needs at some distance r, then the total length of all the hops that the request traverses should be $O(r)$. We consider an approach due to Plaxton, Rajaraman, and Richa [327].

10.4.1 The construction

The space is defined by a metric distance function $d(v, v')$ defined on node (or ID) pairs. We assume that the metric space grows *polynomially*, so that each ball $B(v, r)$ of radius r around a node v contains $O(r^\alpha)$ nodes, for some constant α. Polynomial growth is satisfied, for instance, if the nodes are embedded in some α-dimensional Euclidean space.

The high-level idea of the construction is similar to the one for Chord we saw in Section 10.1. As with Chord, we consider node names x_v to be strings of length $L = O(\log N)$, written using digits $\Sigma = \{0, 1, \ldots, b - 1\}$, for some suitable value of b to be determined later. (For now, think of $b = 2$.)

For strings x, y (IDs of nodes or files), we say that x and y *match* in the first i digits if $x_{i'} = y_{i'}$ for all $i' \leq i$. We write $V_{v,i,j} = \{u \mid x_u$ and x_v match in the first $i - 1$ digits, and digit i of x_u equals $j.\}$. Then, we let $X_{v,i,j}$ contain the c closest nodes to v (with respect to d) from $V_{v,i,j}$, or all of $V_{v,i,j}$, if it contains at most c nodes, for a constant c we determine later. In other words, $X_{v,i,j}$ contains the c closest nodes to v whose ID matches that of v in the first $i-1$ digits, and has j in its i^{th} digit (while later digits $i' > i$ can be arbitrary). Thus, the routing tables have size $O(bcL)$. Whenever a node v' is in a routing table for node v, it also maintains a *back pointer*, i.e., each node is aware of all routing tables it is a part of.

Each file (with ID y) is stored at (all) nodes x matching the longest prefix of y; for much of the analysis, we will pretend that there is a node with ID exactly y, or that there exists a unique node with the longest prefix match. (We will revisit this issue in Section 10.5.) In addition, if node v stores a file y, it will also store a copy of the file at each neighbor in any of the $X_{v,i,j}$; thus, each file is stored at $O(bcL)$ nodes.

Suppose that a node v with ID x is looking for a file with ID y. First, x checks with all of its $O(bcL)$ neighbors if they have the file; if so, the request is routed to that neighbor. Otherwise, suppose that x differs from y in digit i, and that y has a value of j in digit i. Then, x forwards the request to the closest node (in terms of the distances d) in $X_{v,i,j}$; that node continues in the same way.

10.4.2 Analysis

In our analysis, we will be imprecise in a number of ways. In particular, we will usually just consider expectations of random variables, and pretend that the variables are equal (or close enough to) their expectations. Similarly, we will pretend that high-probability events always happen, and make a number of similar simplifications. A detailed analysis, keeping track of all cases and making tail bounds explicit, would be significantly more technical, and can be found in [327].

As a first observation, notice that the distance from a node v to neighbors matching the first i digits of its ID increases exponentially in i.

Lemma 10.2 *The closest node to v whose ID matches that of v in the first i digits is at distance about $O(b^{i/\alpha})$. The c^{th} closest node to v whose ID matches that of v in the first i digits is at distance about $O((cb^i)^{1/\alpha})$.*

Proof Sketch. The expected fraction of nodes whose ID agrees with x in the first i digits is b^{-i}. The locations of nodes (in the metric space) and their IDs are independent. Thus, to collect at least c such nodes for $X_{v,i,j}$, in expectation, one needs to consider the closest cb^i nodes. Those nodes — assuming that the metric space is roughly uniform — will be at distance $O((cb^i)^{1/\alpha})$. ■

Consider a file request with ID y, and a current node v with ID x. Let $\hat{v}$ be the node closest to v (in terms of the metric space) holding a copy of the file with ID y, and let $\hat{x}$ be its ID. Let i be such that the smallest ball containing v and $\hat{v}$ contains approximately b^i nodes. Thus, the distance between v and $\hat{v}$ is approximately $b^{i/\alpha}$.

We assume that x and $\hat{x}$ match in the first $i-1$ digits, and use j to denote the i^{th} digit of y (and thus also of $\hat{x}$). By Lemma 10.2, the lengths of the hops in routing increase exponentially in the hop number. Therefore, at the cost of a constant factor, we can disregard all the routing steps until the request reached v, and focus only on the step originating with v.

Let v_i be the closest neighbor of v whose ID matches that of $\hat{v}$ in the first i digits. Recall that x and $\hat{x}$ match in the first $i-1$ digits, so $v_i \in X_{v,i,j}$. By Lemma 10.2, the distance between v and v_i is about $b^{(i-1)/\alpha}$.

Next, we want to argue that v_i is also in the routing table of $\hat{v}$, and thus holds a copy of the request with ID y. (Recall that a node which is assigned a file also stores it at each neighbor.) Because the ID of v_i matches with $\hat{x}$ for (at least) the first i digits, it is a candidate for $X_{\hat{v},i',j'}$ for some $i' \geq i+1$ and some j'. It will be included in such a set unless there are at least c candidates that are closer to $\hat{v}$ than v_i.

By triangle inequality, the distance from $\hat{v}$ to v_i is at most the distance from $\hat{v}$ to v plus the distance from v to v_i, so at most $b^{i/\alpha} + b^{(i-1)/\alpha} \leq b^{(i+1)/\alpha}$. The ball $B(\hat{v}, b^{(i+1)/\alpha})$ contains about $\Theta(b^{i+1})$ nodes. Of these, in expectation about $\Theta(b^{i+1} \cdot b^{-i}) = \Theta(b)$ match the first i digits of $\hat{x}$, and with high probability, at most $\Theta(b^2)$ do. So with $c = \Omega(b^2)$, with sufficiently high probability, v_i will indeed be a neighbor of $\hat{v}$.

10.5 Tapestry, and Dealing with Node Sparseness

For the analysis in Section 10.4, we assumed that for each file ID y, there was a unique node to store y at. The simplest view of this assumption is when there is a unique node v whose ID $x_v = y$; more generally, the analysis applies when there is a unique node with longest matching prefix. (The reason is that for the analysis, we can simply pretend that all IDs are truncated at the corresponding digit.)

To avoid ambiguity about which node holds a file, we can use an approach similar to Chord or Probing in hashing: consider an ID $y = y_1 y_2 y_3 \cdots y_L$. When inserting the file with ID, first route to fix (at least) one digit at a time, as described in Section 10.4. When some digit y_i is not matched by any neighbor of the current node v, try replacing the i^{th} digit with $y_i + 1, y_i + 2, \ldots$ until a match is made. Route to the corresponding node and continue routing from there. This approach essentially routes y to the next larger ID $x \geq y$ that is actually present in the system. However, doing so makes maintaining routing tables non-trivial, which is the focus of the Tapestry [393] architecture, and outlined in this section.

For any string β (of length at most L, the maximum number of digits), we say that there is a *hole* at β if no node name starts with β. We say that a prefix β of length i is *sparse* if there is at least one digit x_{i+1} such that the concatenation $beta \cdot x_{i+1}$ is a hole.

Consider a prefix β of length i, such that t nodes have IDs starting with β. Then, for any particular digit x_{i+1}, the probability at least one node has prefix $\beta \cdot x_{i+1}$ is at least $1 - (1 - 1/b)^t$. When $t \geq b \log N$, this probability is at least $1 - 1/N$. So if β is a sparse prefix, then with high probability, at most $b \log N$ nodes have a prefix of β. Therefore, at an overhead of at most $O(b \log N)$, node v can augment its routing table with all node v' with prefix β, for every sparse prefix β of its node ID x.

Now consider a new node v with ID x which is inserting itself. Suppose that v fails at some node v' with ID x'. Say that it managed to match a prefix β of i digits (which is common between x and x'), but the routing table of v' contains no node with ID prefix $\beta \cdot x_{i+1}$. Then, β is sparse, so with high probability, there are only $O(b \log N)$ nodes total with prefix β, and they are all stored in the routing table of v', whence v can copy them. For shorter prefixes, v can use routing tables it saw along the path to v'. We now specify in more detail how the routing tables are assembled.

To motivate the subsequent approach, assume that the metric d has growth bounded by γ, so that $|B(v, 2r)| \leq \gamma |B(v, r)|$. In choosing the base b of the node IDs, we will ensure that $b > \gamma^2$.

Let $c = \Theta(\log N)$ be the number of closest neighbors included in $X_{v,i,j}$. Let B_i be the smallest ball around v containing at least k nodes u whose prefix matches x in (at least) the first i digits. Let δ_i be the radius of B_i. We first show that δ_i grows fast as a function of i.

