ABSTRACT

Title of dissertation: USING AND MANIPULATING PROBABILISTIC CONNECTIVITY IN SOCIAL NETWORKS

Thomas DuBois, Doctor of Philosophy, 2011

Dissertation directed by: Professor Aravind Srinivasan
Department of Computer Science

Probabilistic connectivity problems arise naturally in many social networks. In particular the spread of an epidemic across a population and social trust inference motivate much of our work. We examine problems where some property, such as an infection or influence, starts from some initially seeded set of nodes and every affected node transmits the property to its neighbors with a probability determined by the connecting edge. Many problems in this area involve connectivity in a random-graph - the probability of a node being affected is the probability that there is a path to it in the random-graph from one of the seed nodes. We may wish to aid, disrupt, or simply monitor this connectivity. In our core applications, public health officials wish to minimize an epidemic's spread over a population, and connectivity in a social network suggests how closely tied its users are. In support of these and other applications, we study several combinatorial optimization problems on random-graphs. We derive algorithms and demonstrate their effectiveness through simulation, mathematical proof, or both.

Using and Manipulating Probabilistic Connectivity in Social Networks

by

Thomas DuBois

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2011

Advisory Committee:
Professor Aravind Srinivasan, Chair/Advisor
Professor William Gasarch
Professor Jennifer Golbeck
Professor Armand Makowski
Professor Madhav Marathe
Professor David Mount

Dedication

I dedicate this dissertation to everyone who has inspired my desire to learn about the world.

Acknowledgments

Many people contributed in one way or another towards this thesis and my development while in graduate school.

First, my adviser and friend, Aravind Srinivasan has my most profound thanks. Aravind has consistently worked to make me the best scientist I can be, and to work on the biggest, most important problems that I can. One the technical side of things, this means that he would suggest new and interesting applications for study, and that he was always ready with great advice on technical details or a write-up. While he pushes me academically, Aravind encourages work-life balance and individual growth, both of which he models in his own life. More recently he has given me invaluable advice on my career options, including using his contacts to find potential matches. Besides being an excellent adviser, Aravind is also a great friend, and I expect to maintain this friendship long after graduation.

My research has been greatly enriched by others I have collaborated with. My work on epidemic minimization results from a collaboration with many people at the Network Dynamics and Simulation Science Laboratory at Virginia Tech. My visits to Blacksburg have always been both productive and enjoyable. Many interesting ideas have come about from my conversations with Chris Barrett, Stephen Eubank, Madhav Marathe, Anil Vulikanti, and others. Closer to home, I greatly value the work on trust inference that I have done with Jennifer Golbeck over the years. We have done very interesting work together, and she is always a good source for career advice too. I have worked with and benefited from a large number of other people,

including but not limited to Bill Gasarch, Raghu Murtugudde, Marc Olano, Amir Sapkota, Sridevi Shivarajan, and Uzi Vishkin.

In addition to those who have supported me professionally, I would also like to thanks everyone who has supported me personally during my time in graduate school. My wife Karolina is first and foremost in this group. I greatly appreciate her love and support during the entire process, but especially during those times when I worked evenings and weekends to make a deadline. My parents Pauline Torisky and Gerald DuBois have always encouraged my education, and have whole heartedly supported my graduate studies. The rest of my family, as well as Karolina's family have supported me as well. Most recently of all, my son Andrew's inquisitiveness and drive to explore are as inspirational as they are adorable.

Finally I acknowledge those who have supported my work financially. During my graduate studies, my work has been continuously funded through either teaching or research assistantships. I thank the many professors: Jennifer Golbeck, Jonathan Katz, Madhav Marathe, Raghu Murtugdde, and Aravind Srinivasan, who have worked to support me through their grants from the National Science Foundation and the Defense Advanced Research Projects Agency. Specific grants include: NFS grants CNS 1010789, ARO Award 1010350, CNS 0426683, CNS 0626636, CNS 0626964, and NFS award 0627306 and a DARPA Seedling grant.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

SOAD	Second order average degree
SOAED	Second order average expected degree
R_0	The reproductive number of an epidemic in a population
SIR	Susceptible, Infectious, Recovered
SIS	Susceptible, Infectious, Susceptible
EIP	External Infection Probability

Chapter 1

Introduction

Probabilistic connectivity problems arise naturally in many social networks. In particular the spread of an epidemic across a population and social trust inference motivate much of our work. We examine problems where some property, such as an infection or influence, starts from some initially seeded set of nodes and every affected node transmits the property to its neighbors with a probability determined by the connecting edge. Many problems in this area involve connectivity in a random-graph - the probability of a node being affected is the probability that there is a path to it in the random-graph from one of the seed nodes. We may wish to aid, disrupt, or simply monitor this connectivity. In our core applications, public health officials wish to minimize an epidemic's spread over a population, and connectivity in a social network suggests how closely tied its users are. In support of these and other applications, we study several combinatorial optimization problems on random-graphs. We derive algorithms and demonstrate their effectiveness through simulation, mathematical proof, or both.

The common thread running through these problems is the connectivity in a random graph, and so our main techniques to tackle them come from graph connectivity algorithms and probability theory. Since connectivity probabilities are #P-complete to measure even in simplified forms [94], we cannot efficiently calcu-

late them exactly. To obtain estimates of these values, we rely on random sampling. Beyond that, we make heavy use of standard algorithm design techniques including linear programming, dynamic programming, greedy algorithms (with analyses using the probabilistic method), non-linear optimization, and more.

1.1 Epidemiology

One of the most pressing settings for these types of problems, and that which we study in the most depth, is that of epidemic transmission across a population. The recent outbreaks of SARS [13] and H1N1 flu [83] serve to highlight the importance of epidemic prevention and mitigation. Various countermeasures have been studied in great depth, including decreasing transmission rates (better personal hygiene or face masks), social distancing (closing schools, encouraging people to stay home when sick), and pharmaceuticals (vaccination, anti-virals). These measures have been studied most thoroughly by assuming a uniform mixing model of the population and applying differential equations. These uniform mixing models have evolved greatly from their introduction by Kermack and McKendrick [65]. Modern models take into account a network's degree structure [12], heterogeneous mixing between distinct subpopulations [83], and game theoretic approaches [88] among other extensions. All of these abstract away the fact that diseases do not jump directly from an infected person in one country, city, or even neighborhood to another without a physical link. Studies that do consider a fixed contact network over which an epidemic propagates tend to focus on purely simulation based results [43, 40, 38]

or look at the contact network for only a subset of the graph such as air travelers [60, 59]. The theoretical results about contact networks usually involve whether or not a disease will spread to a large fraction of people instead of how to combat the epidemic [87, 21].

Our work in Chapter 3 bridges the gap between simulation based epidemic minimization and mathematical results for when an unchecked epidemic will be wide-spread. We focus on using the graph-theoretic aspects of the local structure of disease transmissions and attempt to develop better targeted algorithms for the applying epidemic countermeasures. The idealized goal of this research is to answer questions such as "given the social contact graph and disease model details, how can we use information that would be available to public health officials in real-time to optimally allocate limited intervention resources to minimize the total number sick/deaths/cost/etc. of the epidemic".

Within this framework we develop network-aware interventions to minimize the effect of an epidemic within the general population. Here, as in all of our studies, we use a two pronged approach. First we derive theoretical results wherever possible for the effectiveness of our intervention strategies. Our most important result in Section 3.1 is the discovery that any "small" set of nodes whose vaccination protects a large fraction of the population will cumulatively have a large expected subtree size. While we do not know how to directly find such sets, our result suggests prioritizing those with high subtree sizes for vaccination. Second, we compare our theory-driven intervention strategies against existing techniques using simulations.

The structure of a population's contact networks plays an important role in the effectiveness of our algorithms. Therefore we develop these algorithms in a data-centric way. In networks with high expansion, the giant component results of Chung, Horn, and Lu [21] come to dominate. In this setting the best containment strategy reduces the second order average degree (SOAD) below 1. This can be approximated well in a localized way by targeting nodes who have large incident edge probability sums. But does this apply to the networks we are interested in? Properties common to large social networks have been studied extensively [85, 76, 69]. For example Leskovec et al. perform a large scale study of the conductance of many online social networks [74]. They conclude that there is a fixed plateau in subset size which does not depend on network size, where no larger subsets with good conductance exist. This suggests that we may not find sufficient community structure to improve on the SOAD-based approach. However real populations have a natural spatial embedding, and therefore differ in important ways from online networks.

While most epidemic containment literature focuses on protected the general population, certain subpopulations may require more protection. In an emergency situation, the effect of losing a large number of people who provide or maintain utilities, police, medical, logistical, or other services is much larger than the direct effects of the incapacitated individuals themselves. One way to supply extra protection involves removing these societally-critial individuals from the general population, and isolating them. We call this intervention *protective sequestration*. At the time of sequestering, some individuals may be latently infected, bringing the epidemic with them. In Section 3.2 we develop an algorithm which takes estimates of the latent

infection probabilities for the sequestered population and partitions them in a way that minimizes the total expected infections. Using this algorithm we go on to explore some of trade-offs involved in implementing a sequestering policy.

1.2 Trust Inference

In Chapter 4, we examine the seemingly unrelated area of social network trust inference. Trust, defined as one person's confidence that another will behave in a desired manner [48], is an important property of human interaction, and thus it is important in the social web as well. Trust can help users make confident purchases [89], decide what statements are likely to be true [63], or find others with similar opinions [104]. Direct trust's usefulness is limited because in a large social network a user will have direct knowledge of the trustworthiness of only a small fraction of others. For trust to be useful in these settings, we need an algorithm that uses direct trust ratings to infer trust between any two users.

We will not consider methods to directly calculate the trust between users, but instead we rely on the network of users rating the "trustworthiness" of their neighbors. Several different varieties of trust inference algorithms have been proposed [103, 70, 31, 77], each with its own strengths and weaknesses. These techniques estimate trus in a simulation-based way, and none of them have a simple mathematical structure that can be exploited. In Section 4.1 we fill in this gap with an algorithm based upon a novel and intuitive interpretation where direct trust ratings

are taken to be edge probabilities and inferred trust is the probability that a trust path exists between two endpoints.

This random graph interpretation has many benefits to explore and implementation challenges to overcome. We observe that the log of the inverse of the path probabilities in a symmetric trust graph form a trust metric space. This immediately admits an extensive array of metric space algorithms, such as clustering and visualization, on our trust dataset. Clustering in particular is an important applications, and we examine, both theoretically and empirically, several ways in which we can cluster based upon trust. Additionally we can use the relationship between direct trust and inferred trust to quickly identify edges which have a large effect on the graph. Any edge whose direct trust is significantly smaller than the inferred trust is in a sense redundant, while those whose direct trust comprises most of the inferred trust are critical.

We encounter several challenges when implementing this algorithm and applying it to real datasets. These datasets typically have some way of quantifying the direct trust on an edge. Our first challenge involves carefully choosing a mapping from trust values to edge probabilities so that we produce informative results. The mapping should be set high enough that trust spreads sufficiently, but low enough that most path probabilities do not approach one. As a general rule, we find that choosing this mapping so that a giant component just barely forms produces good results. The second major challenge is scalability. Computing path probabilities exactly is #P-complete, so we use random sampling to estimate the probabilities accurately. Storing these pairwise estimates explicitly takes space $O(|V|^2)$ which is

not good enough for very large networks. Instead, for k samples we store $O(k \cdot |V|)$ values from which computing a single pair's connectivity probability takes time $O(k)$.

Our initial algorithm, as well as most trust inference schemes, considers only positive trust. In this setting, any edge indicates more trust than a lack of an edge, and there is no way for a user to differentiate between not knowing someone and knowing someone to be dishonest. Some studies have specifically addressed distrust [105], but none present fully satisfying approaches that integrate trust and distrust. In Section 4.2 we develop a method based upon spring embedding - a simulated process where positive edges attract and negative edges repel, and we combine these results with our positive-only path probability algorithm. We discuss some of this technique's properties and validate it on the hidden-sign predictions problem – where we remove some edges from the graph and attempt to infer their signs from the remaining graph. We find that by combining when endpoint path probabilities and spring embedding distances into a set of two-dimensional points, we can infer hidden edge signs quite well for a number of large networks.

Trust inference can be used directly to inform one user about another, but it can also be used to partition users into groups with high intra-group trust. There are two ways that we can validate how well our algorithm clusters a network. First, we can show how to do something well using the clusters. DuBois et al. show the such clusters can help improve recommender system accuracy [33]. Second, we can show that the resulting clusters resemble some inherent groupings within the dataset. Unfortunately we are unable to find any large social networks with

meaningful, non-trivial groups highlighted for comparison. Even if we did, there would be no reason to believe that the supplied grouping was the only one that made sense. In order to test our algorithm at a proof-of-concept level, in Section 4.3 we create an artificial network where the clusters are well defined, but non-trivial to infer from the network alone. We visually demonstrate that the spring-embedding algorithms reconstructs the clusters quite well.

As networks scale into the millions of nodes and beyond, our trust inference techniques may not be fast enough for some applications. In Section 4.4 we develop a linear-time, randomized algorithm for clustering. While we empirically examine its application to trust networks, our algorithm has probabilistic approximation guarantees which apply to any system where clusterings can only be sampled randomly from a black-box. Using either of two distance metrics on clusterings we prove that our resulting clustering is within a factor of 2 or 3 of optimal in expectation. We also show how repeated sampling can reduce the probability of producing a clustering much worse than this expectation.

Chapter 2

Mathematical Background

Since we develop algorithms with rigorous proofs behind them, in this section we present a brief overview of some related mathematical methods and ideas. We do this with a focus towards their use in our work. In addition, we introduce some of our mathematical constructs which are useful throughout our research.

All of our work involves some sort of connectivity in random graphs. Erdős and Rényi's pioneering work [36] in this area considered a complete graph on n vertices where each edge is kept with probability p. They show that if $np > 1+\Omega(1)$ then with high probability the largest connected component has size $\Theta(n)$ and if $np < 1 - \Omega(1)$ then all of the components will almost certainly be $O(\log n)$. This result is a discrete analog of the reproductive number R_0 [79] in epidemiology (the number of secondary infections resulting from a single initial infection). If a single person (or node) will connect with more than one other ($R_0 > 1$ or $np > 1$) then the number of infections increases exponentially with time and a large number of others will be affected. If s single person connects with less than one other ($R_0 < 1$ or $np < 1$), an epidemic will die out exponentially with time. We are most interested in the existence of giant components in epidemiology (we want our control measures to stop their formation) and trust inference parameter selection (the probability of keeping edges should be kept small enough that giant components do not dominate).

Of course these networks are rarely complete graphs, so we rely on the many extensions available in the literature. Two of the most relevant are those of Moore and Newman [87] which analyses the formation of a giant component on a small-world network (a type of network proposed by Jon Kleinberg that combines many local and few long-range edges to yield fast, decentralized routing [67]), and of Chung, Horn, and Lu [21]. The latter of these builds on powerful spectral graph theory techniques [19] to establish edge probabilities which bring about a giant component for a large class of graphs. They show that for most graphs with second order average degree (the sum of the squares of the degrees divided by the sum of the degrees) of $\hat{d}$, a giant component develops if and only if p is at least $1/\hat{d}$.