Lemma 10.3 $\delta_{i+1} > 4\delta_i$.

Proof Sketch. As in Lemma 10.2, we use that roughly a $1/b^i$ fraction of nodes match the first i digits of x. Thus, a ball containing k matching nodes will contain about $|B_i| \approx k b^i$ nodes. A ball with radius $4\delta_i$ contains at most $\gamma^2 k b^i < k b^{i+1}$ nodes (because we chose $b > \gamma^2$). Thus, a ball containing at least k nodes matching $i + 1$ digits must have radius more than $4\delta_i$. ■

While it takes a significantly larger radius to have k nodes matching $i + 1$ digits with v, with high probability, at least *one* node in B_i does match $i + 1$ digits with v:

Lemma 10.4 *With high probability, B_i contains at least one node matching the first i digits of x.*

Proof Sketch. The probability that no node in B_i matches $i + 1$ digits of x is at most $(1 - b^{-(i+1)})^{k \cdot b^i} \leq \mathrm{e}^{-k/b}$, which is small because b is a constant while $k = \Theta(\log N)$. ■

Return to our node v trying to insert itself. Let v' be a node at distance at most δ_i from v, whose ID x' matches x in the first i digits, but not in digit $i + 1$. This means that v' should really be in the routing table for v (because it is among the c closest prefix matches), so we need to show that v will learn about v'.

By Lemma 10.4, there is a node u' at distance at most δ_i from v that shares a prefix of length $i + 1$ with v. By triangle inequality, u' is at distance at most $2\delta_i$ from v'. Among all nodes that share a prefix of length

$i+1$ with v, let u be the one closest to v'. Because u' is a candidate, the distance from v' to u is at most $2\delta_i$, so the distance from v to u is at most $d(v,u) \leq d(v,v') + d(v',u) \leq 2\delta_i + \delta_i = 3\delta_i < \delta_{i+1}$, by Lemma 10.3. Therefore, we get that $u \in B_{i+1}$.

Because $u \in B_{i+1}$, v has learned about u, simply from copying routing tables as it routes the ID x. And because u is the closest node to v' among all nodes with ID prefix $x_1 x_2 \cdots x_i x_{i+1}$ (it matches the first i digits with v'), u must be in $X_{v',i,x_{i+1}}$; in particular, v' has u in its routing table, and because we also maintain back pointers for routing tables, u knows about v' as well.

The high-level idea of the insertion protocol at a node v with ID x is now the following:

- Routing tables are assembled from larger i to smaller i.
- For each of the $c = \Theta(\log N)$ closest nodes matching the first $i+1$ digits of x, collect their c nearest neighbors matching the first i digits of x.
- Form $V_{v,i,j} = \bigcup_{v'} X_{v',i,j}$, i.e., merge all of the routing tables from these nodes.
- Let $X_{v,i,j}$ contain the c nodes of $V_{v,i,j,}$ closest to v.
- Continue to the next smaller i, by obtaining routing tables from the nodes in $X_{v,i,j}$.

Bibliography

[1] Emmanuel Abbe, Afonso S. Bandeira, and Georgina Hall. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory*, 62(1):471–487, 2016.

[2] Dimitris Achlioptas. Lower bounds for random 3-SAT via differential equations. *Theoretical Computer Science*, 265(1/2):159–185, 2001.

[3] Dimitris Achlioptas, Aaron Clauset, David Kempe, and Cris Moore. On the bias of traceroute sampling, or, power-law degree distributions in regular graphs. *Journal of the ACM*, 56(4), 2009.

[4] Dimitris Achlioptas, Amos Fiat, Anna Karlin, and Frank McSherry. Web search via hub synthesis. In *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, pages 500–509, 2001.

[5] Gaurav Agarwal and David Kempe. Modularity-maximizing graph communities via mathematical programming. *Europ. Physics Journal B*, 66(3), 2008.

[6] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proc. 13th ACM SIGMOD Intl. Conference on Management of Data*, pages 207–216, 1993.

[7] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proc. 20th Intl. Conf. on Very Large Data Bases*, pages 487–499, 1994.

[8] Ravindra K. Ahuja, Dorit Hochbaum, and James B. Orlin. Solving the convex cost integer dual of minimum cost network flow problem. *Management Science*, 49(7):950–964, 2003.

[9] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows.* Prentice Hall, 1993.

[10] William Aiello, Fan Chung, and Linyuan Lu. Random evolution of massive graphs. In James Abello, Panos M. Pardalos, and Mauricio G. C. Resende, editors, *Handbook of Massive Data Sets*, pages 97–122. Kluwer, 2002.

[11] Nir Ailon. Aggregation of partial rankings, p-ratings and top-m lists. In *Proc. 18th ACM-SIAM Symp. on Discrete Algorithms*, pages 415–424, 2007.

[12] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM*, 55(5):1–27, 2008.

[13] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.

[14] Noga Alon and Vitali D. Milman. λ_1, isoperimetric inequalities for graphs and superconcentrators. *J. Combinatorial Theory, Ser. B*, 38:73–88, 1985.

[15] Brian S. Amento, Loren G. Terveen, and William C. Hill. Does “authority” mean quality? Predicting expert quality ratings of web documents. In *Proc. 23rd Intl. Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 296–303, 2000.

[16] Reid Andersen and Fan R. K. Chung. Detecting sharp drops in pagerank and a simplified local partitioning algorithm. In *2nd Intl. Conf. on Theory and Applications of Models of Computation (TAMC)*, pages 1–12, 2007.

[17] Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. Using pagerank to locally partition a graph. *Internet Mathematics*, 4(1):35–64, 2007.

[18] Dana Angluin, James Aspnes, and Lev Reyzin. Inferring social networks from outbreaks. In *Proc. 20th Intl. Conf. on Algorithmic Learning Theory (ALT 2010)*, pages 104–118, 2010.

[19] Arvind Arasu, Jasmine Novak, Andrew Tomkins, and John Tomlin. Pagerank computation and the structure of the web: Experiments and algorithms. In *11th Intl. World Wide Web Conference*, 2002.

[20] Kenneth Arrow. *Social Choice and Individual Values.* Wiley, 1951.

[21] David Arthur, Rajeev Motwani, Aneesh Sharma, and Ying Xu. Pricing strategies for viral marketing on social networks. In *Proc. 5th Workshop on Internet and Network Economics (WINE)*, pages 101–112, 2009.

[22] Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. In *Proc. 33rd ACM Symp. on Theory of Computing*, 2001.

[23] Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. Greedily finding a dense subgraph. *Journal of Algorithms*, 34, 2000.

[24] Chalee Asavathiratham. *The Influence Model: A Tractable Representation for the Dynamics of Networked Markov Chains.* PhD thesis, Massachusetts Institute of Technology, 2000.

[25] Chalee Asavathiratham, Sandip Roy, Bernard C. Lesieutre, and George C. Verghese. The influence model. *IEEE Control Systems*, pages 52–64, 12 2001.

[26] Yossi Azar, Amos Fiat, Anna Karlin, Frank McSherry, and Jared Saia. Spectral analysis of data. In *Proc. 33rd ACM Symp. on Theory of Computing*, pages 619–626, 2001.

[27] Norman T. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications.* Hafner Press, 1975.

[28] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56:89–113, 2004.

[29] Amotz Bar-Noy, Sudipto Guha, Joseph Naor, and Baruch Schieber. Message multicasting in heterogeneous networks. *SIAM Journal on Computing*, 30:347–358, 2001.

[30] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[31] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Mean-field theory for scale-free random networks. *Physica A*, 272:173–187, 1999.

[32] Salvador Barberà. An introduction to strategy-proof social choice functions. *Social Choice and Welfare*, 18:619–653, 2001.

[33] Salvador Barberà, Faruk Gul, and Ennio Stacchetti. Generalized median voter schemes and committees. *Journal of Economic Theory*, 61:262–289, 1993.

[34] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proc. 37th IEEE Symp. on Foundations of Computer Science*, pages 184–193, 1996.

[35] Yair Bartal. On approximating arbitrary metrices by tree metrics. In *Proc. 30th ACM Symp. on Theory of Computing*, pages 161–168, 1998.

[36] Frank M. Bass. A new product growth model for consumer durables. *Management Science*, 15:215–227, 1969.

[37] Itai Benjamini and Noam Berger. The diameter of long-range percolation clusters on finite cycles. *Random Structures and Algorithms*, 19(2):102–111, 2001.

[38] Eli Berger. Dynamic monopolies of constant size. *Journal of Combinatorial Theory Series B*, 83:191–200, 2001.

[39] Noam Berger, Béla Bollobás, Christian Borgs, Jennifer T. Chayes, and Oliver Riordan. Degree distribution of the FKP network model. In *Proc. 30th Intl. Colloq. on Automata, Languages and Programming*, pages 725–738, 2003.

[40] Tim Berners-Lee. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web.* Collins, 2000.

[41] Julian Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36:192–236, 1974.

[42] Krishna Bharat, Andrei Broder, Monika Henzinger, Puneet Kumar, and Suresh Venkatasubramanian. The connectivity server: fast access to linkage information on the web. *Computer Networks*, 30:469–477, 1998.

[43] Krishna Bharat, Bay-Wei Chang, Monika Henzinger, and Matthias Ruhl. Who links to whom: Mining linkage between web sites. In *Proc. 1st Intl. Conf. on Data Mining*, pages 51–58, 2001.

[44] Krishna Bharat and Monika Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proc. 21st Intl. Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 104–111, 1998.

[45] Shishir Bharathi, David Kempe, and Mahyar Salek. Competitive influence maximization in social networks. In *Proc. 3rd Workshop on Internet and Network Economics (WINE)*, pages 306–311, 2007.

[46] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. Detecting high log-densities — an $O(n^{1/4})$ approximation for densest k-subgraph. arXiv:1001.2891.

[47] Monica Bianchini, Marco Gori, and Franco Scarselli. Inside pagerank. *ACM Transactions on Internet Technology*, 5:92–128, 2005.

[48] Norman L. Biggs. *Algebraic Graph Theory.* Cambridge University Press, 1994.

[49] Duncan Black. On the rationale of group decision making. *J. Political Economy*, 56:23–34, 1948.

[50] Duncan Black. *The Theory of Committees and Elections.* Cambridge University Press, 1958.

[51] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proc. 18th Intl. Conf. on Machine Learning*, pages 19–26, 2001.

[52] Lawrence E. Blume. The statistical mechanics of strategic interaction. *Games and Economic Behavior*, 5(3):387–424, 1993.

[53] Béla Bollobás. *Modern Graph Theory.* Springer, first edition, 1998.

[54] Béla Bollobás. *Extremal Graph Theory.* Dover, second edition, 2004.

[55] Phillip Bonacich. Power and centrality: A family of measures. *American Journal of Sociology*, 92(5):1170–1182, 1987.

[56] Christian Borgs, Michael Brautbar, Jennifer Chayes, and Brendan Lucier. Maximizing social influence in nearly optimal time. In *Proc. 25th ACM-SIAM Symp. on Discrete Algorithms*, pages 946–957, 2014.

[57] Allan Borodin, Yuval Filmus, and Joel Oren. Threshold models for competitive influence in social networks. In *Proc. 6th Workshop on Internet and Network Economics (WINE)*, pages 539–550, 2010.

[58] Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, and Panayiotis Tsaparas. Finding authorities and hubs from link structures on the world wide web. In *10th Intl. World Wide Web Conference*, 2001.

[59] Allan Borodin, Gareth O. Roberts, Jeffrey S. Rosenthal, and Panayiotis Tsaparas. Link analysis ranking: Algorithms, theory and experimets. *ACM Transactions on Internet Technologies (TOIT)*, 5(1), 2005.

[60] Justin Boyan, Dane Freitag, and Thorsten Joachims. A machine learning architecture for optimizing web search engines. In *AAAI Workshop on Internet Based Information Systems*, 1996.

[61] Stephen Boyd, Persi Diaconis, and Lin Xiao. Fastest mixing markov chain on a graph. *SIAM Review*, 46:667–689, 2004.

[62] Stephen Boyd, Aarpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Mixing times for random walks on geometric random graphs. In *Proc. 2nd SIAM Workshop on Analytic Algorithms and Combinatorics*, 2005.

[63] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Gossip algorithms: Design, analysis and applications. In *Proc. 24th IEEE INFOCOM Conference*, pages 1653–1664, 2005.

[64] Yuri Boykov, Olga Veksler, and Ramin Zabih. Markov random fields with efficient approximations. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 648–655, 1998.

[65] Yuri Boykov, Olga Veksler, and Ramin Zabih. A new algorithm for energy minimization with discontinuities. In *International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, 1999.

[66] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[67] Ulrik Brandes and Thomas Erlebach, editors. *Network Analysis*. Springer, 2005.

[68] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30:107–17, 1998.

[69] Andrei Broder, Robert Krauthgamer, and Michael Mitzenmacher. Improved classification via connectivity information. In *Proc. 11th ACM-SIAM Symp. on Discrete Algorithms*, pages 576–585, 2000.

[70] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. In *9th Intl. World Wide Web Conference*, 2000.

[71] Anna D. Broido and Aaron Clauset. Scale-free networks are rare. https://arxiv.org/abs/1801.03400, 2018.

[72] Jacqueline J. Brown and Peter H. Reinegen. Social ties and word-of-mouth referral behavior. *Journal of Consumer Research*, 14(3):350–362, 1987.

[73] Ronald S. Burt. *Structural Holes : The Social Structure of Competition.* Harvard University Press, 1995.

[74] Ronald S. Burt. The network structure of social capital. In B. Staw and R. Sutton, editors, *Research in Organizational Behavior*, pages 345–423. JAI Press/Elsevier, 2000.

[75] Vannevar Bush. As we may think. *The Atlantic Monthly*, pages 101–108, July 1945.

[76] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[77] Jean Carlson and John C. Doyle. Highly optimized tolerance: A mechanism for power laws in designed systems. *Physical Review E*, 60:1412–1427, 1999.

[78] Tim Carnes, Chandrashekar Nagarajan, Stefan M. Wild, and Anke van Zuylen. Maximizing influence in a competitive social network: A follower's perspective. In *Proc. Intl. Conf. on Electronic Commerce (ICEC)*, pages 351–360, 2007.

[79] Soumen Chakrabarti, Byron Dom, David Gibson, Jon Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Mining the link structure of the world wide web. *IEEE Computer*, 32:60–67, 1999.

[80] Soumen Chakrabarti, Byron Dom, and Piotr Indyk. Enhanced hypertext categorization using hyperlinks. In *Proc. 18th ACM SIGMOD Intl. Conference on Management of Data*, pages 307–318, 1998.

[81] Soumen Chakrabarti, Mukul Joshi, and Vivek Tawde. Enhanced topic distillation using text, markup tags, and hyperlinks. In *Proc. 24th Intl. Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 208–216, 2001.

[82] David G. Champernowne. A model of income distributions. *Economic Journal*, 63:318–351, 1953.

[83] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning.* MIT Press, 2006.

[84] Moses Charikar. Greedy approximation algorithms for finding dense components in graphs. In *Proc. 3rd Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2000.

[85] Moses Charikar, Venkat Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, pages 360–383, 2005.

[86] Moses Charikar and Anthony Wirth. Maximizing quadratic programs: Extending grothendieck's inequality. In *Proc. 45th IEEE Symp. on Foundations of Computer Science*, pages 54–60, 2004.

[87] Chandra Chekuri, Sanjeev Khanna, Joseph Naor, and Leonid Zosin. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms*, pages 109–118, 2001.

[88] Ning Chen. On the approximability of influence in social networks. *SIAM Journal on Discrete Mathematics*, 23(3):1400–1415, 2009.

[89] Qian Chen, Hyunseok Chang, Ramesh Govindan, Sugih Jamin, Scott J. Shenker, and Walter Willinger. The origin of power laws in Internet topologies revisited. In *Proc. 21st ACM SIGCOMM Conference*, 2002.

[90] Wei Chen, Laks V.S. Lakshmanan, and Carlos Castillo. *Information and Influence Propagation in Social Networks.* Synthesis Lectures on Data Management. Morgan & Claypool, 2013.

[91] Wei Chen, Yifei Yuan, and Li Zhang. Scalable influence maximization in social networks under the linear threshold model. In *Proc. 10th Intl. Conf. on Data Mining*, pages 88–97, 2010.

[92] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Math. Statistics*, 23:493–509, 1952.

[93] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Rumor spreading in social networks. In *Proc. 36th Intl. Colloq. on Automata, Languages and Programming*, pages 375–386, 2009.

[94] Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi. Rumour spreading and graph conductance. In *Proc. 21st ACM-SIAM Symp. on Discrete Algorithms*, 2010. to appear.

[95] Fan Chung. *Spectral Graph Theory.* American Mathematical Society, 1997.

[96] Vašek Chvátal. *Linear Programming.* Freeman, 1983.

[97] Aaron Clauset and Cris Moore. Accuracy and scaling phenomena in Internet mapping. *Physical Review Letters*, 94:018701, 2005.

[98] Aaron Clauset, Mark Newman, and Cris Moore. Finding community structure in very large networks. *Physical Review E*, 70, 2004.