While the existence of a giant component relates to our goals, on a more fine grained level we are interested in the connectivity probability between individual pairs of nodes. Measuring these probabilities is $\#P$-complete [94]. Fortunately we can sample instances of the random graph some k times, and for each pair u, v record the number of times they are connected $X_{u,v}$. Since the samples are independent, we can apply Chernoff bounds [18] to the probability that our estimates deviate significantly from their expected value. Specifically, the probability that $X_{u,v}/k$ deviates from the true connectivity probability $E[X_{u,v}/k]$ by more than δ is at most $2e^{-k^2\delta^2/E[X_{u,v}]} \leq 2e^{-k\delta^2}$. If we set k as low as $O(\log n/\delta^2)$, we can then apply Markov's inequality to each X value simultaneously and find that with high probability all $E[X/k]$ values will be estimated simultaneously to within δ. We use this same technique of random sampling in several other places as well, including

accurately estimate the number of infections that can be traced back through a specific node.

Once we have connectivity probability estimates, we use them to establish a metric space on the nodes as follows. For any nodes u and v with connectivity probability p, define their distance as $d(u, v) = \log 1/p$. As long as the edges are chosen independently and we consider any two nodes which are always connected to be equivalent, d satisfies all of the properties of a metric space. Namely $\forall u, v$:

$$d(u, v) = \log 1/p \geq 0$$

$$d(u, v) = \log 1/p = d(v, u)$$

$$d(u, w) = \log 1/p_{u,w} \leq \log 1/(p_{u,v} \cdot p_{v,w}) = d(u, v) + d(v, w).$$

The last property holds because the existence of a path in a random-graph is a monotone function of the edges kept by the random graph – once a path exists, adding more edges can not disrupt the path. Therefore, by the FKG inequality, the probability of paths from u to v and v to w (which implies a path from u to w) is at least the product of the individual probabilities of the two paths. In the case where edges in the graph are directed, this formulation yields an asymmetric metric space.

Since we have a novel metric space on the nodes where the further apart two nodes are, the lower the probability of a path between them, we can make use of existing metric clustering algorithms to partition the nodes. A clustering algorithm takes a set of points in a metric space and groups them in a way that tries to optimize some property related to the groups' internal closeness. Examples include, k-centers which finds a set of k points S which minimizes the maximum distance

from any point to its closest point in S, k-means which partitions the points into k sets in a way that minimizes the variance within each group, and correlation clustering which partitions the points in a way that minimizes the sum of distances within groups minus the sum of distances across groups. Each of these clustering objective functions have good approximation algorithms when applied to points in a symmetric metric space [56, 64, 8, 1], and some even have good approximations in an asymmetric metric space [3].

In applying our algorithms to real networks, the network expansion can significantly affect the outcomes. Define $E(S,T)$ to be the set of edges with one endpoint in S and the other in T. The edge expansion ϵ of a graph (V,E) is defined as $\min_{S\subseteq V,|S|<|V|/2} |E(S,\overline{S})|/|S|$, or the minimum over all "small" subsets S of the ratio of edges leaving S to the size of S. To understand how expansion affects our applications, consider the expander mixing lemma as applied to a d-regular graph [4]. The lemma says that given a d-regular graph with a second highest eigenvalue magnitude λ,

$$\forall S,T \subset V : \left| E(S,T) - d \cdot \frac{|S| \cdot |T|}{|V|} \right| \leq \lambda \sqrt{|S| \cdot |T|}.$$

High expansion restricts λ to be small, in which case the number of edges between any two sets S and T is close to what we would expect in a random d-regular graph. Intuitively this means that the higher a graph's expansion, the closer it behaves to the differential model of propagation. We can also think of this as saying that the lower the expansion, the more the network's structure can be exploited. As a simple example, consider a contact network where a relatively small number of edge

removals can contain an epidemic by significantly reducing the size of the largest component. Because the large contained set of nodes must have a small number of edges to the rest of the graph, such a graph cannot have high expansion. For a thorough discussion of expander graphs, see the survey by Hoory, Linial, and Wigderson [58].

Since our work centers on random graphs, we also make use of a number of powerful probabilistic techniques. We already discussed the application of taking independent (or negatively correlated) random samples and applying Chernoff bounds to their sum. When random variables are not independent, we turn to other methods. One such method is the linearity of expectation. It says that for any two random variables A and B, regardless of their dependence $E[A+B] = E[A]+E[B]$. We also make use of union bounds, which say for any random events A and B, the probability that either happens is at most the sum of their individual probabilities. In some cases we have an explicit, limited dependency structure that we can exploit to show bounds much tighter than if there was no structure. In these cases, we may be able to apply large deviation bounds such as those of Fortuin, Kasteleyn, and Ginibre [41] and Janson [61].

In addition to the mathematical techniques we use, we also use some programming techniques that warrant a brief overview. The most significant of these is dynamic programming. Dynamic programming breaks a problem into smaller subproblems of the same type which can be efficiently combined into a solution to the larger problem. Consider an example we use as a subroutine in one of our algorithms: given k independent indicator random variables $X_1, \ldots X_k$, what

is the probability distribution on their sum $\sum_{l=1}^{k} X_l$? To solve this problem, create a k by k array A, and solve for $A_{i,j}$ equals the probability that $\sum_{l=1}^{i} X_l = j$. The i^{th} row contains the distribution we want. Initialize the trivial entries of $A_{i,j}$ when $i = 1$ or $j = 1$. For all other entries, the probability that $\sum_{l=1}^{i} X_l = j$ is $Pr[X_i = 0] \cdot A_{i-1,j} + Pr[X_i = 1] \cdot A_{i-1,j-1}$. Thus we iterative build up solutions to larger and larger subproblems until we reach the desired level.

Chapter 3

Epidemiology

Infectious diseases, and particularly respiratory infectious diseases, account for a large fraction of deaths worldwide [66]. In addition to the seasonal flu that takes a significant toll on health and productivity, we have the ever looming threat of a global pandemic. The main public health response to contain these outbreaks consists of increased detection and isolation of infected individuals, changing contact patterns by closing schools or other institutions, and giving anti-virals or (once an effective vaccine is prepared) vaccines to large segments of the population [43, 40, 38]. Broadly, the research into epidemic containment uses one of two categories of models for how the disease spreads: ones that assume some variant of uniform mixing populations and use differential equations, and ones that take into account the contact network structure. Most of these take a close look at some aspect of the epidemic transmission process (such as mixing rates of different subpopulations [83], air travelers [60, 59], or disease parameter uncertainty [99]), and evaluate the effects of several heuristic mitigation strategies using simulations. See the survey conducted by Meyers [84] for a more thorough description of a wide range of studies in the area.

Most of this work uses the SIR model, and so do we. This model dictates that individuals can be in one of three main states for which the model is named: susceptible, infectious, or recovered. Susceptible people become infected through

contact with an infected individual, infected people eventually recover, and recovered people retain an immunity and neither spread the disease nor become infected again. There are other models (susceptible, infectious, susceptible, or SIS for example), however SIR corresponds most naturally to diseases where vaccinations are effective, which is our concentration.

We adopt a further refinement of the SIR model where the transmission of the disease from an infected individual to an uninfected one happens probabilistically with a constant rate per unit of contact time [90]. Therefore if there is a fixed amount of contact time between two people, we can simplify their edge in the contact graph by giving it a single probability p of remaining in the graph. Once we reduce the contacts to probabilistic edges, a person will become infected if and only if they are in the same connected component as one of the initial seeds of the epidemic. This view motivates our focus on graph cut-like problems. If we can minimize the expected size of the components containing infected individuals, we successfully minimize the epidemic's spread. Note that in this analysis, time is not considered explicitly. Rather, it is implicitly embedded in the probabilistic edges and the lengths of paths.

A substantial portion of our work on epidemic containment focuses on removing nodes or edges from an existing contact graph. This can correspond to vaccination, administering anti-virals, or social distancing. The problems take the form: given a graph G, a disease model, probabilistic information about who is currently infected, and a resource constrained countermeasure, how can we apply the countermeasure to minimize some epidemic cost function. We develop provable results whenever possible, derive heuristics based upon our findings, and experimen-

tally validated these heuristics. Conducting controlled experiments in the field is impractical for many reasons. Therefore in order to generate experimental results we use the EpiFast simulation infrastructure developed at the Network Dynamics and Simulation Science Laboratory which is part of the Virginia Bioinformatics Institute at Virginia Tech [38]. This setup includes several large scale datasets where a city or region-wide person-person graph is constructed implicitly from a detailed and realistic person-location graph, as well as a distributed program which simulates the progress of an epidemic across these populations. We describe EpiFast in more detail in Section 5.1.

In Section 3.1 we start our technical discussion with an algorithm for targeted vaccinations. Given a social contact graph and probabilistic information on the initially infected set, we create estimates, for each node, of the expected number of nodes infected through that node which we call the node's expected subtree size. We do this through repeated simulation, where for each simulation we record the paths the epidemic takes. For each node, we count its subtree size, and take the average over many iterations. We need $O(\log n)$ simulations to have highly accurate estimates of the expected subtree sizes for all nodes in the graph. Vaccinating one person, and therefore decreasing or elimination its subtree does not necessarily reduce the epidemic much, as other subtrees many become larger. The key here is to find groups of nodes whose mutual vaccination significantly diminishes the total epidemic size. This subtree based method has the nice property: if there exists a set of nodes S such that their vaccination reduces the total expected outbreak size by X, then the sum of their expected subtree sizes is at least X. Roughly this means that

under certain conditions, if there exists a small group of nodes whose vaccination will save much of the graph, then we can find a (hopefully not too large) superset of that set efficiently.

We empirically compare this algorithm with three others: one that vaccinates randomly, one that vaccinates in decreasing order of infection probability, and one that vaccinates purely based on local criteria - the sum of a nodes incident edge probabilities. Random vaccinations provide a base-line from which to measure improvement. Vaccinating based upon infection probability is one of the simplest global heuristics. We chose vaccination by decreasing order or local connectivity for several reasons. First, there are spectral reasons to believe it should be effective. Specifically, it approximates a strategy which decreases the graph's second order average degree (SOAD) as much as possible and thus significantly decreases the largest eigenvalue as well. In addition, it is an easier strategy to implement in a real-world setting because estimating local connectivity is a simpler problem than reconstructing a full contact network.

While evaluating the strength of these algorithms is important, we also wish to understand why the algorithms perform as well as they do. To do this, we need to understand the data they are operating on. Networks that are good expanders by definition do not have small sets of nodes whose protection will stop an epidemic. In these cases, the spectral based methods may be the best available. However networks with a natural embedding of their nodes, such as a geographical embedding, may admit much better solutions. We use spectral measures to describe the New River

Area contact network, argue intuitively why its mathematical description makes sense, and conjecture how other types of contact networks might differ.

Then we move to a discussion about protective sequestering, a mitigation strategy designed to protect a small critical population from an epidemic in the general population. Sequestering involves taking the critical population out of the general population and placing them into isolated groups (because of logistical reasons isolating individuals by themselves is impractical, instead we think of groups as being tens, hundreds, or even thousands of people). These groups could naturally correspond to a military barracks, a section of a hospital, or any other geographically or socially isolated location. If these groups are kept sufficiently isolated, the only members who become infected are those who were latently infected before sequestering began and those whom they subsequently infect. There are two main challenges involved with sequestering in a useful way. The first is how to trigger the sequestering process, and the second is how to partition people into groups. We have considerable work done on when to trigger the sequestering, and under certain conditions we have an optimal partitioning algorithm. We present our algorithm along with several refinements that make it more efficient in both space and time. Then we study several tradeoffs inherent in any real-world use of protective sequestering. These results come from the joint work of myself and others currently available as a technical report [15].

We finish this section with a look at what happens to the degree sequence of a graph when its edges are subject to independently random removal. The degree sequence of a graph is a list $d_1, \ldots d_k$ where, for all degrees i, d_i is the number of

nodes with degree i. Degree sequences are one of many well-studied parameters of large graphs, and many other properties often relate to it. Rather than focusing on a particular degree sequence, people often study asymptotic ones. Common classes of degree sequence include exponential (as in a $G(n, p)$ random graph), power-law (which for a time was thought to characterize many large graphs [39, 10, 11, 68]), and stretched-exponential[26]. We derive three categories of smooth degree sequences, and for each we show how the degree sequence changes after independently random edge removals. Our results take the form of a new sequence of expected degrees plus large deviation bounds from these expectations.

3.1 Vaccination Strategies

We start our discussion of vaccination strategies with an overview of the disease model we use. When a new flu-like epidemic first infects a human population, it spreads from host to host in a discrete and localized way. We model this localization by placing an edge between any two people who come into sufficient contact that a transmission may occur. Parameterizing these edges presents a challenge since the details of how the contagion travels across an edge from one host to another can be very complex. Different individuals have varying levels of natural immunity, infection incubation periods, infectious periods, personal hygiene, etc. Whether or not a potential transmission occurs similarly depends upon a large number of factors including closeness and duration of contact, type of interaction, and air temperature.

Rather than fix our algorithm to one set of parameters, we hide the details of how the disease travels within a black box simulator. We only require that the simulator output, for a random progression of the epidemic, a forest rooted at the initially infected nodes. In this forest, every non-root node has a parent from which it acquired the infection. Our later experiments use the EpiFast [14] simulator, which we describe in detail in Section 5.1.

Thus we take as given a population of size n, a probability distribution on who becomes initially infected and when, a simulator which encapsulates all of the other disease parameters, and a number k of 100% effective vaccines to take effect on the d^{th} day of the infection. Our goal is to find a set of people to vaccinate who minimize the expected number of total infections (or a weighted sum of those who

become infected). A exponential algorithm could look at all $\binom{n}{k}$ sets of nodes to vaccinate, and choose the best one, but we want to do better.

3.1.1 Algorithms and Analysis

In order to build up to our algorithm, we begin by contrasting this problem with other optimization problems. Let us start with the work on deterministic, budgeted edge removal by Hayrapetyan et al [54]. They set up a linear program where every node v has an infection level I_v in the interval $[0, 1]$. The algorithm chooses, for each edge e, how much to remove that edge, or R_e. The total edges removed $\sum_e R_e$ must be at most the budget B. Every node's infection level is the maximum over all of its incident edges of the neighbors infection level minus the connecting edges removal level, or $I_v = \max_{u:(v,u)\in E} I_u - R_{u,v}$. The free variables in this formulation are the edge removals, and the optimization is over the sum of the infections. They show how to use random rounding to go over budget by at most a factor of $1/\lambda$ while infecting at most a factor of $1/(1-\lambda)$ more than an optimal solution for any λ. They go on to show how to achieve the same results using a more efficient min-cut based approach.

We encounter three types of problems trying to adapt this approach to a random graph. The first is that good cuts may be passed over because they contain edges which are unlikely to be in the final graph. The second is that it may be infeasible to completely isolate the infection, but removing high degree (or otherwise important nodes) may be sufficient to stop the epidemic. The LP based approach

creates rings of total protection around the initial infections, and has no way of considering this type of solution. Finally, we want to handle cases with a dynamic set of initial infections. It is not clear how to adapt their method to uncertainty in where the epidemic starts.

An intuitively appealing strategy involves simulating the epidemic, and vaccination those with the highest probability of infection first. As an immediate benefit we observe that those with the highest infection probabilities contribute the most direct effect on the total number of infections. Therefore, if network affects could be ignored, this would be the most effective strategy. Additionally, nodes can only pass along the disease if they contract it. Therefore, and once again ignoring the importance of network location, nodes which are more likely to fall ill have a greater chance of spreading the disease.

There are several related drawbacks with this approach. First, a node which often becomes infected but is otherwise isolated could still have a high infection probability. Vaccinating such a node only decreases the expected outbreak size by at most one. Second, it ignores correlations. For example, consider a clique where one of its members connects to an initially infected node. Every node in the clique is almost as likely as the bridge node to become infected, but there is no marginal benefit to vaccinating these other nodes.