[99] Aaron Clauset, Cosma R. Shalizi, and Mark Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[100] Peter Clifford and Aidan Sudbury. A model for spatial conflict. *Biometrika*, 60(3):581–588, 1973.

[101] Bram Cohen. Incentives build robustness in bittorrent. In *Proc. 1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[102] William W. Cohen, Robert E. Shapire, and Yoram Singer. Learning to order things. *J. of Artificial Intelligence Research*, 10:243–270, 1999.

[103] David A. Cohn and Huan Chang. Learning to probabilistically identify authoritative documents. In *Proc. 17th Intl. Conf. on Machine Learning*, pages 167–174, 2000.

[104] David A. Cohn and Thomas Hofmann. The missing link — a probabilistic model of document content and hypertext connectivity. In *Proc. 15th Advances in Neural Information Processing Systems*, pages 430–436, 2001.

[105] James S. Coleman, Herbert Menzel, and Elihu Katz. *Medical Innovations: A Diffusion Study.* Bobbs Merrill, 1966.

[106] Vincent Conitzer. Anonymity-proof voting rules. In *Proc. 4th Workshop on Internet and Network Economics (WINE)*, pages 295–306, 2008.

[107] Brian Conrad and Michael Mitzenmacher. Power laws for monkeys typing randomly: the case of unequal probabilities. *IEEE Transactions on Information Theory*, 50(7):1403–1414, 2004.

[108] World Wide Web Consortium. A little history of the world wide web, 1945-1995, 2000. available at http://www.w3.org/History.html.

[109] Don Coppersmith, David Gamarnik, and Maxim Sviridenko. The diameter of a long-range percolation graph. *Random Structures and Algorithms*, 21(1):1–13, 2002.

[110] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, second edition, 2001.

[111] Gérard Cornuéjols, Marshall L. Fisher, and George L. Nemhauser. Location of bank accounts to optimize float. *Management Science*, 23:789–810, 1977.

[112] Leon Danon, Jordi Duch, Albert Diaz-Guilera, and Alex Arenas. Comparing community structure identification. *J. Stat. Mech.*, 2005.

[113] Abhimanyu Das and David Kempe. Sensor selection for minimizing worst-case prediction error. In *Proc. 7th Intl. Symp. on Information Processing in Sensor Networks*, pages 45–54, 2008.

[114] Brian D. Davison. Recognizing nepotistic links on the web. In *Artificial Intelligence for Web Search*, pages 23–28, 2000.

[115] Brian D. Davison. Topical locality in the web. In *Proc. 23rd Intl. Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 272–279, 2000.

[116] Jean-Charles de Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royale des Sciences, Paris*, pages 657–665, 1784.

[117] M. J. A. Nicolas de Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix.* Imprimerie Royale, Paris, 1785.

[118] Jeffrey Dean and Monika Henzinger. Finding related web pages in the world wide web. *Computer Networks*, 31:1467–1479, 1999.

[119] Supratim Deb, Muriel Médard, and Clifford Choute. Algebraic gossip: a network coding approach to optimal multiple rumor mongering. *IEEE Transactions on Information Theory*, 52(6):2486–2507, 2006.

[120] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

[121] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *J. of the American Society for Information Sciences*, 41:391–407, 1990.

[122] Maria Deijfen and Willemien Kets. Random intersection graphs with tunable degree distribution and clustering. *Probab. Eng. Inform. Sci*, to appear.

[123] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott J. Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proc. 6th ACM Symp. on Principles of Distributed Computing*, pages 1–12, 1987.

[124] Persi Diaconis and Ronald L. Graham. Spearman's footrule as a measure of disarray. *Journal of the Royal Statistical Society, Series B*, 39:262–268, 1977.

[125] Odo Diekmann and J. A. P. Heesterbeek. *Mathematical Epidemiology of Infectious Diseases : Model Building, Analysis and Interpretation.* Wiley, 2000.

[126] Reinhard Diestel. *Graph Theory.* Springer, second edition, 2000.

[127] Stephen Dill, Ravi Kumar, Kevin McCurley, Sridhar Rajagopalan, D. Sivakumar, and Andrew Tomkins. Self-similarity in the web. *ACM Transactions on Internet Technology*, 2(3):205–223, 2002.

[128] Alexandros G. Dimakis, Anand D. Sarwate, and Martin J. Wainwright. Geographic gossip: Efficient aggregation for sensor networks. In *Proc. 5th Intl. Symp. on Information Processing in Sensor Networks*, pages 69–76, 2006.

[129] Pedro Domingos and Matthew Richardson. Mining the network value of customers. In *Proc. 7th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 57–66, 2001.

[130] Pedro Domingos and Matthew Richardson. The intelligent surfer: Probabilistic combination of link and content information in pagerank. In *Proc. 16th Advances in Neural Information Processing Systems*, pages 1441–1448, 2002.

[131] David L. Donoho. High-dimensional data analysis: The curses and blessings of dimensionality. Notes to accompany a lecture at the AMS Conference on Mathematical Challenges of the 21st Century, 2000.

[132] Nan Du, Yingyu Liang, Maria-Florina Balcan, and Le Song. Influence function learning in information diffusion networks. In *Proc. 31st Intl. Conf. on Machine Learning*, pages 2016–2024, 2014.

[133] Nan Du, Le Song, Song Yuan, and Alex J. Smola. Learning Networks of Heterogeneous Influence. In *Proc. 26th Advances in Neural Information Processing Systems*, pages 2780–2788, 2012.

[134] Pradeep Dubey, Rahul Garg, and Bernard de Meyer. Competing for customers in a social network: The quasi-linear case. In *Proc. 2nd Workshop on Internet and Network Economics (WINE)*, pages 162–173, 2006.

[135] Jordi Duch and Alex Arenas. Community detection in complex networks using extremal optimization. *Physical Review E*, 72, 2005.

[136] Cynthia Dwork, Ravi Kumar, Moni Naor, and D. Sivakumar. Rank aggregation methods for the web. In *10th Intl. World Wide Web Conference*, 2001.

[137] Glenn Ellison. Learning, local interaction, and coordination. *Econometrica*, 61(5):1047–1071, 1993.

[138] Robert Elsässer. On the communication complexity of randomized broadcasting in random-like graphs. In *Proc. 18th ACM Symp. on Parallel Algorithms and Architectures*, pages 148–157, 2006.

[139] Robert Elsässer and Thomas Sauerwald. On the runtime and robustness of randomized broadcasting. In *Proc. 17th Intl. Symp. on Algorithms and Computation*, pages 349–358, 2006.

[140] Robert Elsässer and Thomas Sauerwald. The power of memory in randomized broadcasting. In *Proc. 19th ACM-SIAM Symp. on Discrete Algorithms*, pages 218–227, 2008.

[141] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 5:17–61, 1960.

[142] Jean-Baptiste Estoup. *Gammes Sténographiques*. Institut Sténographique de France, 1916.

[143] Guy Even, Joseph Naor, Satish Rao, and Baruch Schieber. Divide-and-conquer approximation algorithms via spreading metrics. In *Proc. 36th IEEE Symp. on Foundations of Computer Science*, pages 62–71, 1995.

[144] Eyal Even-Dar and Asaf Shapira. A note on maximizing the spread of influence in social networks. In *Proc. 3rd Workshop on Internet and Network Economics (WINE)*, pages 281–286, 2007.

[145] Alex Fabrikant, Elias Koutsoupias, and Christos Papadimitriou. Heuristically optimized trade-offs: A new paradigm for power laws in the Internet. In *Proc. 29th Intl. Colloq. on Automata, Languages and Programming*, pages 110–122, 2002.

[146] Ronald Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, and Erik Vee. Comparing and aggregating rankings with ties. *SIAM Journal on Discrete Mathematics*, 20(3):628–648, 2006.

[147] Ronald Fagin, Ravi Kumar, and D. Sivakumar. Comparing top k lists. *SIAM Journal on Discrete Mathematics*, 17:134–160, 2003.

[148] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *Journal of Computer and System Sciences*, 69(3):485–497, 2004.

[149] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the Internet topology. In *Proc. 18th ACM SIGCOMM Conference*, pages 251–262, 1999.

[150] Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45:634–652, 1998.

[151] Uriel Feige, Guy Kortsarz, and David Peleg. The dense k-subgraph problem. *Algorithmica*, 29:410–421, 2001.

[152] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. In *Proc. 48th IEEE Symp. on Foundations of Computer Science*, pages 461–471, 2007.

[153] Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal. Randomized broadcast in networks. *Random Structures and Algorithms*, 1:447–460, 1990.

[154] Uriel Feige and Michael Seltser. On the densest k-subgraph problem. Technical report, The Weizmann Institute, Rehovot, 1997.