This led us to the notion of a set of nodes' criticality - the difference that their mutual vaccination has on the expected outbreak size. This notion aligns nicely with our goal, which we can restate as finding the most critical set of size k. For any fixed set S, we can easily approximate its criticality through simulation. However, since

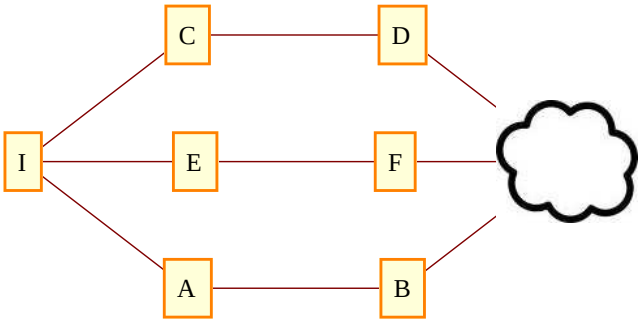


Figure 3.1: Consider the network above where an initially infected node I is connected to the general population (the cloud) by three disjoint paths with edge probabilities of one. The criticality of any strict subset of $\{A, C, E\}$ is zero, while the criticality of the entire set is high. Also, the criticalities of $\{A, C, E\}$ and $\{B, D, F\}$ are both high, but there is no marginal benefit in vaccinating both sets rather than just the first.

we cannot efficiently try all sets of size k, we need another way to find a good set. We evaluated a number of standard methods including greedily building the set and using various matroid techniques [16]. Unfortunately for our purposes, criticality does not compose well. Two sets with large criticality may overlap completely so that their union is no more critical than one of them alone. On the other hand, the union of two sets with extremely low criticality may have high criticality. See Figure 3.1 for specific examples.

Before we present our main algorithm, we present one more alternative which follows naturally from the giant component work of Chung et al. [21]. They show that given a graph with largest eigenvalue λ and each edge kept with probability p, $p = 1/\lambda$ is the cutoff point between a giant component forming or not. They

argue that since $\lambda = \Theta(\hat{d})$, this results can simplify to show that the transition happens when $p = \Theta(1/\hat{d})$. Equivalently we can say that the giant component forms when the second order average expected degree (SOAED), which we define as the sum of the squares of the expected degrees squared divided by the sum of the expected degrees, is one. They show this cutoff for when edge probabilities are uniform, but we build our algorithm under the assumption that a similar cutoff exists for general edge probabilities. This idea leads to a vaccination strategy that tries to minimize the SOAED. We approximate SOAED minimization as well as give an algorithm that could realistically approximated in the real-world by vaccinating people in decreasing order of their expected final degrees.

All of these ideas are helpful, but they still do not necessarily help us find small sets whose vaccination significantly reduces infections. We need to look deeper into what properties such a set must have. To this end, we derive the following theorem:

Theorem 1 *Let a random graph be given with probabilities on each of its directed edges and a probability distribution on which nodes become initially infected. Consider the random process where each edge is chosen independently at random creating a potential transmission graph. From that graph, an initially infected set, and a vaccinated set, a deterministic process creates a particular infection tree by iteratively adding an arbitrary edge which connects an infected node to a susceptible one until no such edges remain.*

Let random variable $X(A)$ be the cumulative number of infections if the set A is vaccinated and random variable S_v be the size of the infection subtree rooted

at node v or 0 if v does not become infected. For any set A of nodes to vaccinate, $\sum_{v \in A} E[S_v] \geq E[X(\{\})] - E[X(A)]$.

Proof of Theorem 1

We prove Theorem 1 though a conditioning argument. Let random variable T be the potential transmission graph, Γ be the set of all potential transmission graphs, and for any $\gamma \in \Gamma$, let $F(\gamma, A)$ be the infection forest resulting from γ and vaccinating the nodes A. Let $|F(\gamma, A)|$ denote the number of nodes in the forest so that $X(A) = |F(T, A)|$. Furthermore, note that the nodes in the forest are exactly those reachable from an initially infected node in the potential transmission graph.

By the definition of the expected value, $E[X(A)] = \sum_{\gamma \in \Gamma} Pr[T = \gamma] \cdot |F(\gamma, A)|$. Similarly, we can express the expected number of nodes protected as $E[X(\{\}) - X(A)] = \sum_{\gamma \in \Gamma} Pr[T = \gamma] \cdot (|F(\gamma, \{\}) - F(\gamma, A)|)$. For a fixed γ, and any node $v \in F(\gamma, \{\}) - F(\gamma, A)$, *all* paths from an initially infected node to v must have passed through A. And thus v must have been in at least one subtree of A.

We can now show the entire bound:

$$\begin{aligned} E[X(\{\})] - E[X(A)] &= E[X(\{\}) - X(A)] \\ &= \sum_{\gamma \in \Gamma} Pr[T = \gamma] \cdot (|F(\gamma, \{\}) - F(\gamma, A)|) \\ &\leq \sum_{\gamma \in \Gamma} Pr[T = \gamma] \cdot \left(\sum_{a \in A} S_a | T = \gamma \right) \\ &= E[\sum_{a \in A} S_a] = \sum_{a \in A} E[S_a]. \end{aligned}$$

■

Theorem 1 can be restated as follows: if there is a set of size k whose joint vaccination protects l people in expectation, than the sum of their expected subtree sizes is at least l. We can take this even further using the pigeon hole principle and say that the minimum expected subtree size is at least l/k.

This property inspires our main vaccination algorithm: vaccinating by average subtree size. Recall that for each infected node our black box simulator outputs who directly infected that node. This creates an infection forest rooted at all of the initially infected nodes. A post-order traversal of this tree allows us to calculate the subtree sizes for each node in linear time. If the problem specifies that vaccines are to be given on day d, then we only count subtrees rooted at day d or later. We repeatedly sample such epidemic simulations to get average subtree sizes that closely approximate the actual expected subtree sizes with high probability. We then select the k nodes with the highest expected subtree size. The algorithm remains the same regardless of the distribution on initial infections, transmission probabilities, etc.

A subtree size approach addresses some of the problems with heuristics discussed previously. It still biases the vaccination towards nodes which are often infected, but unlike vaccination based upon infection probability it places more emphasis on how many active infection paths went through a node. This means that a node at which the epidemic frequently comes to a dead-end will receive low priority. Similarly, nodes which sometimes carry the infection to many others, but for which there are a large number of alternate paths will also receive lower priorities.

Vaccinating by subtree size has some of the same drawbacks too. High priority nodes may be strongly correlated, so vaccinating either could be almost as good as

vaccinating only one. Additionally, a deep epidemic forest can lead to many nodes having high subtree sizes. In this case a small critical set may be obscured by many less important nodes with larger expected subtrees. On the other hand, shallow epidemic trees lead to much smaller subtrees, and thus critical sets which stand out more.

3.1.2 Empirical Results

We perform a large number of simulations in order to empirically validate our subtree size based algorithm compared with the others we discussed, as well as understand how various parameters affect their behavior. For this study we use a dataset approximating the population of approximation 160,000 people in the New River area – that is the area in and around Virginia Tech. Virginia Tech is a large university with roughly 30,000 students plus many additional faculty and staff. It acts as a focal point for this population, perhaps giving it atypical population dynamics – small network diameter, faster mixing, etc.

This datset does not explicitly contain a person-person contact network. Instead it consists of a person-location graph which implicitly gives the person-person graph [38]. A person-location graph is bipartite and connects every individual to the locations they visit throughout the day. Each such edge lists when the person visited that location, and what type of activity they engaged in. From here two people are implicitly connected if they are in the same location at the same time. Maintaining the network as a person-location graph saves on storage space and helps partition

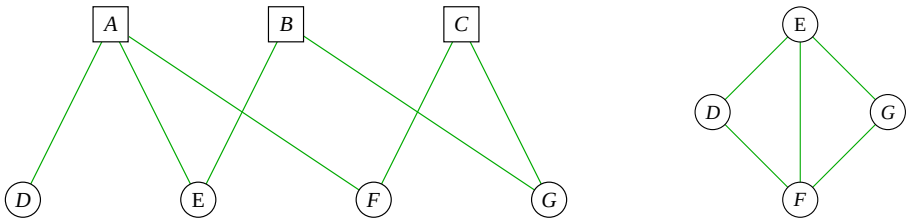


Figure 3.2: The left graph is a person-location graph where boxes A, B, and C are locations and circles D, E, F, and G are people. The right graph represents the corresponding person-person graph.

the data for parallel simulation. Figure 3.2 shows a pictorial representation of a person-location graph and its implicit person-person graph.

While every potential epidemic will behave in a unique way, we focus our results on one particular set of influenza-like parameters. We run the simulated epidemic for 200 days using a transmission probability of $4 \cdot 10^{-5}$ per minute of contact between an infectious person and a susceptible person. Once a person contacts the disease, they remain noninfectious for an incubation period of 1 to 2 days, and then infectious for 3 to 6 days. For each set of parameters, we run 50 simulations to get the statistics which tell us who to vaccinate, and then another 50 to get the results.

In our simulations we study how varying the distribution of initial infections affects the outcome of the epidemic. We vary the distribution in two ways. First, we consider starting from the same set for each iteration versus choosing a new set every time. Using a static set corresponds to knowing an infections current, fixed state and getting the statistics going forward. Alternatively, choosing a new

set each time models not knowing where an epidemic will start. We also examine starting from only 2 infections, which may be small enough to completely contain, and starting from 10 infections, which produces a much more consistent number of final infections. Ten initial infections results in 80k final infections, or roughly half of the total population.

When evaluating different algorithms, we use uniformly random vaccinations as a baseline. Its results are quite poor. In general, up to 10k vaccinations make very little difference. Randomly vaccinating 20k and 40k people saves approximation 15k and 35k respectively. A delay in vaccination and causes no significant change in the final number of infections whether the initially infected set is static or random. The other parameter, whether there are 2 or 10 initial infections has a noticeable effect. With 2 initial infections the epidemic dies out fairly often, which brings down the average number of total infections. If the epidemic does not die out early, the difference in negligible.

Figure 3.3 contains plots for the static initial infections, while Figure 3.4 contains those for randomized initial infections. All three of the test vaccination strategies perform very similarly to each other. With vaccination delays of 20 days or less, vaccinating 20k is sufficient to limit the outbreak to a few thousand cases ($< 1\%$ of the population). In most cases vaccinating 5k and 10k save roughly 10k and 30k respectively. Of particular interest is that in almost all of our cases, the local connectivity based strategy performs slightly better than our other strategies. The containment based strategies using subtree size and vulnerabilities only perform better when the initially infected people can be complete isolated. Just as in the

random vaccination case, the distribution on the initially infected set has little effect on the final outcome except when the disease dies out very quickly, which happens more often when there are fewer initial infections.

3.1.3 Network Properties

Our simulation results demonstrate that the vulnerability and subtree-based heuristics work fairly well, but not quite as well as when we use local connectivity. This would not be true for all graphs, so what properties of the New River area graph lead to this outcome, and are those properties true of similarly sized, realistic contact networks? If most relevant contact networks behave this way, then local connectivity-based algorithms could be very effective in practice. Can we tell when a contact network will have this form, and are there other types of networks where our subtree-based heuristic will perform better?

The New River area consists of Virginia Tech and the surrounding communities. Out of a total population of nearly 160,000, a full 30,000 of them are students at the university [100]. Even without counting faculty and staff, these students make up a very significant fraction of the total population. It is reasonable to assume that a large number of people spend at least some time on campus in a given day. The relatively compact geography of the population means those who do not travel to campus often have friends and neighbors who do. The low average path lengths (most pairs are within 3 or 4 edges of each other) supports this idea.

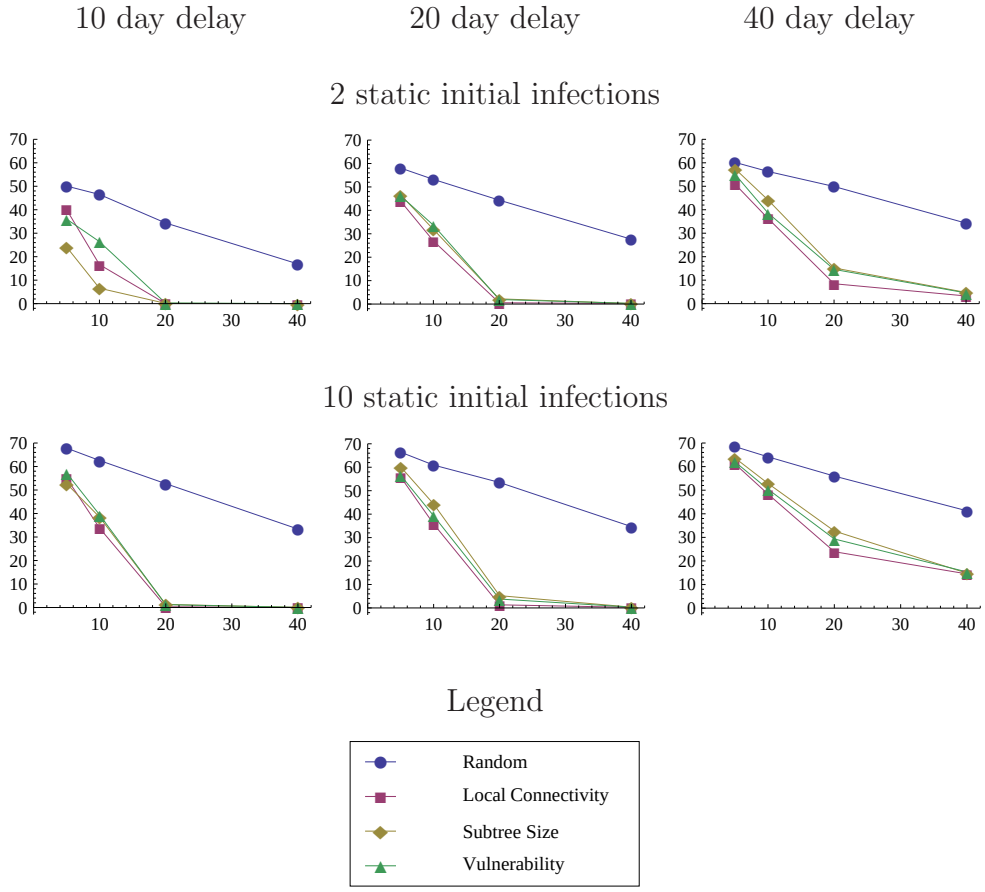


Figure 3.3: This figure contains plots of the mean number of final infections for the four algorithms. The y-axes give thousands of total infections, and the x-axes give thousands of vaccinations. Each individual plot shows the results for a specific vaccination delay and initial infection type. Within a plot, each line corresponds to one of the four vaccination algorithms, and points are plotted for 5k, 10k, 20k, and 40k vaccinations (out of a population of 160k).