[155] Yaacov Fernandess and Dahlia Malkhi. On collaborative content distribution using multi-message gossip. *J. Parallel Distrib. Comput*, 67(12):1232–1239, 2007.

[156] Gary W. Flake, Steve Lawrence, C. Lee Giles, and Frans Coetzee. Self-organization of the web and identification of communities. *IEEE Computer*, 35, 2002.

[157] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486:75–174, 2010. Eprint arXiv: 0906.0612.

[158] Santo Fortunato and Mark Barthélemy. Resolution limit in community detection. *Proc. Natl. Acad. Sci. USA*, 104(1):36–41, 2007.

[159] Santo Fortunato and Claudio Castellano. Community structure in graphs. In R. Meyers, editor, *Encyclopedia of Complexity and System Science*. Springer, 2009. Eprint arXiv:0712.2716.

[160] Pierre Fraigniaud. Small worlds as navigable augmented networks: Model, analysis, and validation. In *Proc. 15th European Symp. on Algorithms*, pages 2–11, 2007.

[161] Ove Frank and David Strauss. Markov graphs. *Journal of the American Statistical Association*, 81:832–842, 1986.

[162] Freenet. http://freenet.sourceforge.net.

[163] Yoav Freund and Robert E. Shapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[164] Alan Frieze and Geoffrey Grimmett. The shortest-path problem for graphs with random arc-lengths. *Discrete Applied Mathematics*, 10:57–77, 1985.

[165] Mark E. Frisse. Searching for information in a hypertext medical handbook. *Communications of the ACM*, 31(7):880–886, 1988.

[166] Marco Gaertler, Robert Görke, and Dorothea Wagner. Significance-driven graph clustering. In *Proc. 3rd Intl. Conf. on Algorithmic Aspects in Information and Management*, pages 11–26, 2007.

[167] Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. Using ghost edges for classification in sparsely labeled networks. In *Proc. 14th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 256–264, 2008.

[168] Giorgio Gallo, Michael D. Grigoriadis, and Robert E. Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18:30–55, 1989.

[169] Lise Getoor and Benjamin Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[170] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The Google file system. *ACM SIGOPS Operating Systems Review*, 37(5):29–43, 2003.

[171] Alan F. Gibbard. Manipulation of voting schemes: a general result. *Econometrica*, 41(4):587–601, 1973.

[172] Alan F. Gibbard. Manipulation of voting schemes that mix voting with chance. *Econometrica*, 45(3):665–681, 1977.

[173] David Gibson, Jon Kleinberg, and Prabhakar Raghavan. Inferring web communities from link topology. In *Proc. 9th ACM Conf. on Hypertext and Hypermedia*, 1998.

[174] Yuval Ginosar and Ron Holzman. The majority action on infinite graphs: strings and puppets. *Discrete Mathematics*, 215:59–71, 2000.

[175] Gnutella. http://www.gnutella.com.

[176] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. *Journal of the ACM*, 35(4):921–940, 1988.

[177] Jacob Goldenberg, Barak Libai, and Eitan Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12:211–223, 2001.

[178] Jacob Goldenberg, Barak Libai, and Eitan Muller. Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata. *Academy of Marketing Science Review*, 2001.

[179] Eric A. Goles and Jorge Olivos. Periodic behavior of generalized threshold functions. *Discrete Mathematics*, 30(2):187–189, 1980.

[180] Gene Golub and Charles van Loan. *Matrix Computations*. Johns Hopkins University Press, third edition, 1996.

[181] Manuel Gomez-Rodriguez, David Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *Proc. 28th Intl. Conf. on Machine Learning*, pages 561–568, 2011.

[182] Manuel Gomez-Rodriguez, Jure Leskovec, and Andrease Krause. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):21, 2012.

[183] Ralph E. Gomory and Tien Chung Hu. Multi-terminal network flows. *J. SIAM*, 9, 1961.

[184] Amit Goyal, Francesco Bonchi, and Laks V. S. Lakshmanan. Learning influence probabilities in social networks. In *Proc. 3rd ACM Intl. Conf. on Web Search and Data Mining*, pages 241–250, 2010.

[185] Sanjeev Goyal and Michael Kearns. Competitive contagion in networks. In *Proc. 44th ACM Symp. on Theory of Computing*, pages 759–774, 2012.

[186] Michel Grabisch, Antoine Mandel, Agnieszka Rusinowska, and Emily Tanimura. Strategic influence in social networks. *Mathematics of Operations Research*, forthcoming, 2017.

[187] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics - A foundation for Computer Science*. Addison Wesley, second edition, 1994.

[188] Mark Granovetter. The strength of weak ties. *American Journal of Sociology*, 78:1360–1380, 1973.

[189] Mark Granovetter. Threshold models of collective behavior. *American Journal of Sociology*, 83:1420–1443, 1978.

[190] D. Greig, B. Porteous, and Allan H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society, Series B*, 51(2):271–279, 1989.

[191] Roger Guimerà and Luis Amaral. Cartography of complex networks: modules and universal roles. *J. Stat. Mech.*, pages 1–13, 2005. P02001.

[192] Roger Guimerà, Marta Sales-Pardo, and Luis Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70, 2004. 025101.

[193] Anupam Gupta and Eva Tardos. A constant factor approximation algorithm for a class of classification problems. In *Proc. 32nd ACM Symp. on Theory of Computing*, pages 652–658, 2000.

[194] Bernhard Haeupler. Analyzing network coding (gossip) made easy. *Journal of the ACM*, 63(3):26:1–26:22, 2016.

[195] Mor Harchol-Balter, F. Tom Leighton, and Daniel Lewin. Resource discovery in distributed networks. In *Proc. 18th ACM Symp. on Principles of Distributed Computing*, 1999.

[196] Jason D. Hartline, Vahab S. Mirrokni, and Mukund Sundararajan. Optimal marketing strategies over social networks. In *17th Intl. World Wide Web Conference*, pages 189–198, 2008.

[197] Taher Haveliwala. Topic-sensitive pagerank. In *11th Intl. World Wide Web Conference*, 2002.

[198] Ara Hayrapetyan, David Kempe, Martin Pál, and Zoya Svitkina. Unbalanced graph cuts. In *Proc. 13th European Symp. on Algorithms*, pages 191–202, 2005.

[199] Jing He, John E. Hopcroft, Hongyu Liang, Supasorn Suwajanakorn, and Liaoruo Wang. Detecting the structure of social networks using (α, β)-communities. In *8th Workshop on Algorithms and Models for the Web Graph (WAW)*, pages 26–37, 2011.

[200] Xinran He and David Kempe. Price of anarchy for the n-player competitive cascade game with submodular activation functions. In *Proc. 9th Conference on Web and Internet Economics (WINE)*, pages 232–248, 2013.

[201] Dorit Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems.* PWS Publishing, 1997.

[202] Dorit Hochbaum. An efficient algorithm for image segmentation, markov random fields and related problems. *Journal of the ACM*, 48(2):686–701, 2001.

[203] Dorit Hochbaum. Ranking sports teams and the inverse equal paths problem. In *Proc. 2nd Workshop on Internet and Network Economics (WINE)*, pages 307–318, 2006.

[204] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[205] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proc. 22nd Intl. Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 50–57, 1999.

[206] Paul W. Holland, Kathryn B. Laskey, and Samuel Leinhardt. Stochastic blockmodels: Some first steps. *Social Networks*, 5:109–137, 1983.

[207] Richard A. Holley and Thomas M. Liggett. Ergodic theorems for weakly interacting infinite systems and the voter model. *Annals of Probability*, 3:643–663, 1975.

[208] Roger A. Horn and Charles R. Johnson. *Matrix Analysis.* Cambridge University Press, 1990.

[209] Charles H. Hubbell. An input-output approach to clique identification. *Sociometry*, 28(4):377–399, 1965.

[210] Bernardo A. Huberman and Lada A. Adamic. Growth dynamics of the world wide web. *Nature*, page 399, 1999.

[211] Nicole Immorlica, Jon Kleinberg, Mohammad Mahdian, and Tom Wexler. The role of compatibility in the diffusion of technologies through social networks. In *Proc. 8th ACM Conf. on Electronic Commerce*, pages 75–83, 2007.

[212] Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, 48(4):761–777, 2001.

[213] Matthew O. Jackson. *Social and Economic Networks.* Princeton University Press, 2008.

[214] Matthew O. Jackson and Leeat Yariv. Diffusion on social networks. *Economie Publique*, 16(1):69–82, 2006.

[215] Matthew O. Jackson and Leeat Yariv. Diffusion of behavior and equilibrium properties in network games. *American Economic Review*, 97(2):92–98, 2007.

[216] Rinku Jain and S. Ramakumar. Stochastic dynamics modeling of the protein sequence length distribution in genomes: implications for microbial evolution. *Physica*, 273:476–485, 1999.

[217] Márk Jelasity, Alberto Montresor, and Özalp Babaoglu. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252, 2005.

[218] Márk Jelasity and Özalp Babaoglu. T-man: Gossip-based overlay topology management. In *Proc. Engineering Self-Organising Applications*, 2005.

[219] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proc. 8th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 133–142, 2002.

[220] Paul A. Dreyer Jr. and Fred S. Roberts. Irreversible k-threshold processes: Graph-theoretical threshold models of the spread of disease and of opinion. *Discrete Applied Mathematics*, 157(7):1615–1627, 2009.

[221] Gil Kalai. A fourier-theoretic perspective for the condorcet paradox and arrow's theorem. *Adv. in Appl. Math.*, 29:412–426, 2002.

[222] Gil Kalai. Noise sensitivity and chaos in social choice theory. Technical Report Discussion Paper Series dp 399, Center for Rationality and Interactive Design Theory, Hebrew University, 2005.

[223] Michihiro Kandori, George J. Mailath, and Rafael Rob. Learning, mutation, and long run equilibria in games. *Econometrica*, 61(1):29–56, 1993.

[224] David R. Karger, Eric Lehman, F. Thomson Leighton, Rina Panigrahy, Matthew S. Levine, and Daniel Lewin. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proc. 29th ACM Symp. on Theory of Computing*, pages 654–663, 1997.

[225] Howard Karloff. *Linear Programming.* Birkhäuser, 1991.

[226] Michal Karoński, Edward R. Scheinerman, and Karen B. Singer-Cohen. On random intersection graphs: the subgraph problem. *Combinatorics, Probability and Computing*, 8:131–159, 1999.

[227] Richard Karp, Christian Schindelhauer, Scott J. Shenker, and Berthold Vöcking. Randomized rumor spreading. In *Proc. 41st IEEE Symp. on Foundations of Computer Science*, pages 565–574, 2000.

[228] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[229] John Kemeny. Mathematics without numbers. *Daidalos*, 88:571–591, 1959.

[230] David Kempe, Alin Dobra, and Johannes Gehrke. Computing aggregate information using gossip. In *Proc. 44th IEEE Symp. on Foundations of Computer Science*, pages 482–491, 2003.

[231] David Kempe and Jon Kleinberg. Protocols and impossibility results for gossip-based communication mechanisms. In *Proc. 43rd IEEE Symp. on Foundations of Computer Science*, pages 471–480, 2002.

[232] David Kempe, Jon Kleinberg, and Alan Demers. Spatial gossip and resource location protocols. *Journal of the ACM*, 51:943–967, 2005.

[233] David Kempe, Jon Kleinberg, and Eva Tardos. Influential nodes in a diffusion model for social networks. In *Proc. 32nd Intl. Colloq. on Automata, Languages and Programming*, pages 1127–1138, 2005.

[234] David Kempe, Jon Kleinberg, and Eva Tardos. Maximizing the spread of influence in a social network. *Theory of Computing*, 11(4):105–147, 2015.

[235] David Kempe and Frank McSherry. A decentralized algorithm for spectral analysis. *Journal of Computer and System Sciences*, 74:70–83, 2008.

[236] Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *Proc. 39th ACM Symp. on Theory of Computing*, pages 95–103, 2007.

[237] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proc. 34th ACM Symp. on Theory of Computing*, pages 767–775, 2002.

[238] Subhash Khot. Ruling out PTAS for graph min-bisection, densest subgraph and bipartite clique. In *Proc. 45th IEEE Symp. on Foundations of Computer Science*, pages 136–145, 2004.

[239] Peter D. Killworth and H. Russell Bernard. The reverse small-world experiment. *Social Networks*, 1(2):159–192, 1978.

[240] Jon Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46:604–632, 1999.

[241] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. 32nd ACM Symp. on Theory of Computing*, pages 163–170, 2000.

[242] Jon Kleinberg. Small-world phenomena and the dynamics of information. In *Proc. 15th Advances in Neural Information Processing Systems*, pages 431–438, 2001.

[243] Jon Kleinberg. Complex networks and decentralized search algorithms. In *Proc. Intl. Congress of Mathematicians (ICM)*, 2006.

[244] Jon Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. The web as a graph: Measurements, models and methods. In *International Conference on Combinatorics and Computing*, 1999.

[245] Jon Kleinberg and Eva Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric partitioning and markov random fields. *Journal of the ACM*, 49(5):616–639, 2002.

[246] Jon Kleinberg and Eva Tardos. *Algorithm Design.* Addison-Wesley, 2005.

[247] Judith S. Kleinfeld. Could it be a big world after all? the 'six degrees of separation' myth. *Society*, 2002.

[248] Isabel M. Kloumann, Johan Ugander, and Jon M. Kleinberg. Block models and personalized pagerank. *Proc. Natl. Acad. Sci. USA*, page 201611275, 2016.

[249] Andreas Krause and Carlos Guestrin. Optimal nonmyopic value of information in graphical models — efficient algorithms and theoretical limits. In *Proc. 19th Intl. Joint Conf. on Artificial Intelligence*, pages 1339–1345, 2005.

[250] Andreas Krause and Carlos Guestrin. Near-optimal observation selection using submodular functions. In *Proc. 22nd AAAI Conf. on Artificial Intelligence*, pages 1650–1654, 2007.

[251] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. In *Proc. 5th Intl. Symp. on Information Processing in Sensor Networks*, pages 2–10, 2006.

[252] Andreas Krause, Ram Rajagopal, Anupam Gupta, and Carlos Guestrin. Simultaneous placement and scheduling of sensors. In *Proc. 8th Intl. Symp. on Information Processing in Sensor Networks*, 2009.

[253] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, D. Sivakumar, Andrew Tomkins, and Eli Upfal. Stochastic models for the web graph. In *Proc. 41st IEEE Symp. on Foundations of Computer Science*, pages 57–65, 2000.

[254] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. In *8th Intl. World Wide Web Conference*, 1999.

[255] Shay Kutten, David Peleg, and Uzi Vishkin. Deterministic resource discovery in distributed networks. *Theory Comput. Syst.*, 36(5):479–495, 2003.

[256] Tsz Chiu Kwok, Lap Chi Lau, Yin Tat Lee, Shayan Oveis Gharan, and Luca Trevisan. Improved cheeger's inequality: Analysis of spectral partitioning algorithms through higher order spectral gap. In *Proc. 45th ACM Symp. on Theory of Computing*, pages 11–20, 2013.

[257] Anukool Lakhina, John W. Byers, Mark E. Crovella, and Peng Xie. Sampling biases in IP topology measurements. In *Proc. 22nd IEEE INFOCOM Conference*, 2003.

[258] Amy N. Langville and Carl D. Meyer. Deeper inside PageRank. *Internet Mathematics*, 1(3):335–380, 2004.

[259] Amy N. Langville and Carl D. Meyer. *Google's PageRank and Beyond: The Science of Search Engine Rankings.* Princeton University Press, 2006.

[260] Emmanuelle Lebhar and Nicolas Schabanel. Close to optimal decentralized routing in long-range contact networks. *Theoretical Computer Science*, 348(2–3):294–310, 2005.

[261] James R. Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order Cheeger inequalities. *Journal of the ACM*, 61(6):37, 2014.

[262] Jon Lee, Vahab S. Mirrokni, Viswanath Nagarajan, and Maxim Sviridenko. Non-monotone submodular maximization under matroid and knapsack constraints. In *Proc. 41st ACM Symp. on Theory of Computing*, pages 323–332, 2009.

[263] Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. In *Proc. 12th Intl. Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2009.

[264] F. Thomson Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes.* Morgan Kaufmann, first edition, 1991.

[265] Ronny Lempel and Shlomo Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In *9th Intl. World Wide Web Conference*, 2000.

[266] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1):2, 2007.

[267] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie S. Glance. Cost-effective outbreak detection in networks. In *Proc. 13th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 420–429, 2007.

[268] Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

[269] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[270] Sofus A. Macskassy. Improving learning in networked data by combining explicit and mined links. In *Proc. 22nd AAAI Conf. on Artificial Intelligence*, pages 590–595, 2007.

[271] Vijay Mahajan, Eitan Muller, and Frank M. Bass. New product diffusion models in marketing: A review and directions for research. *Journal of Marketing*, 54(1):1–26, 1990.

[272] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: a scalable and dynamic emulation of the butterfly. In *Proc. 21st ACM Symp. on Principles of Distributed Computing*, pages 183–192, 2002.