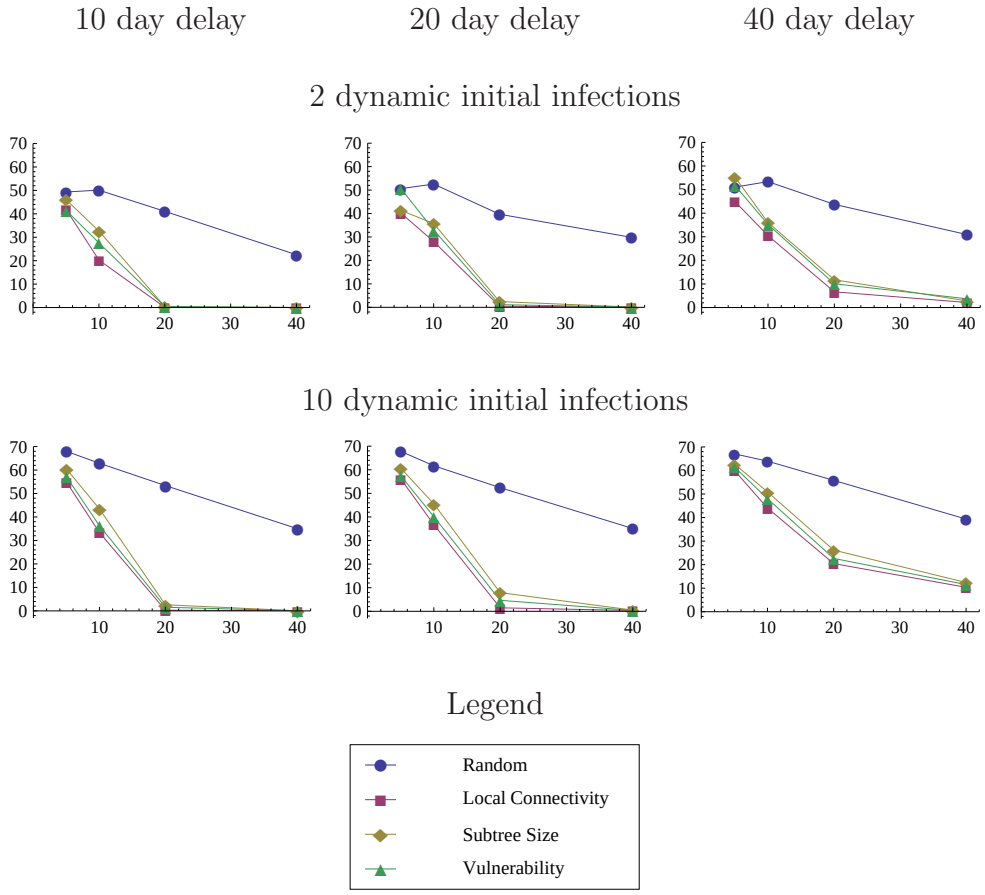


Figure 3.4: This figure contains plots of the mean number of final infections for the four algorithms. The details are exactly the same as in Figure 3.3 except the initial infections in this figure change with each iteration.

We look at several other important graph properties in addition to average distance. For these properties, we consider both the deterministic version of the graph (where all edges are treated equally) and the probabilistic version. For the probabilistic version, we use the same $4 \cdot 10^{-5}$ transmissions per minute as in our simulations. The degree distribution has a long tail with an average degree of 156 with a standard deviation of 131. The mean expected degree is 1.65 with a similarly long tail and standard deviation of 1.33. The second order average degree (SOAD) is 2.72. Spectral graph theory suggests that the SOAD asymptotically approximates the highest eigenvalue, which not too far off at 5.68. To explain our simulation results we turn to the giant component work of Chung et al. [21]. They show that given an unweighted, undirected graph and an edge probability p, that a giant component forms when $p > 1/\hat{d} + \Omega(1)$ and does not form when $p < 1/\hat{d} - \Omega(1)$, where $\hat{d}$ is the SOAD. We restate their tipping point as

$$1 = p \cdot \hat{d} \qquad = p \cdot \left(\sum_v d_v^2 \right) / \left(\sum_v d_v \right) = \left(\sum_v (p \cdot d_v)^2 \right) / \left(\sum_v p \cdot d_v \right),$$

which is the second order average expected degree (SOAED). While their results assume that every edge has the same probability p, the SOAED is well defined for arbitrary edge probabilities. We hypothesize that to stop an epidemic from spreading over a network with high mixing, we need to reduce the SOAED below 1.

To test this hypothesis, we vaccinate the same sets of nodes as in our simulations and calculate the resulting SOAED of the network. Without any vaccinations the SOAED is 2.72 and nearly half of the population becomes infected. All of the targeted interventions reduce the SOAED to close to one with 20K vaccinations

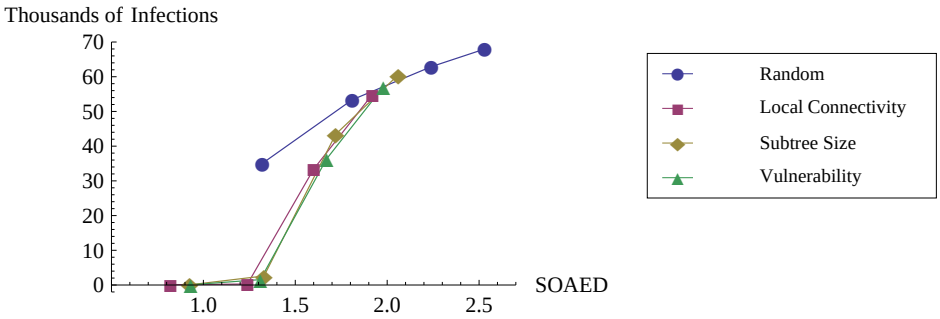


Figure 3.5: Here we show how SOAED and total infections are related for the various vaccination strategies. Our targeted interventions exhibit a nearly linear relationship between SOAED and total infections, with the epidemic completely isolated at roughly with an SOAED as large as 1.25 – somewhat higher than the predicted value of 1.

and below 1 with 40K. Figure 3.5 gives a detailed comparison between SOAED and infections for the case of 10 initial, random infections and a 5 day vaccination delay.

When we prioritize vaccinations by local connectivity, we reduce SOAED slightly more than when using the other targeted interventions. The fact that this approach performs better than the others leads us to believe that the New River area network may be too uniformly connected for there to be any "small" sets whose mutual vaccination protects a large fraction of the network. State or country-wide contact networks almost certainly have lower expansion than those covering a smaller, centralized region, and can thus be exploited better by our subtree-based heuristic. As the size and scope of realistic, synthetic networks expand, we plan to expand our study to those networks as well.

3.2 Sequestering of Critical Sub-Populations

When an epidemic emerges, there is more to safeguarding public health and safety than minimizing the epidemic's direct effects. An important problem that arises in this context is the protection of critical sub-populations such as military and national guard personnel, emergency responders, and public health officials. This problem is different from typical public-health problems where the primary goal is to protect the general population. Sequestering protects smaller sub-populations that are critical to the effective functioning of society during such large-scale crises by isolating them in small groups. Protective sequestering's importance is magnified when effective vaccines and anti-virals are not quickly available, as in the recent H1N1 influenza pandemic [27].

The need for protective sequestration is perhaps best highlighted by considering a military context [35, 66]. Naval ships and military bases contain personnel in confined settings, wherein infectious diseases can be easily transmitted. Personnel can be called up and assigned in a way that is carefully designed to minimize their total infections. This context also provides authorities the level of control necessary to effectively sequester a population. We study how early estimates of individual infection-probabilities can help sequester individuals better in such confined settings. In order to minimize the expected spread of the disease, we develop optimal strategies and discuss their experimental performance and empirical improvements.

In this section we study the problem of protective sequestering of a population of individuals into small, isolated groups, from a stochastic-optimization perspec-

tive. Given *a priori* estimates on the infection probabilities of the people to be sequestered, along with a natural percolation model for disease-spread, and the constraints on individual group sizes, how should authorities place the people into groups in order to minimize the expected outbreak-size of the disease? We give a polynomial-time algorithm for this problem, and discuss some of the tradeoffs involving group size, epidemic transmission rates, and sequestering trigger threshold.

To put the problem into mathematical notation, we are given: a set V of n people (or nodes), and a set of groups with capacities $m_1, m_2, \ldots, m_k$. Groups are rooms or other tightly-constrained locations and are typically small enough that it is natural to assume that the contact graph induced by a subset $V' \subset V$ assigned to a group is complete [91]. Disease transmission is a stochastic percolation process, in which an infected node spreads the disease to each node in the same group with some probability. Each person $v \in V$ could be exposed to the disease even before being sequestered - this is captured by a quantity known as the external infection probability (abbreviated EIP) and denoted as s_v for node $v \in V$. (We assume that these initial exposures happen independently for all $v \in V$: i.e., each $v \in V$ *independently* gets infected initially, with probability s_v. The symbol "s" in s_v denotes "susceptibility".) The EIPs can be estimated by combining computer simulations, demographic characteristics and ground measurements [38]. If a set of nodes V' with size v' is confined to a single group (i.e., if all people in V' get assigned to the same location), a subset U of V' becomes initially infected according to the EIPs defined above; each of these infected nodes could spread the infection to every other node in V' with a transmission probability, denoted by p. As will be

discussed later in Section 3.2.1, disease transmission in the complete graph (within each group) is equivalent to percolation in the Erdős-Rényi random graph $G(V', p)$, and the random set of nodes in V' that finally become infected is the set of nodes reachable in the random-graph from some node in U. The different groups are isolated from each other and therefore do not interact.

Given the above setup and a partition $V_1, V_2, \ldots, V_k$ of the population V (wherein $|V_i| \leq m_i$ for all i, for feasibility), we can consider the *expected* number of finally-infected nodes, where the expectation is taken both over the random choice of the initially-infected set given by the EIPs, as well as the random choices made in the percolation (disease-spreading) process. The goal of the Sequestering problem is to find a feasible partitioning so that the expected number of infections (also referred to as the *outbreak size*) is minimized. The inputs to the problem are the person-to-person transmission probability p as well as the values s_j, for $j = 1, 2, \ldots, n$ and m_i for $i = 1, 2, \ldots, k$; the output is the partition.

If the EIPs vary quite a bit, the partitioning can significantly affect the expected outbreak size. In particular, the natural heuristic of random assignment can perform very poorly. A simple example of this is the following: let $|V| = k^2$ and the group capacities be $m_1 = m_2 = \cdots = m_k = k$. Let $s_i = 1$ for $i = 1, \ldots, k$, and $s_j = 0$ for $j > k$. Assume the disease is very contagious, so that the presence of an infected node in a group will infect everyone else in that group: i.e., p is essentially one. The optimal solution places all of the initial k nodes with $s_i = 1$ into one group, and partition the rest into the remaining $k - 1$ groups - this would have a cost of k. However, a random partitioning and assignment of V to the groups will result in

$\sim k(1-1/e)$ groups having some node $i \leq k$, which results in an expected outbreak of size $\Theta(k^2)$. The optimal solution in this example groups the nodes according to similar EIP values, and our algorithm is based on this idea; see Figure 3.2 for an illustration.

What follows is a brief summary of our specific contributions. In Theorem 2 and Corollary 3, we prove the following property which is crucial for the algorithm's correctness: for any two groups G_1 and G_2 in a given optimal solution, either all s_v for $v \in G_1$ are less or equal to all s_u for $u \in G_2$, or the other way around. Then we give the algorithm's details. Suppose there are r distinct elements from among the m_i – i.e., there are r distinct group-sizes – and that u_i is the number of groups with the i^{th} smallest size. Thus we have $\sum_{i=1}^{r} u_i = k$. All groups of the same size and transmission rate have the same "type". We develop an algorithm that is polynomial in n and m, and exponential in the number of types r. We extend our algorithm to answer the question of when to leave some of the critical population unsequestered. We follow-up with an analysis of the algorithm's running time in Section 5.2.1. A direct implementation has a running time of $O(\sum_i m_i^5 + n \cdot (\sum_i m_i^2) \cdot \prod_i u_i)$ and space complexity of $O(\max_i m_i^3 + n \prod_i u_i)$; in the case of a single group type, with all rooms being of size m, we can improve the space complexity to $O(m^3 + \frac{n^{1.5}}{\sqrt{m}})$. We conclude by running our algorithm on several artificial datasets and report on the results. We observe as much as 50% improvement in the outbreak sizes compared to a random partitioning. Furthermore, we show that our model is fairly robust to EIP estimation error in that errors in EIPs lead to small changes in expected outbreak size.

Our results show that for diseases with high reproductive numbers (R_0), very small EIPs can lead to large outbreaks within sequestered population. This suggests that to be effective, critical workers should be sequestered very early on during an outbreak. The group sizes and the time to enforce sequestration are both important logistically as well as socially – from a logistical standpoint, one would like to keep the group sizes large, from a societal standpoint, one would like to sequester individuals as late as possible. Our results, were based on a number of implicit assumptions that are nevertheless defensible. First, we assume that individuals who get infected are removed from the sequestered population in a predictable way. Removal is possible only after the individual is symptomatic; in case of flu-like diseases, this usually happens a couple of days after the individual is infectious. We incorporate this time into the transmission probability p. Second, we assume that EIPs can be effectively estimated. EIPs capture the uncertainty inherent in this complex process; the sensitivity studies reported in the experimental results section show that our results are fairly robust. Third, we assume completely mixed groups with uniform transmission probabilities. Again while not completely true it appears to be a reasonable assumption for flu-like illness.

3.2.1 Preliminaries

We develop a combinatorial model for the sequestering problem in the rest of this section. Let V denote a set of people (also referred to as nodes), who need to be sequestered within a base that has k groups with sizes $\{m_1, \ldots, m_k\}$.

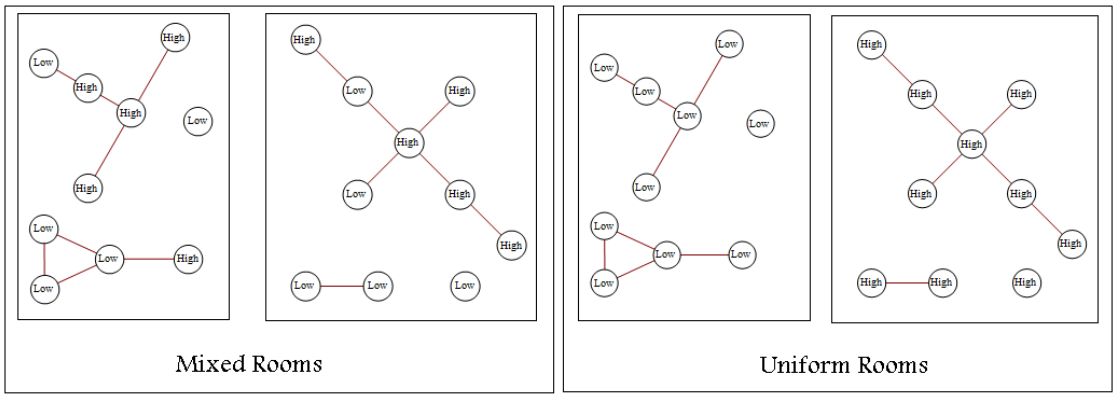


Figure 3.6: Here we see two partitions of a population into two groups each. The nodes represent individuals and are labeled "'high"' or "'low"' based upon the person's EIP. For this example think of "'high"' being close to 1 and "'low"' close to 0. The edges represent random disease transmission paths. In these instances, an outbreak spreads from any externally infected person to all others in the same connected component. In the left example, the population is divided into groups randomly. All but 4 people are connected to those who are likely infected. In the right example, there is a low EIP group and a high EIP group. Here all 10 people in the low EIP group are likely to remain healthy.

The goal is to partition V into groups $V_1, \ldots, V_k$, so that $|V_i| \leq m_i$. Naturally, we need $\sum_i m_i \geq |V|$ for a feasible solution. We are given an external infection probability s_i for each $i \in V$, which means that node i is infected with probability s_i. As discussed earlier, we assume that these probabilities have been estimated through epidemic simulations and demographic analysis of the population. However, other individual differences in how people respond to a disease are not as obvious and may not be possible to determine until after the fact. Therefore we make the simplifying assumptions that everyone has the same resistance characteristics, and that external infections happen independently. Many of our results hold under the weaker condition that within any group, individuals resistance characteristics are all drawn from the same distribution, however the stricter assumption simplifies the presentations.