[273] Benoît B. Mandelbrot. An informational theory of the statistical structure of languages. In W. Jackson, editor, *Communication Theory*, pages 486–502. Academic Press, 1953.

[274] Benoît B. Mandelbrot. A multi-fractal walk down Wall Street. *Scientific American*, 1999.

[275] Gurmeet S. Manku, Moni Naor, and Udi Wieder. Know thy neighbor's neighbor: The power of lookahead in randomized P2P networks. In *Proc. 36th ACM Symp. on Theory of Computing*, pages 54–63, 2004.

[276] Massimo Marchiori. The quest for correct information on the web: Hyper search engines. *Computer Networks*, 29:1225–1236, 1997.

[277] Laurent Massoulié. Community detection thresholds and the weak Ramanujan property. In *Proc. 46th ACM Symp. on Theory of Computing*, pages 694–703, 2014.

[278] Colin McDiarmid. Concentration. In Michel Habib, Colin McDiarmid, Jorge Ramirez-Alfonsin, and Bruce Reed, editors, *Probabilistic Methods for Algorithmic Discrete Mathematics*, pages 195–247. Springer, 1998.

[279] Frank McSherry. Spectral partitioning of random graphs. In *Proc. 42nd IEEE Symp. on Foundations of Computer Science*, pages 529–537, 2001.

[280] Frank McSherry. A uniform approach to accelerated pagerank computation. In *14th Intl. World Wide Web Conference*, pages 575–582, 2005.

[281] Filippo Menczer. Growing and navigating the small world web by local content. *Proc. Natl. Acad. Sci. USA*, 99, 2002.

[282] Filippo Menczer. Lexical and semantic clustering by web links. *J. of the Am. Soc. for Information Science and Technology*, 55(14):1261–1269, 2004.

[283] Weiyi Meng, Clement Yu, and King-Lup Liu. Building efficient and effective metasearch engines. *ACM Computing Surveys*, 34(1):48–89, 2002.

[284] Stanley Milgram. Behavioral study of obedience. *J. of Abnormal and Social Psychology*, 67:371–378, 1963.

[285] Stanley Milgram. The small world problem. *Psychology Today*, 1:61–67, 1967.

[286] George A. Miller. Some effects of intermittent silence. *American Journal of Psychology*, 70:311–314, 1957.

[287] Michael Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1(2):226–251, 2004.

[288] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, 2005.

[289] Bojan Mohar. The Laplacian spectrum of graphs. In Y. Alavi, G. Chartrand, O. Oellermann, and A. Schwenk, editors, *Graph Theory, Combinatorics, and Applications*, volume 2, pages 871–898. Wiley, 1991.

[290] Michael Molloy and Bruce Reed. A critical point for random graphs with a given degree sequence. *Random Structures and Algorithms*, 6:161–180, 1995.

[291] Michael Molloy and Bruce Reed. The size of the largest component of a random graph on a fixed degree sequence. *Combinatorics, Probability and Computing*, 7:295–306, 1998.

[292] Andrea Montanari and Amin Saberi. Convergence to equilibrium in local interaction games. In *Proc. 50th IEEE Symp. on Foundations of Computer Science*, 2009.

[293] Gadi Moran. On the period-two-property of the majority operator in infinite graphs. *Transactions of the American Mathematical Society*, 347(5):1649–1667, 1995.

[294] Stephen Morris. Contagion. *Review of Economic Studies*, 67:57–78, 2000.

[295] Damon Mosk-Aoyama and Devavrat Shah. Fast distributed algorithms for computing separable functions. *IEEE Transactions on Information Theory*, 54(7):2997–3007, 2008.

[296] Elchanan Mossel, Joe Neeman, and Allan Sly. Belief propagation, robust reconstruction, and optimal recovery of block models. In *Proc. 27th Conference on Learning Theory*, pages 356–370, 2014.

[297] Elchanan Mossel, Joe Neeman, and Allan Sly. Consistency thresholds for the planted bisection model. In *Proc. 47th ACM Symp. on Theory of Computing*, pages 69–75, 2015.

[298] Elchanan Mossel, Joe Neeman, and Allan Sly. A proof of the block model threshold conjecture. *Combinatorica*, pages 1–44, 2017.

[299] Elchanan Mossel and Sebastien Roch. Submodularity of influence in social networks: From local to global. *SIAM Journal on Computing*, 39(6):2176–2188, 2010.

[300] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms.* Cambridge University Press, 1990.

[301] Hervé Moulin. On strategy-proofness and single peakedness. *Public Choice*, 35:437–455, 1980.

[302] Harikrishna Narasimhan, David C. Parkes, and Yaron Singer. Learnability of influence in networks. In *Proc. 29th Advances in Neural Information Processing Systems*, pages 3168–3176, 2015.

[303] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[304] Yurii Nesterov. Semidefinite relaxation and nonconvex quadratic optimization. *Optimization Methods and Software*, 9:141–160, 1998.

[305] Praneeth Netrapalli and Sujay Sanghavi. Learning the graph of epidemic cascades. In *Proc. 2012 ACM Sigmetrics Conf. on Measurement and Modeling of Computer Systems*, pages 211–222, 2012.

[306] Jennifer Neville and David Jensen. Relational dependency networks. *J. of Machine Learning Research*, 8:653–692, 2007.

[307] Mark E. J. Newman. The structure of scientific collaboration networks. *Proc. Natl. Acad. Sci. USA*, 98:404–409, 2001.

[308] Mark E. J. Newman. The spread of epidemic disease on networks. *Physical Review E*, 66, 2002.

[309] Mark E. J. Newman. Properties of highly clustered networks. *Physical Review E*, 68, 2003.

[310] Mark E. J. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69, 2004.

[311] Mark E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74, 2006.

[312] Mark E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. USA*, 103:8577–8582, 2006.

[313] Mark E. J. Newman, Albert-László Barabási, and Duncan J. Watts. *The Structure and Dynamics of Networks*. Princeton University Press, 2006.

[314] Mark E. J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69, 2004.

[315] Mark E. J. Newman, Steven Strogatz, and Duncan J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64, 2001. 026118.

[316] Andrew Y. Ng, Alice X. Zheng, and Michael I. Jordan. Link analysis, eigenvectors, and stability. In *Proc. 17th Intl. Joint Conf. on Artificial Intelligence*, pages 903–910, 2001.

[317] Andrew Y. Ng, Alice X. Zheng, and Michael I. Jordan. Stable algorithms for link analysis. In *Proc. 24th Intl. Conf. on Research and Development in Information Retrieval (SIGIR)*, pages 258–266, 2001.

[318] Van Nguyen and Charles U. Martel. Analyzing and characterizing small-world graphs. In *Proc. 16th ACM-SIAM Symp. on Discrete Algorithms*, pages 311–320, 2005.

[319] Ryan O'Donnell. Some topics in analysis of boolean functions. In *Proc. 40th ACM Symp. on Theory of Computing*, pages 569–578, 2008.

[320] Christos Papadimitriou, Prabhakar Raghavan, Hisao Tamaki, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proc. 17th ACM Symp. on Principles of Database Systems*, pages 159–168, 1998.

[321] Vilfredo Pareto. *Cour d'Economie Politique*. Droz, 1896.

[322] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86:3200–3203, 2001.

[323] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic dynamics in finite scale-free networks. *Physical Review E*, 65, 2002.

[324] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[325] Gabriel Pinski and Francis Narin. Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Information Processing and Management*, 12(5):297–312, 1976.

[326] Boris Pittel. On spreading a rumor. *SIAM J. Applied Math.*, 47:213–223, 1987.

[327] C. Greg Plaxton, Rajmohan Rajaraman, and Andréa W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory of Computing Systems*, 32(3):241–280, 1999.

[328] Svatopluk Poljak and Miroslav Sura. On periodic behavior in societies with symmetric influences. *Combinatorica*, 3(1):119–121, 1983.

[329] Mason A. Porter, Jukka-Pekka Onnela, and Peter J. Mucha. Communities in networks. *Notices of the American Mathematical Society*, 56(9):1082–1097, 2009.

[330] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proc. Natl. Acad. Sci. USA*, 101:2658–2663, 2004.

[331] Filip Radlinski and Thorsten Joachims. Query chains: Learning to rank from implicit feedback. In *Proc. 11th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 239–248, 2005.

[332] Davood Rafiei and Alberto O. Mendelzon. What is this page known for? Computing web page reputations. *Computer Networks*, 33:823–835, 2000.

[333] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott J. Shenker. A scalable content-addressable network. In *Proc. 20th ACM SIGCOMM Conference*, pages 161–172, 2001.