We assume that the contact graph $G[V_i]$ induced by the set V_i assigned to a group is a clique, i.e., any pair of nodes $u, v \in V_i$ come in contact, which is quite reasonable for small groups. We assume the SIR model of disease transmission [91], in which nodes are initially either *Susceptible* or *Infectious*. Each infected node u spreads the disease to each susceptible neighbor v with probability $p(u, v)$, and then transitions into the *Recovered* state. For simplicity, we assume a uniform probability p in most of this paper, though some of the results can be extended to the general case. The spread of the disease stops when all the nodes are either susceptible or recovered, i.e., there are no infectious nodes. The disease transmission process is equivalent to "bond percolation" in the Erdős-Rényi random graph model, where each edge is chosen with probability p [91]. Within a group formed by the set S

of people, a random subset $S' \subseteq S$ obtained by picking each node $u \in S$ with probability s_u is the initially infectious set, and the remaining nodes, $S \setminus S'$ are all susceptible. By the percolation process, the (random) set of nodes in $S \setminus S'$ that become infected is the set of nodes reachable from S in the random subgraph $G(S, p)$. Let X_S denote the final number of infections within this group. The goal of the sequestering problem is to find a feasible partitioning $V_1, V_2, \ldots, V_k$ so that the expected outbreak size $\sum_i E[X_{V_i}]$ is minimized.

3.2.2 An Efficient Algorithm for Sequestering

We now describe our main algorithm for optimal sequestering. As discussed earlier, there are two natural contrasting heuristics for grouping people: (i) load-balancing–type heuristics where we try and keep the total carrier probabilities approximately the same across the groups, which is usually well-achieved by a random partitioning, and (ii) where people with high carrier probabilities are all grouped together where possible: i.e., viewing the objective of number-of-infected-people as something like a concave function. As mentioned earlier, the former heuristic can lead to sub-optimal assignment, and our algorithm is based on a refinement of the latter heuristic. We start with the following "well-ordered" property of an optimal solution, and then discuss how this leads to a natural dynamic programming algorithm.

A very useful property of the optimal assignment is that it is well-ordered, which we define as follows: for any two groups G_1 and G_2 in a given solution, and

people $u, w \in G_1$ and $v \in G_2$, if $s_u < s_v < s_w$ then swapping either u and v or v and w will not increase the final expected cost; this is summarized in the following theorem.

Theorem 2 *For any two groups and a subset of the people to be assigned to these two groups, there exists an optimal partition where all* EIP*s in one group are less than or equal to all* EIP*s in the other.*

Proof of Theorem 2 For any person a let I_a denote the probabilistic event that a is externally infected (thus $s_a = Pr[I_a]$). For any set of people S, define the random variable X_S to be the number of final infections among the people S. Similarly for any set of people S, define $Y_S = E[X_{S\cup\{a\}}|I_a] - E[X_{S\cup\{a\}}|\overline{I_a}]$ where a is any person not in S. Intuitively, Y_S is the expected marginal number of infections caused when an additional person added to the group S is externally infected versus when the additional person is not infected. Using the linearity of expectation we have that $E[X_{S\cup\{a\}}] = Y_S Pr[I_a] + E[X_{S\cup\{a\}}|\overline{I_a}]$. This means that the net cost from swapping a from a group with S for another person a' from a group with S' is

$$E[X_{S\cup\{a'\}}] + E[X_{S'\cup\{a\}}] - E[X_{S\cup\{a\}}] - E[X_{S'\cup\{a'\}}] = (Pr[I_{a'}] - Pr[I_a])(Y_S - Y_{S'}).$$

There are two parts necessary to prove the theorem. The first is that if two people a, b are in a group with additional people S, then $Pr[I_a] \leq Pr[I_b]$ implies $Y_{S\cup\{a\}} \geq Y_{S\cup\{b\}}$. We show this by conditioning on the connected components of the group. Define random variable C to be a connected component decomposition (a set of disjoint subsets whose union is the entire group) of a the group $S \cup \{a, b\}$. Assuming $Pr[I_a] \leq Pr[I_b]$ we have:

$$
\begin{aligned}
Y_{S\cup\{a\}} =& E[X_{S\cup\{a,b\}}|I_b] - E[X_{S\cup\{a,b\}}|\overline{I_b}] \\
=& \sum_C Pr[C] \sum_{c\in C, b\in c} \left(E[X_c|I_b] - E[X_c|\overline{I_b}]\right) \\
=& \sum_C Pr[C] \left(\sum_{c\in C, b\in c, a\notin c} \left(E[X_c|I_b] - E[X_c|\overline{I_b}]\right) \right. \\
& \left. + \sum_{c\in C, b\in c, a\in c} |c| \left(1 - \prod_{x\in(c-\{b\})} Pr[I_x]\right)\right) \\
=& \sum_C Pr[C] \left(\sum_{c\in C, a\in c, b\notin c} \left(E[X_c|I_a] - E[X_c|\overline{I_a}]\right) \right. \\
& \left. + \sum_{c\in C, a\in c, b\in c} |c| \left(1 - \frac{Pr[I_a]}{Pr[I_b]} \cdot \prod_{x\in(c-\{a\})} Pr[I_x]\right)\right) \\
\geq& \sum_C Pr[C] \sum_{c\in C, a\in c} \left(E[X_c|I_a] - E[X_c|\overline{I_a}]\right) \\
=& Y_{S\cup\{b\}}.
\end{aligned}
$$

The second part is that in an optimal solution, if a and b are in different groups along with S_a and S_b respectively, then $Pr[I_a] < Pr[I_b]$ implies $Y_{S_a} \geq Y_{S_b}$. This can be seen easily because the cost of swapping a and b is $(Pr[I_a] - Pr[I_b])(Y_{S_b} - Y_{S_a})$. If the original partition was optimal, the change in cost incurred by the swap must be at least 0. Since by assumption $Pr[I_a] - Pr[I_b] < 0$, $Y_{S_b} - Y_{S_a}$ cannot be positive.

Taken together these give that for any a, b, c with $Pr[I_a] < Pr[I_b] < Pr[I_c]$ with a and c in the same group with S_{ac} others and b in a different group with S_b in an optimal solution, $Y_{S_{ac}\cup\{a\}} \geq Y_{S_{ac}\cup\{c\}}$. This follows because a and c are in the same group and $Y_{S_{ac}\cup\{a\}} \leq Y_{S_b} \leq Y_{S_{ac}\cup\{c\}}$ because the group assignment is optimal.

Since $Y_{S_{ac}\cup\{a\}} \leq Y_{S_b} \leq Y_{S_{ac}\cup\{c\}} \leq Y_{S_{ac}\cup\{a\}}$, the three quantities must be equal and any two of the three people can be swapped while keeping the cost optimal. Given any optimal solution, as long as there exists such a, b, c we can continue making these swaps, moving those with higher EIPs to the right, until an optimal solution of the desired form is reached. ■

When there are more than two groups, applying Theorem 2 to all pairs of groups yields the following corollary:

Corollary 3 *Given any set of groups and a set of people with known* EIP*s, there exists an optimal partition which orders the groups, and if group i comes before group j than all of the* EIP*s in i are less than or equal to all of the* EIP*s in group j.*

Theorem 2 tell us that an optimal solution keeps people with like EIPs together, however it does not say when to stop filling one group and start on the next - this can be determined by dynamic programming. We develop a dynamic programming algorithm whose running-time and space bound are exponential in the number of group-sizes r but polynomial in both the number of people and the total number of groups. Our algorithm, which heavily exploits the symmetry of infectivities is detailed in Figure 3.7 for the primary case in which all group capacities are uniform.

Algorithm SEQUESTER requires the function $g(S)$, which is the number of infections if the set S forms a group. In Figure 3.8, we describe a dynamic programming algorithm to compute $g(S)$; it can also be estimated arbitrarily well by Monte-Carlo simulations. In fact, if the disease response characteristics are not uniform across a population, exact calculations may not be feasible and such simu-

lations may be necessary. The proof of optimality of Algorithm SEQUESTER follows from straightforward induction on the dynamic programming arrays which show that it produces an optimal well-ordered partition and Theorem 2 and Corollary 3, which show that an optimal well-ordered partition is an optimal partition.

Also, in Figure 3.7, Algorithm SEQUESTER is described for the setting in which all group sizes are uniform. This algorithm can be extended to handle r group types (where a type i is given by a capacity m_i and a value of p) by making OPT into an $r+1$ dimensional array and computing

$$OPT(a, b_1, \ldots, b_r) = \min_{j=0}^{r} \min_{i=0}^{m_j} [OPT(a-i, b_1, \ldots, b_j-1, \ldots, b_r) + g(\{a-i+1, \ldots, a\})].$$

Note that our analysis requires all of the direct transmission probabilities within a group to be equal, but it does not require different groups to have the same probabilities. Therefore Theorem 2 applies even when transmissions within one group are more likely than those in another group. This may arise frequently as one would expect different levels of sanitation, contact time, or symptom monitoring in different locations.

An interesting feature of our algorithm is that it not only determines how to optimally partition a critical subpopulation, but it can also be used to determine when it is better to leave some critical individuals unsequestered where they will presumably become infected. For example, if there is one group available for sequestering, a single individual with a high EIP, and many with low EIP, then leaving the high EIP person out of the group can result in more of the critical population remaining healthy. In this case the person who is a likely carrier is left behind to

protect the rest of the critical population. We address this problem of incomplete sequestering by adding an additional group where everyone assigned to that group becomes infected. This new group represents those who are not sequestered. The optimal partition produced by our algorithm for this modified instance, minus the group where everyone becomes infected, is an optimal incomplete sequestering for the original problem.

3.2.3 Experimental Analysis

In this section we examine the behavior of our algorithm when applied to a few artificial datasets. We look at three specific aspects:

- How much better is an optimal solution than a random solution?
- How sensitive are our results to errors in the EIP estimates?
- What is the time and space usage of an implementation in the C programming language?

First we study how the EIP estimates can be used by our algorithm. We evaluate the effectiveness of our algorithm, relative to a random assignment which does not use the EIP estimates, and find that the random assignment could lead to outbreaks which are twice as large as our algorithm or larger. Next, we study the sensitivity of our algorithm to the accuracy of the EIP estimates, and find that it is fairly robust. Thus, the EIP estimates provide valuable information to policy planners.

In our study, we assume that a critical population (e.g., the military) is sequestered at a base a few days after the onset of an outbreak of a simulated disease in the general population. We assume that the EIP estimates of such individuals are known; and for our simulations we assume the EIPs are exponentially distributed. For a given maximum group size, we find the optimal assignment using our algorithm, and compare it with a random assignment. Figure 3.2.3 shows a histogram of the ratio of an optimal solution to a random solution for a large number of simulations using a variety of values for p and quantized exponential like distributions for the EIPs. The optimal sequestering by our algorithm is up to 50% better than a random sequestering, and often at least 25% better. We see the best improvements at moderate disease transmission probabilities, and the worst at the extremes of low or high transmission probabilities. This is because moderate transmission probabilities permit the most room for improvement. With low enough probabilities, connected components are very small, effectively isolating sick individuals from others in the room. When probabilities are large enough, rooms become almost fully connected, and most of the population falls ill despite our best efforts.

This sequestering scenario has an obvious susceptibility to errors in the EIP estimates. For example, if the EIP estimates are no better than random, an optimal sequestering with respect to those estimates would be essentially random with respect to the true values. To examine the effect of estimation error we performed a number of simulations where we found the optimal sequestering assignment for a set of EIPs, and then compared that assignment's cost to the cost of the same assignment but with the EIPs perturbed. The first way we perturbed the EIPs was to

change each one by up to 30% while keeping the sum of the EIPs constant. This set was designed to measure random, directionally symmetric errors. In all of our simulations the relative change (absolute value of perturbed minus optimal divided by optimal) was at most .0012. This tells us that our assignments are very insensitive to random, symmetric errors.

We also examine the effect of one sided estimation error. Using the same sets of parameters from the symmetric case, we separately examine the effect of increasing and decreasing the EIPs after an optimal assignment is made. When we increase all of the EIPs by 30%, we see a maximum increase over the estimated values in the expected infected set of at most 23.4% with an average increase of 8.7%. When we decrease the EIPs by 30% we see a decrease in the expected infected set of at most 26% with an average decrease of 11%. While the 30% increase is an upper bound (in general a factor of x bound on EIP errors yields at most of factor of x bound on epidemic size error because in any single connected component two nodes with increases EIPS will partially cancel each others increase out), it is interesting to note that in both cases the effect of one sided error is significantly less than the factor of perturbation of the EIPs. A full plot of the scaling factors in our simulations is presented in Figure 3.10. In summary, neither under nor over-estimating the EIP values has a compounding effect on our results, and symmetrical incorrect estimates cancel out in practice. Therefore our sequestering scheme is fairly insensitive to estimation errors.

We conclude this section by empirically showing some of the trade-offs involved in an implementation of protective sequestering. Experiments in this section

use EpiFast [14]. To establish the statistical properties of the system's range of behavior, an epidemic outbreak over a large population is simulated for each experimental case for 50 iterations with identical conditions and different random seeds. Infections within the essential subpopulation ($\approx 188K$ individuals) are tallied for each day of a 254-day epidemic. This gives us, for every day t, the distribution of the number of infections within the socially essential subpopulation on that day. This is used to estimate the EPI for a given individual on any day t. We then simulate a triggered decision to sequester the protected population on each successive day t and compute the total number infected in that subpopulation during the entire epidemic as a consequence of this decision. This amounts to a controlled representation of a decision to trigger sequestering at that day. It allows us to establish the effects of what different threshold triggers would be, had they been given a priori and used by the authorities. Therefore we can experimentally compare *in silico*, the effects of different triggers.

Next, we consider group size. Because the number of people in a sequestered group defines the impact of sequestering an infectious person in that group, we include a sweep of this factor in our experimental design. In the simulation, we sequester the protected population into group sizes of 30, 50, or 70. Transmissibility of diseases vary, so the experimental design also factors person-to-person transmission rates (defined as the probability per unit of contact time that an uninfected person will catch the disease from a nearby infectious person) of 0.05 and 0.1. These values correspond roughly to infection incidences of $2.52 * 10^{-3}$ per hour over 20 and 40 hours respectively. To compute the number of infections if we start sequestering

on day t, we run our optimal partition algorithm on the subpopulation given the EPIs for that day. We then add the expected number of people infected within the sequestered groups to the number infected before sequestering began. The results are shown in Figure 3.11.

These results show that sequestering is most effective when triggered before the disease has spread very much (when EPIs are low) and when the outbreaks within groups are likely to be small (when the transmissibility times group size is small). In fact, unless these two factors are kept small enough, sequestering may lead to more infections within the critical subpopulation. These plots also suggest a trade-off between group size and latent infection rate. We examine this trade-off in greater detail in Figure 3.12.