[334] R. Ravi. Rapid rumor ramification: Approximating the minimum broadcast time. In *Proc. 35th IEEE Symp. on Foundations of Computer Science*, pages 202–213, 1994.

[335] Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *Proc. 29th ACM Symp. on Theory of Computing*, 1997.

[336] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74, 2006. 016110.

[337] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of Mathematics*, 155:157–187, 2002.

[338] Diana Richards, Whitman A. Richards, and Brendan McKay. Collective choice and mutual knowledge structures. *Advances in Complex Systems*, 1:221–236, 1998.

[339] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *Proc. 8th Intl. Conf. on Knowledge Discovery and Data Mining*, pages 61–70, 2002.

[340] Garry Robins, Philippa E. Pattison, Yuval Kalish, and Dean Lusher. An introduction to exponential random graph ($p*$) models for social networks. *Social Networks*, 29(2):173–191, 2007.

[341] Everett Rogers. *Diffusion of innovations.* Free Press, 5th edition, 2003.

[342] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. 18th IFIP/ACM Intl. Conf. on Distributed Systems Platforms (Middleware 2001)*, pages 329–350, 2001.

[343] Kazumi Saito, Ryohei Nakano, and Masahiro Kimura. Prediction of information diffusion probabilities for independent cascade model. In *Proc. 12th Intl. Conf. on Knowledge-Based and Intelligent Information & Engineering Systems*, pages 67–75, 2008.

[344] Mahyar Salek, Shahin Shayandeh, and David Kempe. You share, I share: Network effects and economic incentives in P2P file-sharing systems. In *Proc. 6th Workshop on Internet and Network Economics (WINE)*, pages 354–365, 2010.

[345] Mark A. Satterthwaite. Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.

[346] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[347] Thomas Schelling. *Micromotives and Macrobehavior.* Norton, 1978.

[348] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels.* MIT Press, 2002.

[349] Lawrence S. Schulman. Long range percolation in one dimension. *J. Phys. A*, 16:L639–L641, 1983.

[350] John Scott. *Social Network Analysis: A Handbook.* Sage Publications, second edition, 2000.

[351] Erik Selberg and Oren Etzioni. The metacrawler architecture for resource aggregation on the web. *IEEE Expert*, 12(1):8–14, 1997.

[352] Devavrat Shah. Gossip algorithms. *Foundations and Trends in Networking*, 3(1):1–125, 2008.

[353] Herbert A. Simon. On a class of skew distribution functions. *Biometrika*, 42(3/4):425–440, 1955.

[354] Karen B. Singer. *Random Intersection Graphs.* PhD thesis, Johns Hopkins University, 1995.

[355] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *Proc. 36th ACM Symp. on Theory of Computing*, pages 81–90, 2004.

[356] Daniel A. Spielman and Shang-Hua Teng. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on Computing*, 42(1):1–26, 2013.

[357] Gilbert W. Stewart and Ji-Guang Sun. *Matrix Perturbation Theory.* Academic Press, 1990.

[358] Mark Steyvers and Joshua B. Tenenbaum. The large-scale structure of semantic networks: statistical analyses and a model of semantic growth. *Cognitive Science*, 29(1):41–78, 2005.

[359] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proc. 20th ACM SIGCOMM Conference*, pages 149–160, 2001.

[360] Gilbert Strang. *Linear Algebra and Its Applications.* Brooks Cole, fourth edition, 2005.

[361] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1):41–43, 2004.

[362] Chaitanya Swamy. Correlation clustering: Maximizing agreements via semidefinite programming. In *Proc. 15th ACM-SIAM Symp. on Discrete Algorithms*, pages 519–520, 2004.

[363] Lei Tang and Huan Liu. *Community Detection and Mining in Social Media.* Morgan & Claypool, 2010.

[364] Youze Tang, Yanchen Shi, and Xiaokui Xiao. Influence maximization in near-linear time: A martingale approach. In *Proc. 34th ACM SIGMOD Intl. Conference on Management of Data*, pages 1539–1554, 2015.

[365] Youze Tang, Xiaokui Xiao, and Yanchen Shi. Influence maximization: near-optimal time complexity meets practical efficiency. In *Proc. 33rd ACM SIGMOD Intl. Conference on Management of Data*, pages 75–86, 2014.

[366] Benjamin Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Proc. 18th Conf. on Uncertainty in Artificial Intelligence*, pages 485–492, 2002.

[367] Jeffrey Travers and Stanley Milgram. An experimental study of the small world problem. *Sociometry*, 32(4):425–443, 1969.

[368] Michel Truchon. An extension of the Condorcet criterion and Kemeny orders. Technical Report cahier 98-15, Centre de Recherche en Économie et Finance Appliquées, 1998.

[369] Thomas W. Valente. *Network Models of the Diffusion of Innovations.* Hampton Press, 1995.

[370] Robbert van Renesse. Scalable and secure resource location. In *Proc. 33rd Hawaii Intl. Conf. on System Sciences*, 2000.

[371] Robbert van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 2003.

[372] Vijay V. Vazirani. *Approximation Algorithms.* Springer, 2001.

[373] Olga Veksler. *Efficient Graph-Based Energy Minimization Methods in Computer Vision.* PhD thesis, Cornell University, 1999.

[374] Adrian Vetta. Nash equlibria in competitive societies with applications to facility location, traffic routing and auctions. In *Proc. 43rd IEEE Symp. on Foundations of Computer Science*, pages 416–425, 2002.

[375] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proc. 40th ACM Symp. on Theory of Computing*, pages 67–74, 2008.

[376] Jan Vondrák. Symmetry and approximability of submodular maximization problems. In *Proc. 50th IEEE Symp. on Foundations of Computer Science*, pages 651–670, 2009.

[377] Chi Wang, Wei Chen, and Yajun Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery Journal*, 25(3):545–576, 2012.

[378] Stanley S. Wasserman and Katherine Faust. *Social Network Analysis.* Cambridge University Press, 1994.

[379] Stanley S. Wasserman and Philippa E. Pattison. Logit models and logistic regressions for social networks: I. an introduction to markov random graphs and p^*. *Psychometrika*, 60:401–425, 1996.

[380] Duncan J. Watts. A simple model of fads and cascading failures. Technical Report 00-12-062, Santa Fe Institute Working Paper, 2000.

[381] Duncan J. Watts, Peter S. Dodds, and Mark E. J. Newman. Identity and search in social networks. *Science*, 296:1302–1305, 2002.

[382] Duncan J. Watts and Steven Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393:440–442, 1998.

[383] Douglas R. White and Frank Harary. The cohesiveness of blocks in social networks: Node connectivity and conditional density. *Sociological Methodology*, 31(1):305–359, 2001.

[384] Harrison C. White. Search parameters for the small world problem. *Social Forces*, 49:259–264, 1970.

[385] Herbert S. Wilf. *generatingfunctionology*. Academic Press, 1994.

[386] Nicholas C. Wormald. Differential equations for random processes and random graphs. *Annals of Applied Probability*, 5:1217–1235, 1995.

[387] Nicholas C. Wormald. The differential equation method for random graph processes and greedy algorithms. In M. Karonski and H. Proemel, editors, *Lectures on Approximation and Randomized Algorithms*, pages 73–155. PWN, 1999.

[388] Shuang-Hong Yang and Hongyuan Zha. Mixture of mutually exciting processes for viral diffusion. In *Proc. 30th Intl. Conf. on Machine Learning*, pages 1–9, 2013.

[389] H. Peyton Young. Condorcet's theory of voting. *American Political Science Review*, 82(4):1231–1244, 1988.

[390] H. Peyton Young. *Individual Strategy and Social Structure: An Evolutionary Theory of Institutions*. Princeton University Press, 1998.

[391] H. Peyton Young. The diffusion of innovations in social networks. Technical Report 02-14-018, Santa Fe Institute Working Paper, 2002.

[392] Hui Zhang, Ashish Goel, Ramesh Govindan, Kahn Mason, and Benjamin van Roy. Making eigenvector-based reputation systems robust to collusion. In *WAW 2004*, pages 92–104, 2004.

[393] Ben Q. Zhao, John D. Kubiatowicz, and Anthony D. Joseph. Tapestry: A fault-tolerant wide-area application infrastructure. *Computer Communications Review*, 32(1):81, 2002.

[394] Ke Zhou, Hongyuan Zha, and Le Song. Learning social infectivity in sparse low-rank network using multi-dimensional hawkes processes. In *Proc. 30th Intl. Conf. on Machine Learning*, pages 641–649, 2013.

[395] George K. Zipf. *Selective studies and the principle of relative frequency in language*. Harvard University Press, 1932.

[396] George K. Zipf. *Human Behavior and the Principle of Least Effort, an Introduction to Human Ecology*. Addison-Wesley, 1949.