Figure 3.11 shows the effect of sequestering on a given day of an epidemic and demonstrates that sequestering early can protect a subpopulation. However, although there are implicit infection rates at the time of initiating the sequestering, it does not show what happens when we start sequestering based only on a criterion trigger threshold and surveillance measurement of the incidence of infection. Since that is an inherent aspect of any real-world sequestering plan, here we extend the analysis to this situation. In order to better understand a realistic setting, we simulate a number of scenarios varying group size, the sequestering trigger, the level of vaccination, and the number of days each individual is placed in a "personal quarantine" before being admitted to a group. The simulated vaccine is poorly matched and reduces the chance of infection by only 30%. We trigger sequestering based upon the rate of illness within the socially essential subpopulation being either

0.5%, 1.0%, or 1.5%, (corresponding to days 89, 100, and 108 in the unvaccinated base reference case). The main results are shown in Figure 3.13, where the labels on the x-axis (size, trig, vax, and qd) refer to the size of the sequestered groups, the levels of infection that triggered the sequestering, the proportion of the subpopulation that was vaccinated, and the number of quarantine days prior to admission to a sequestered group respectively.

Group sizes had the greatest impact, followed by the trigger threshold. Next is the vaccination level, which is the only factor that affects the final attack rate when the subpopulation is not sequestered. The least important factor was the number of days of quarantine prior to placement within sequestered groups. The intra-group transmission rates were calculated based upon a transmission probability of $2.52 * 10^{-3}$ per hour of contact (when both individuals are unvaccinated) and an average of 16 hours of contact time between when an individual becomes infectious, and when they are diagnosed and removed from the population.

At one extreme, these results show that sequestering a subpopulation can effectively protect them provided group sizes are small and sequestering is implemented when only a few individuals in the subpopulation are infected. In our subpopulation, using a sequestering group size of 30, with or without vaccination, and even with the relatively late trigger, only a small number of individuals become infected. This small group size limits the attack rate to an average of 15.6% across the 3 other factors. For group sizes of 50, as long as sequestering is triggered before 1.5% of the subpopulation is ill, sequestering remains an effective strategy (reducing attack rates to an average of 17.9%). This remains true even when group sizes reach 70 if the

sequestering is triggered early enough (at 0.05%, which occurs on average 11.7 days earlier than 1%). This trend demonstrates how sequestering the socially essential subpopulation with fewer latent infections reduces the number of outbreaks within groups. Setting a lower trigger threshold for sequestering is the primary means for accomplishing this, with temporary individual quarantining being another, less effective and likely impractical technique. In our example subpopulation (without individual quarantine), the only way to keep the total infection rate below 10% involves sequestering at a threshold of 1% while maintaining group sizes of 50 or smaller. At the other extreme, our results also show that under certain conditions sequestering can be considerably worse than doing nothing. Having groups so large that statistically most will contain latently infected individuals and/or transmission rates large enough to create outbreaks infecting nearly everyone in a group, result in a large fraction of the sequestered subpopulation becoming infected. For instance with group size 70 and triggering sequestering after 1.5% are ill, we observe attack rates that are on average 15% *higher* (regardless of vaccination and quarantine) than if no sequestering had occurred. Waiting too long to commit to the decision to sequester and investing too little in sequestering preparations will have bad outcomes.

The results further suggest that we can group our containment strategies into two categories: those that reduce the size of outbreaks within groups and those that reduce the number of such outbreaks. Group size and vaccination level naturally fall into the former category. The group size provides an upper bound on the size of an outbreak (since an infection in one group cannot spread to another), but group size

and person-to-person transmission rate have a more subtle effect as well. We know from the seminal work of Erdős and Rényi [37] that when the product of group size and transmission rate is less than one, these outbreaks tend to be small (no more than $O(\log(\text{group size}))$); when this product is greater than 1, the outbreaks tend to approach the entire group size. This effect can be seen in our results since the number of total infections stays small until the size-rate product approaches 1, at which point many individuals start to become infected within their group.

Algorithm SEQUESTER

Given: set $V = \{1, \ldots, n\}$ of people, EIPs s_i for each $i \in V$, and k groups of size at most m

Output: partition of V into groups of size at most m, so that the final expected outbreak size is minimized.

1. Define $OPT(a, b)$ to be the expected number of finally-infected people, in an optimal solution for the problem restricted to the people indexed $\{1, 2, \ldots, a\}$, and using groups $\{1, 2, \ldots, b\}$ (for any a, b that satisfy $a \leq mb$). Let $g(S)$ denote the expected number of infections if the group of individuals S is put in one group.

2. Sort the people in V, such that $i < j \rightarrow s_i < s_j$.

3. For all b set $OPT(0, b) = 0$.

4. For $b = 1, \ldots, k$ and for $a = 1, \ldots, n$, compute

$$OPT(a, b) = \min_{i=0}^{m} [OPT(a - i, b - 1) + g(\{a - i + 1, a - i + 2, \ldots, a\})].$$

5. $OPT(n, k)$ gives the expected infection size of an optimal sequestering, and tracing back through OPT reveals the partitioning which achieves that value.

Figure 3.7: Algorithm SEQUESTER for the simplest setting in which all allowed group capacities are uniform, though the final group sizes need not be uniform. The algorithm is a dynamic program, based on the recursive expression for the optimum.

Algorithm for computing $g(S)$ exactly.

Given: set $S \subseteq V$ forming a single group

Output: the expected number of infections, $E[X_S]$, in S.

1. Initialization the array $P(j, x, y)$ to contain the probability that there are x infected nodes at distance at most j from one of i initially infected nodes, and y nodes at exactly distance j in a $G(|S|, p)$ random graph.

2. Initialize $P(0, i, i) = 1$ and $P(0, *, *) = 0$ for all other entries.

3. For each j from 1 to $|S|$,

   $$P(j, x, y)$$
   $$= \sum_{0 \leq z \leq x-y} P(j-1, x-y, z) \binom{\ell - x + y}{y} ((1-p)^z)^{\ell - x} (1 - (1-p)^z)^y.$$

   Save the array $B(i) = \sum_{x=0}^{|S|} x \cdot \sum_{y=0}^{|S|} P(|S|, i, y)$.

4. Upon each invocation of $g(S)$, compute $A(i, j)$ – the probabilities that there are i initial infections among the first j people.

5. $A(0, 0) = 1$, $A(*, 0) = 0$ otherwise, and $A(i, j) = A(i, j-1) \cdot (1 - s_j) + A(i-1, j-1) \cdot s_j$. Computing A in this way computes values for several subsets at once, and they can be stored between calls to g.

6. $g(S) = \sum_{i=1}^{|S|} A(i, |S|) * B(i)$.

Figure 3.8: Algorithm for computing $g(S)$ exactly, though in practice a Monte-Carlo estimate can be used.

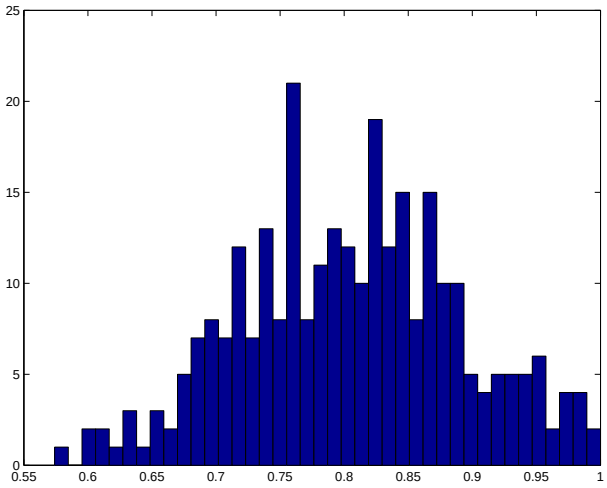


Figure 3.9: Histogram of optimal sequestering's ratio of expected infection size over that of random sequestering.

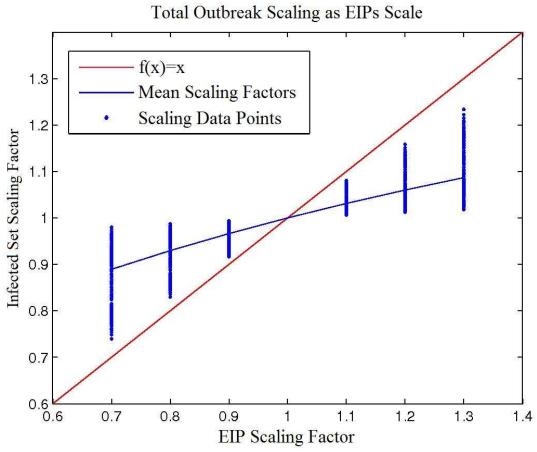
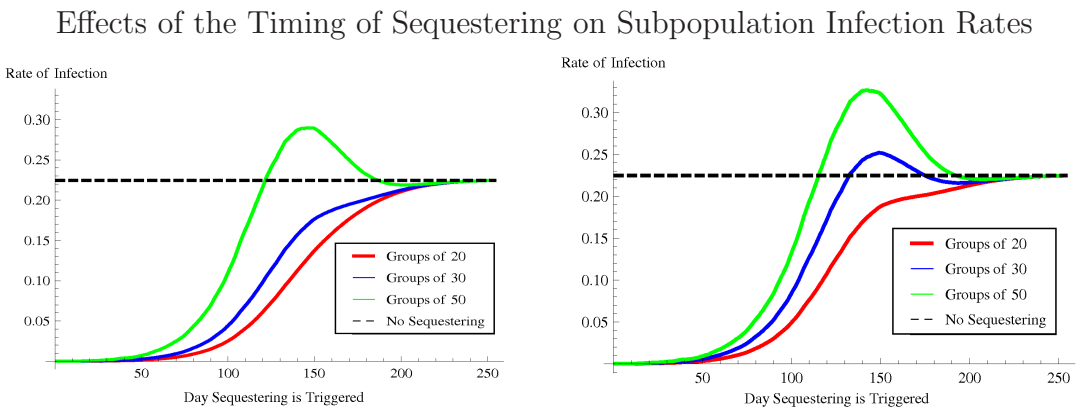
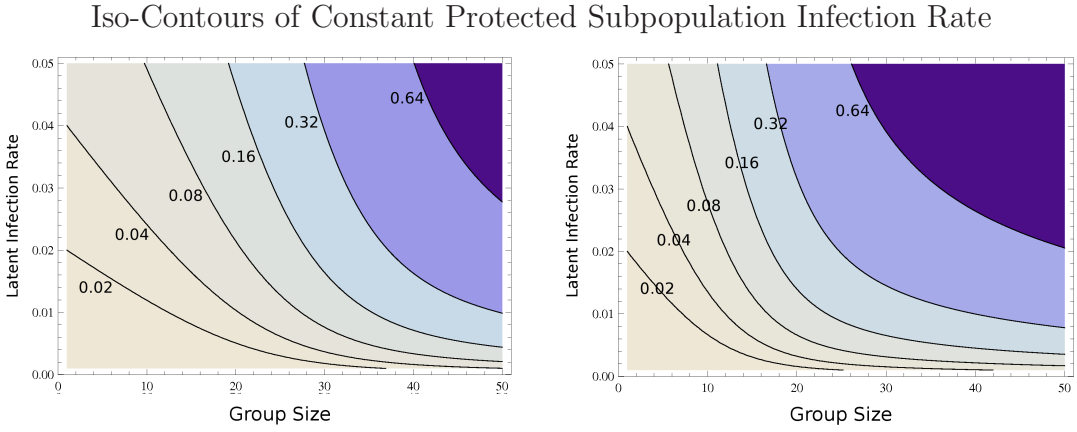


Figure 3.10: Outbreak scaling as a function of EIP scaling.



$p = 0.05$ $p = 0.1$

Figure 3.11: For every day of a simulated epidemic, these plots show the fraction of the socially essential population that get sick during the epidemic, if we start sequestering on that day. Group sizes of 20, 30, and 50 are shown, along with the baseline case without sequestering. Transmissibilities on the left are $p = 0.05$ and $p = 0.1$ on the right.



$p = 0.05$ $p = 0.1$

Figure 3.12: The contour lines indicate equal infection rates as group size and latent infection rate vary. The key observation here is that the higher the latent infection rate, the more important group sizing becomes. If we trigger sequestering late, we can make up for it to a point, but only with significantly smaller group sizes or settling for much higher infection rates.

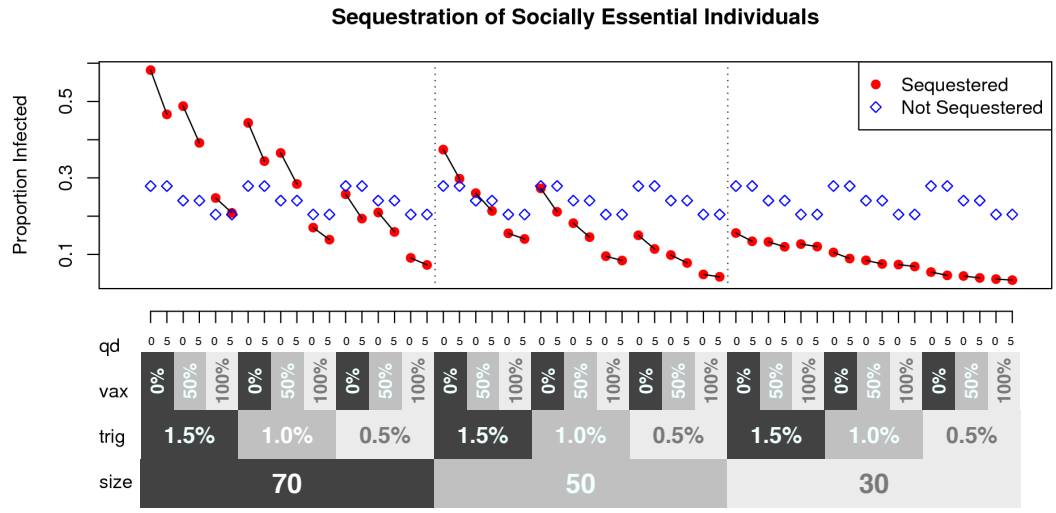


Figure 3.13: Effects of various interventions following sequestering. The proportion of individuals infected in the non-sequestered population (blue diamonds) and sequestered population (red dots) are plotted vs. which optional features are applied to the sequestered group. The features are: the size of the sequestering group (size), the threshold at which sequestering is triggered (trig), the proportion of subpopulation vaccinated (vax), and the number of quarantined days following sequestering (qd). These interventions are ordered by their impact on the final proportion infected, size having the largest impact and qd the least.

3.3 Random Edge Removal and Network Degree Sequence

In the previous two sections we developed and examined optimization algorithms for fighting epidemics. In this section, we step back from that specific application and focus on how random edge removals affect a more fundamental graph property.

Many processes can cause the edges of a given graph G to fail independently, say with identical probabilities. This classical idea captures the original random-graph model $G(n, p)$ of Erdös & Rényi [36], with the host graph G being the complete graph. This topic has received much attention recently in the case where the host graph G is *not* complete, but comes from some interesting family of graphs. Motivations for studying random edge-removal are many: e.g., link failure in peer-to-peer networks [98], instant-messaging or the fact that we can only sample (random) subgraphs of massive graphs [21], pruning of relationships in online social networks, and disease propagation or network attacks [81].

Many properties such as diameter, emergence of the giant component, and various spectral parameters (e.g., spectral gap, mixing, expansion) have been studied when edge removal is conducted on graphs from various natural families [21, 36, 20, 19, 2]. Another natural parameter of a graph family is its degree sequence: indeed, much work has focused on various properties of random graphs with given (expected) degree sequences [22, 25, 24]. For instance, Chung & Lu [23] show how the size of such a random graph's likely giant-component depends on the average expected degree and the second order average degree.

The key role often played by the degree sequence leads us to the question: what happens to the degree sequence of a graph family when edges are removed independently with identical probabilities? If the initial graph has $d(j)$ vertices of degree j and edges are each removed with probability p, there are essentially $d'(k) \doteq \sum_{j=k}^{\infty} d(j) \cdot \binom{j}{k} \cdot p^{j-k}(1-p)^k$ expected nodes of final degree k. We provide a characterization of how this new distribution relates to d for most interesting graph families. We also provide concentration bounds that show that the degree sequence matches the sequence d' with high probability.

Let us start by discussing two basic types of degree sequences. It is known that real-world, large-scale graphs often have a degree structure similar to power-law. These graphs include the World Wide Web [10, 11, 68], internet routers [39], many social networks including scientific co-authorship [9], as well as biological networks such as protein-protein interaction graphs [101]. Some graphs previously identified as being power-law may have degree sequences closer to stretched exponential, log-normal, or other. For a thorough examination of many empirical datasets, see Clauset et al. [26]. We will focus on power-law sequences as a concrete example, and later show that the analysis works for essentially all other interesting sequences as well.

Roughly, a power-law graph has many nodes with each low degree and a few nodes with each high degree. A graph has a power-law structure if there exist parameters α and γ such that for essentially all k the number of nodes of degree k is $\sim \alpha k^{-\gamma}$. To accommodate small deviations, we define a family of power-law-like networks such that for some interval I of degrees, the number of nodes of degree

$k \in I$ is within a constant factor of power-law (i.e., $\Theta(\alpha k^{-\gamma})$). What happens to the degree structure of the graph family when edges are removed independently, with some probability p? Does a power-law-like graph remain power-law-like? If so, how do its parameters vary, and how good is the concentration of the number of final nodes with some degree k? What about graphs with exponential degree sequences, where the degree sequence decays exponentially starting at some minimum degree of interest?

Callaway et al. [17] address some of these questions using generating functions to show how random edge removal affects the existence of the giant component for original random graphs of arbitrary degree distribution. Martin et al. [81] is a more direct predecessor of our work. They considered the special case of $p = \frac{1}{2}$ and $\gamma = 2$ for power-law graphs, and derived empirical results demonstrating that removing edges with probability $\frac{1}{2}$ from a power-law graph gives an expected new degree structure which is close to power-law. More recently Cooper and Lu [29] have shown that only power-law distributions are "scale-free", in the sense that a random subgraph of a power-law graph is likely to be "scale-free" also. (Technically, the work of [29] is on "site percolation", wherein vertices are removed randomly, while those of [17, 81] and ours are on "bond percolation", wherein edges are removed. Because our bounds for $E[d_k]$ rely on a sum over the expected contribution from each node and every node not removed by site percolation has the same behavior as it does in bond percolation, our qualitative statements about expected degree sequences apply to the bond percolation context as well.)

In Sections 3.3.1 and 3.3.2 we prove that after random edge removal power-law-like networks retain expected degree sequences that are power-law-like with the same value of γ. Then we go on to show how the value p changes the exponent in exponential degree sequences. We get explicit, analytical bounds by algebraically combining a binomial distribution, a step function, and the degree distribution. These results are in contrast with the three works mentioned above, as follows. Given the general generating-function bounds of [17], one needs to evaluate numerically the parameters of interest, in some cases by iterated numerical evaluation. The main results of [81] are largely empirical, and focus on the case where $p = \frac{1}{2}$ and $\gamma = 2$. Finally, the generating function based work of [29] requires that the graph have *bounded* degree D, and that for all $k \leq D$, the number of nodes of degree k is *precisely power-law* up to a lower-order term: i.e., $(c + o(1))nk^{-\gamma}$ for some constant c, whereas we allow a multiplicative-constant deviation from strict power-law behavior. (We also improve upon [29] in a tail bound, as described below. Note, however, that given a precise "$(c + o(1))nk^{-\gamma}$" bound as above, the elegant work of [29] shows that only such degree sequences remain purely "scale-free" under edge removal.)

In Section 3.3.3 we give a unified view of how degree distributions change with edge removal. Specifically, for any general, non-increasing degree distribution where $d(j)$ nodes have degree j, the degree distribution after edge removal can be classified based on the limit behavior of $d(j)/d(j-1)$. We study three main classes based on whether $\lim_{j\to\infty} d(j)/d(j-1)$ exists and is bounded away from both 0 and 1, approaches 1, or approaches 0. For $d(j)/d(j-1)$ bounded away from 0 and 1, the

distribution is exponential and after edge removal the new distribution is a different exponential; for $d(j)/d(j-1)$ approaching 0, the new distribution $d'(k)$ is $\Theta((1-p)^k d(k))$; and for $d(j)/d(j-1)$ approaching 1 quickly enough, the new distribution is $\Theta(d(j))$. This final result means that *for almost any distribution of interest, random edge-removal does not affect the distribution type, only the "scale" of the distribution.* Thus whether a network is power-law or has a similar distribution does not matter (which is useful due to the subtleties in distinguishing such distributions [80, 86]), and the relationship between the expected distribution in the new network and the original distribution can be easily determined.

The above discussion focused on the expected number of nodes of degree k in the graph after random edge removal, for an arbitrary but *single* value of k. How well is the final degree sequence concentrated around this target? In Section 3.3.4, we demonstrate a constant factor concentration, by a careful grouping of the vertices and through Martingale inequalities. One further way in which we improve upon [29] here is as follows: they require that the sum of the degrees-squared in the original graph be $O(n^{2-\Omega(1)})$ for their tail-bounds to hold. We require something much weaker than the analog of this for bond percolation. In Section 3.3.5 we discuss the effect of edge removal on probabilistic degree distributions: in models like those proposed by Chung and Lu [23] or Leskovec et al. [75] edges in a graph are already specified probabilistically. Thus nodes do not have degrees until after a graph is instantiated; instead they have expected degrees. We briefly define what it means for such a graph to have a power-law or exponential distribution, and show what effect edge-removals have on the distributions.

Thus, this work conducts a systematic study of what happens to a fundamental parameter of a graph, its degree sequence, under edge removal. The general flavor of the results obtained is that for essentially all of the interval of degrees of interest, the degree sequence retains its qualitative character, and several quantitative aspects as well.

3.3.1 Graphs with Power-Law Degree Sequences

In this section we derive specific bounds for how a power-law-like degree sequence changes through independently random edge removal. Letting $\deg(v)$ denote the degree of a node v, we define a graph with n vertices to be power-law-like with parameters $c_1, c_2, d_{\min}, d_{\max}$, and γ if $\forall i \in [d_{\min}, d_{\max}]$, $|\{v|\deg(v) = i\}| \in [\frac{c_1 n}{i^\gamma}, \frac{c_2 n}{i^\gamma}]$. In doing so, we develop the intuition and details which lead to the general categorization presented later.

Define $\deg(v)$ to be the degree of any vertex v. Given any graph G that has a power-law-like distribution on its node degrees we remove each edge independently with probability p. We show for all degrees k in the range $R = [d_{\min} \cdot (1-p), d_{\max} \cdot (1-p)]$, that there are an expected $\Theta(nk^{-\gamma})$ nodes of degree k in the new graph. The size of the new graph's power-law-like degree range is the original graph's power law range scaled down by a factor of $1-p$. Note that this places an implicit limit on how close p can be to 1 and still have a largely power-law new graph. If $(1-p)(d_{\max}-d_{\min})$ is small then the range of degrees for which the power-law-like property holds will be small as well. This makes intuitive sense because if p is small, only a few edges are

removed and the graph should not change much. If p is close to one, then most edges are removed, the graph changes drastically, power-law-like behavior only remains for a small range of nodes.

We define the random variable d_k to be the number of vertices with degree k after edge removal. This leads to the equation:

$$\sum_{j=\max(d_{\min},k)}^{d_{\max}} \binom{j}{k} p^{j-k}(1-p)^k \frac{c_1 n}{j^\gamma} \leq E[d_k] \leq \sum_{j=\max(d_{\min},k)}^{d_{\max}} \binom{j}{k} p^{j-k}(1-p)^k \frac{c_2 n}{j^\gamma}. \tag{3.1}$$

This equation comes from taking a sum over all degrees $j \geq k$ of the number of nodes with degree j times the probability that this node has degree k after the edge removal. If the distribution is exactly power-law ($c_1 = c_2 = c$, $d_{\min} = 1$, and $d_{\max} = \infty$) this equation reduces to $E[d_k] = \sum_{j=k}^{\infty} \binom{j}{k} p^{j-k}(1-p)^k cnj^{-\gamma}$, which is commonly used elsewhere [81, 29, 28]. If $c_1 \neq c_2$ then we can use c_1 throughout and introduce only a constant factor error term of $\frac{c_2}{c_1}$ which can be applied to the upper bound at the end, so for the remainder of our analysis we assume that $c_1 = c_2$.

Based on (3.1) we define for all $k \in R$ the functions g_k and f_k. Specifically we define $g_k(j)$ for all $j \geq k$ to be the "j term" in the summation for $E[d_k]$:

$$g_k(j) \quad = \tbinom{j}{k} p^{j-k}(1-p)^k c_1 n j^{-\gamma}. \tag{3.2}$$

We view $g_k(j)$ as the product of three separate functions. First is a step function which is 1 when $d_{\min} \leq j \leq d_{\max}$ and 0 otherwise. The second is a power-law function giving the number of nodes of degree j: $c_1 n j^{-\gamma}$. Finally we have the function that gives the probability of a node with degree j having final degree k. The type of degree distribution only affects the second of these functions; the other

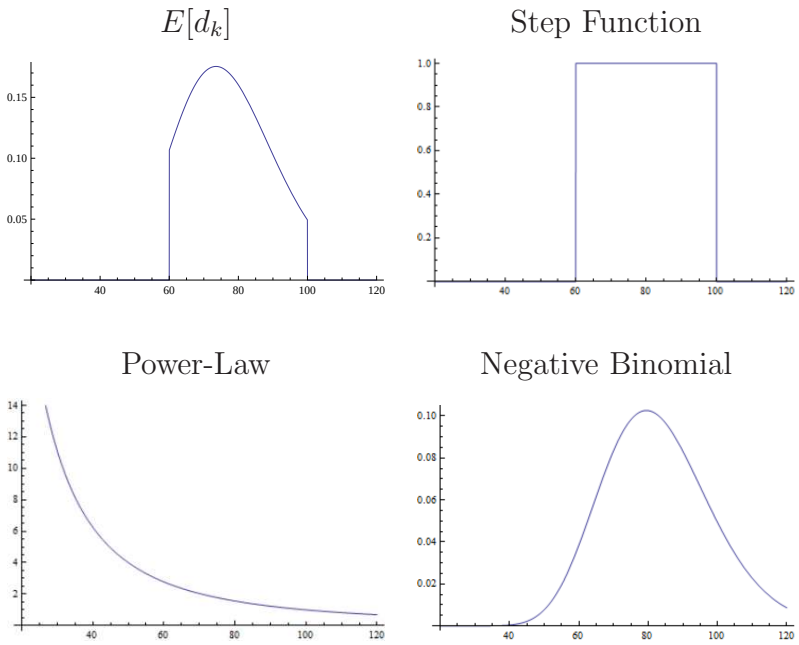


Figure 3.14: Examples of the three functions which make up g_k for $k = 20, d_{\min} = 60, d_{\max} = 100, c_1 n = 10000, \gamma = 2, p = .75$.

two are degree distribution independent. Examples of the three functions appear in Figure 3.14.

The intuition behind our method comes in two parts. First, for any given positive σ the third component function (which is almost a negative binomial probability distribution) has the vast majority of its mass (all but an amount exponential in $-\sigma$) within an $O(\sigma \cdot \frac{\sqrt{k}}{1-p})$ range around $j = \frac{k}{1-p}$. And second, that within this range the value of $j^{-\gamma}$ does not change much.

We also define the function $f_k(j)$ for $j \geq k$ to be the ratio of successive terms in the sum $E[d_k]$ such that

$$f_k(j) = g_k(j+1)/g_k(j) = \frac{pj^{\gamma}}{(j+1-k)(j+1)^{\gamma-1}}. \tag{3.3}$$

This leads to the following lemma:

Lemma 4 *$\forall k > \gamma$, the function $f_k(j)$ is strictly decreasing as j increases.*

The proof proceeds by treating f_k as a continuous function, and showing that the derivative is always negative.

Proof of Lemma 4 Taking the derivative of $f_k(j)$ with respect to j yields:

$$\begin{aligned}
&f_k'(j) \\
&= \frac{\gamma p j^{\gamma-1}}{(j+1-k)(j+1)^{\gamma-1}} - \frac{pj^{\gamma}\left((j+1)^{\gamma-1} + (j+1-k)(\gamma-1)(j+1)^{\gamma-2}\right)}{(j+1-k)^2(j+1)^{2\gamma-2}} \\
&= \frac{\gamma p j^{\gamma-1}(j+1-k)(j+1)^{\gamma-1} - pj^{\gamma}(j+1)^{\gamma-1} - pj^{\gamma}(j+1-k)(\gamma-1)(j+1)^{\gamma-2}}{(j+1-k)^2(j+1)^{2\gamma-2}} \\
&= \frac{pj^{\gamma-1}(j+1)^{\gamma-2}}{(j+1-k)^2(j+1)^{2\gamma-2}}\left(\gamma(j+1-k)(j+1) - j(j+1) - j(j+1-k)(\gamma-1)\right).
\end{aligned} \tag{3.4}$$

The fractional part of (3.4) is strictly positive. Therefore $f_k'(j)$ has the same sign as

$$\begin{aligned}
&\gamma(j+1-k)(j+1) - j(j+1) - j(j+1-k)(\gamma-1) \\
&= (j+1-k)(\gamma j + \gamma - \gamma j + j) - j(j+1) \qquad (3.5) \\
&= (j+1-k)\gamma - kj \\
&\leq j\gamma - jk \\
&< 0.
\end{aligned}$$

Since the derivative of $f_k(j)$ with respect to j is always negative when $k > \gamma$, $f_k(j)$ always decreases as j increases. ■

Using the fact that f_k is strictly decreasing we next find the maximum value of g_k for all $k > \gamma$. When $f_k(j) \geq 1$, g_k increases from j to $j+1$ and when $f_k(j) < 1$, g_k decreases from j to $j+1$. Therefore the maximum term $j_{\max}$ in (3.1) occurs within one of where $f_k(j)$ crosses from above 1 to below 1. We treat $f_k(j)$ as a continuous function and round to find the desired j:

$$1 = f_k(j) = \frac{pj}{j+1-k} \cdot \frac{j^{\gamma-1}}{(j+1)^{\gamma-1}} \quad \text{i.e., } j = \frac{k-1}{1-p\left(j/(j+1)\right)^{\gamma-1}}. \tag{3.6}$$

It immediately follows from (3.6) that $j_{\max}$ is always less than $\frac{k-1}{1-p}$ for any $\gamma > 1$, which is inherent in a power-law distribution. Furthermore, since $j \geq k$, $\frac{j}{j+1}$ is typically close to 1, thus $j_{\max}$ is typically close to $\frac{k-1}{1-p}$ as well. This observation drives our focus on $j \approx \frac{k}{1-p}$ from here on. Specifically we will use $g_k(\frac{k}{1-p})$ as a proxy for the more elusive $g_k(j_{\max})$.

We proceed by showing bounds on $g_k(\frac{k}{1-p})$, using the error factor $\xi = \cdot(1 - O(\frac{1}{pk}))$.

$$\begin{aligned} g_k(\frac{k}{1-p}) &= \frac{\left(\frac{k}{1-p}\right)!}{k!\left(\frac{pk}{1-p}\right)!} p^{pk/(1-p)}(1-p)^k c_1 n \left(\frac{k}{1-p}\right)^{-\gamma} \\ &= \frac{\sqrt{2\pi\frac{k}{1-p}} \cdot \left(\frac{k}{1-p}\right)^{\frac{k}{1-p}} e^{\lambda_{k/(1-p)} - \lambda_k - \lambda_{pk/(1-p)}}}{\sqrt{2\pi k} \cdot \sqrt{2\pi\frac{pk}{1-p}} \cdot k^k \left(\frac{pk}{1-p}\right)^{pk/(1-p)}} p^{\frac{pk}{1-p}}(1-p)^k c_1 n \left(\frac{k}{1-p}\right)^{-\gamma} \cdot \xi \end{aligned} \tag{3.7}$$

$$= \frac{(1-p)^\gamma}{\sqrt{2\pi p}} \cdot c_1 n \cdot k^{-\gamma-.5} \cdot \xi \tag{3.8}$$

where (3.7) relies on Stirling's approximation for factorials: $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \cdot e^{\lambda_n}$ where $\lambda_n \in [1/(12n+1), 1/(12n)]$.

Next we use $j = k/(1-p)$ as a starting point and bound how much g_k changes around this point. To that end we derive the bounds $f_k((k+\sigma\sqrt{k})/(1-p)) \leq 1 - ((1-p)\sigma)/(p\sqrt{k}+\sigma)$ and $g_k(\frac{k+\sigma\sqrt{k}}{1-p})/g_k(\frac{k}{1-p}) \leq e^{p\gamma/(1-p)-|\sigma|/8}$ for both positive and negative σ.

We have already shown that $g_k(\frac{k}{1-p}) \approx \left(\frac{(1-p)^\gamma}{\sqrt{2\pi p}}\right) c_1 n k^{-\gamma-.5}$ with very small error and that $\frac{k}{1-p}$ is close to $j_{\max}$. In this section we show bounds on how quickly g_k and f_k change as j moves away from $\frac{k}{1-p}$.

We start by showing bounds on $f_k(\frac{k}{1-p} + \frac{\sigma}{1-p}\sqrt{k})$ and $\frac{g_k(\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k})}{g_k(\frac{k}{1-p})}$ for any real (positive or negative) σ such that $\sigma > -p\sqrt{k}$. We need these bounds to prove that the overwhelming amount of the probability weight of nodes with final degree k comes from nodes with initial degrees in the range $\frac{k \pm O(\sqrt{k})}{1-p}$.

First we upper bound $f_k(\frac{k}{1-p} + \frac{\sigma}{1-p}\sqrt{k})$.

$$\begin{aligned} f_k(\frac{k}{1-p} + \frac{\sigma}{1-p}\sqrt{k}) &= \frac{p(\frac{k}{1-p} + \frac{\sigma}{1-p}\sqrt{k})}{\left(\frac{k}{1-p} + \frac{\sigma}{1-p}\sqrt{k} + 1 - k\right)} \frac{(\frac{k}{1-p} + \frac{\sigma}{1-p}\sqrt{k})^{\gamma-1}}{(\frac{k}{1-p} + \frac{\sigma}{1-p}\sqrt{k} + 1)^{\gamma-1}} \\ &\leq \frac{pk + p\sigma\sqrt{k}}{pk + \sigma\sqrt{k}} \\ &= 1 - \frac{(1-p)\sigma}{p\sqrt{k}+\sigma} \end{aligned} \tag{3.9}$$

Next we upper bound $\frac{g_k(\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k})}{g_k(\frac{k}{1-p})}$ when σ is positive:

$$\begin{aligned}
\frac{g_k(\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k})}{g_k(\frac{k}{1-p})} &= \frac{\binom{\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k}}{k}(1-p)^k p^{\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k}-k}(\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k})^{-\gamma}}{\binom{\frac{k}{1-p}}{k}(1-p)^k p^{\frac{k}{1-p}-k}\frac{k}{1-p}^{-\gamma}} \\
&= \left(\frac{(\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k})!}{(\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k}-k)!}\right) / \left(\frac{\frac{k}{1-p}!}{(\frac{k}{1-p}-k)!}\right) p^{\frac{\sigma}{1-p}\sqrt{k}} \left(\frac{k}{k+\sigma\sqrt{k}}\right)^{\gamma} \qquad (3.10) \\
&\leq \left(\frac{(\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k})!}{\frac{k}{1-p}!}\right)\left(\frac{(\frac{k}{1-p}-k)!}{(\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k}-k)!}\right)\cdot p^{\frac{\sigma}{1-p}\sqrt{k}} \\
&= \prod_{i=1}^{\frac{\sigma}{1-p}\sqrt{k}} \left(\frac{\frac{k}{1-p}+i}{\frac{k}{1-p}-k+i}p\right) \qquad (3.11) \\
&= \prod_{i=1}^{\frac{\sigma}{1-p}\sqrt{k}} \left(\frac{k+(1-p)i}{kp+(1-p)i}p\right) \\
&\leq \prod_{i=\frac{\sigma}{2-2p}\sqrt{k}+1}^{\frac{\sigma}{1-p}\sqrt{k}} \left(\frac{k+\frac{\sigma}{2}\sqrt{k}}{kp+\frac{\sigma}{2}\sqrt{k}}p\right) \\
&\leq \left(\frac{k+\frac{\sigma}{2}\sqrt{k}}{k+\frac{\sigma}{2p}\sqrt{k}}\right)^{\frac{\sigma}{2-2p}\sqrt{k}} \\
&\leq \left(1-\frac{(1-p)\sigma}{2p\sqrt{k}+\sigma}\right)^{\frac{\sigma}{2-2p}\sqrt{k}} \\
&\leq e^{\frac{-\sigma^2}{4p+2\sigma/\sqrt{k}}} \leq e^{\max(\frac{-\sigma^2}{8p},\frac{-\sigma\sqrt{k}}{4})} \leq e^{-\sigma/8}. \qquad (3.12)
\end{aligned}$$

The bound on $\frac{g_k(\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k})}{g_k(\frac{k}{1-p})}$ when σ is negative is a little more involved. First, the $\left(\frac{k}{k+\sigma\sqrt{k}}\right)^{\gamma}$ term in (3.10) is greater than 1 when σ is negative, so it cannot simply be discounted. However it is at most $\left(\frac{k}{k-p\sqrt{k}\sqrt{k}}\right)^{\gamma} = \left(1+\frac{p}{1-p}\right)^{\gamma}$ which is less than $e^{p\gamma/(1-p)}$, which only depends on the fixed parameters p and γ.

The next change from positive to negative σ comes on (3.11) which is replaced by:

$$\begin{aligned}
\frac{g_k(\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k})}{g_k(\frac{k}{1-p})} &\leq e^{p\gamma/(1-p)} \prod_{i=1}^{\frac{-\sigma}{1-p}\sqrt{k}} \left(\frac{\frac{k}{1-p}-k-i}{\frac{k}{1-p}-i}\frac{1}{p}\right) \\
&\leq e^{p\gamma/(1-p)} \prod_{i=\frac{-\sigma}{2-2p}\sqrt{k}+1}^{\frac{-\sigma}{1-p}\sqrt{k}} \left(\frac{pk-i(1-p)}{pk-pi(1-p)}\right) \\
&\leq e^{p\gamma/(1-p)} \left(1-\frac{\frac{-\sigma}{2}\sqrt{k}(1-p)}{pk-p\frac{-\sigma}{2}\sqrt{k}}\right)^{\frac{-\sigma}{2-2p}\sqrt{k}} \\
&\leq e^{p\gamma/(1-p)-\frac{\frac{-\sigma}{2}\sqrt{k}(1-p)}{pk-p\frac{-\sigma}{2}\sqrt{k}}\cdot\frac{-\sigma}{2(1-p)}\sqrt{k}} \\
&\leq e^{p\gamma/(1-p)-\frac{\sigma^2}{4p(1+\sigma/(2\sqrt{k}))}} \\
&\leq e^{p\gamma/(1-p)-\frac{\sigma^2}{4p}} \text{ if } \sigma > -2\sqrt{k}.
\end{aligned}$$

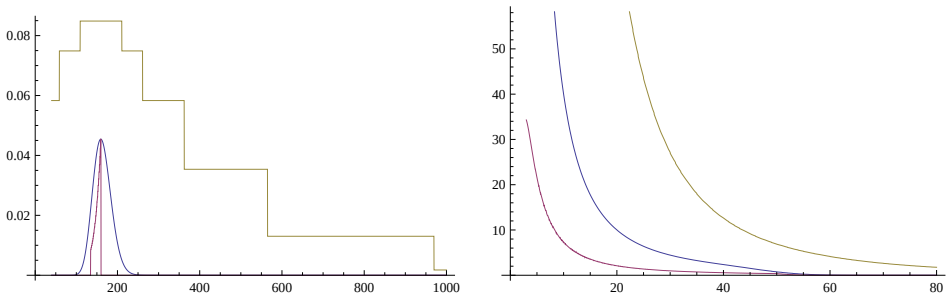
While our analysis runs into a discontinuity at $\sigma = -2\sqrt{k}$, this is unimportant. When $\sigma < -p\sqrt{k}$ then $\frac{k}{1-p}+\frac{\sigma}{1-p}\cdot\sqrt{k} < k$. Since there are no nodes with initial degree $< k$ that end up with degree k, in these cases it follows that $g_k(\frac{k}{1-p}+\frac{\sigma}{1-p}\cdot\sqrt{k}) = 0$.

Combining all of these bounds gives that for any meaningful σ,

$$\frac{g_k(\frac{k}{1-p}+\frac{\sigma}{1-p}\sqrt{k})}{g_k(\frac{k}{1-p})} \leq e^{p\gamma/(1-p)-|\sigma|/8}. \tag{3.13}$$

Since we have a dropoff in $g_k(j)$ from $g_k(\frac{k}{1-p})$ that is exponential in $(j-\frac{k}{1-p})/\sqrt{k}$, almost all (at least $1-O(\frac{1}{n})$) of the concentration of $E[d_k]$ comes from a $\sqrt{k}\ln n$ region around $g_k(\frac{k}{1-p})$.

From here we use the bounds on $f_k((k+\sigma\sqrt{k})/(1-p))$ to create a geometric series whose sum is a lower bound for the $O(\sqrt{k})$ terms of g_k leading up to $g_k(k/(1-p))$, which is itself a lower bound on $E[d_k]$. We go on to use the bounds



(a) (b)

Figure 3.15: Figure (a) shows the function g_{40} with $p = .75$, $\gamma = 2$ and $c_1 = 1000$ in blue along with our upper and lower bounds for it. Figure (b) shows $E[d_k]$ for various values of k along with our upper and lower bounds. With these parameters, the lower bound is approximately $E[d_k]/5$ and the upper bound approximately $14E[d_k]$.

on $g_k(\frac{k+\sigma\sqrt{k}}{1-p})/g_k(\frac{k}{1-p})$ directly to upper bound $E[d_k]$. Figure 3.15 gives a graphical illustration of our upper and lower bounds. This leads to the following theorem:

Theorem 5 *Suppose we are given a fixed edge removal probability p and a power-law-like graph G given by constants γ, c_1, c_2 and range $d_{\min}, d_{\max}$. The degree distribution of the induced graph where each edge in G is removed with probability p is power-law-like as well. Specifically for all k in $[\frac{d_{\min}}{1-p}, \frac{d_{\max}}{1-p}]$ the term $E[d_k]$ falls between*

$$\left(\left(1 - e^{-\frac{1}{p}}\right)(p - k^{-.5})(1-p)^{\gamma-1}/\sqrt{2\pi p}\right) \cdot \left(c_1 n k^{-\gamma}/\gamma^{\gamma}\right) \cdot (1 - O(1/(12pk)))$$

(where the "γ^{γ}" term is only necessary when $k < \gamma$) and

$$\left(20e^{(p\gamma/(8(1-p)))}(1-p)^{\gamma-1}/\sqrt{2\pi p}\right) \cdot \left(c_2 n k^{-\gamma}\right).$$

Proof of Theorem 5

Our general technique to lower bound $E[d_k]$ is to lower bound $g_k(j)$ for the $\sqrt{k}$ values of j preceding $\frac{k}{1-p}$. We do this by taking $g_k(\frac{k}{1-p})$ and successively dividing it by the intermediate $\frac{k}{1-p} - j$ values of f_k. From Eq (3.9) we know that these f_k values are not too large. The sum of these lower bounds, which we show to be $\Theta(c_1 n k^{-\gamma})$, is itself a lower bound on $E[d_k]$. Figure 3.15 shows our upper and lower bounds for both g_k and $E[d_k]$.

For any k such that $k \geq \left(\frac{1}{p}\right)^2$ and $d_{\max} \geq \frac{k}{1-p}$:

$$\begin{aligned}
E[d_k] = \sum_{i=k}^{d_{\max}} g_k(i) &\geq \sum_{i=\frac{k-\sqrt{k}}{1-p}}^{\frac{k}{1-p}} g_k(i) \\
&= \sum_{i=\frac{k-\sqrt{k}}{1-p}}^{\frac{k}{1-p}} \left(g_k(\frac{k}{1-p}) \prod_{j=0}^{\frac{k}{1-p}-i-1} \frac{1}{f_k(i+j)} \right) \\
&\geq \sum_{i=\frac{k-\sqrt{k}}{1-p}}^{\frac{k}{1-p}} \left(g_k(\frac{k}{1-p}) \prod_{j=0}^{\frac{k}{1-p}-i-1} \frac{1}{f_k(\frac{k-\sqrt{k}}{1-p})} \right) \quad (f_k \text{ is decreasing}) \\
&\geq \sum_{i=\frac{k-\sqrt{k}}{1-p}}^{\frac{k}{1-p}} \left(g_k(\frac{k}{1-p}) \prod_{j=0}^{\frac{k}{1-p}-i-1} \frac{1}{1+\frac{1-p}{p\sqrt{k}-1}} \right) \quad (\text{from (3.9)}) \\
&\geq \sum_{i=\frac{k-\sqrt{k}}{1-p}}^{\frac{k}{1-p}} \left(g_k(\frac{k}{1-p}) \prod_{j=0}^{\frac{k}{1-p}-i-1} (1-\frac{1-p}{p\sqrt{k}-1}) \right) \\
&\geq g_k(\frac{k}{1-p}) \sum_{i=0}^{\frac{\sqrt{k}}{1-p}-1} \left(1-\frac{1-p}{p\sqrt{k}-1}\right)^i \\
&= g_k(\frac{k}{1-p}) \frac{1-\left(1-\frac{1-p}{p\sqrt{k}-1}\right)^{\frac{\sqrt{k}}{1-p}}}{\frac{1-p}{p\sqrt{k}-1}} \\
&\geq \frac{(1-p)^\gamma}{\sqrt{2\pi p}} c_1 n k^{-\gamma-.5} \frac{\left(1-e^{-\frac{1}{p}}\right)(p\sqrt{k}-1)}{1-p} \\
&= \left(\frac{\left(1-e^{-\frac{1}{p}}\right)(p-k^{-.5})(1-p)^{\gamma-1}}{\sqrt{2\pi p}} \right) c_1 n k^{-\gamma}. \qquad (3.14)
\end{aligned}$$

Note that we could equivalently bound $E[d_k] \geq \sum_{j=\frac{k}{1-p}}^{\frac{k+\sqrt{k}}{1-p}} g_k(j) \geq \Theta(\frac{pk^{-\gamma}}{1-p})$ using nearly identical steps. As long as $d_{\max} - d_{\min} > 2\sqrt{k}$ there are always enough j values either above or below $\frac{k}{1-p}$ to contribute enough probability weight to this lower bound that $E[d_k] \geq \Theta(\frac{pk^{-\gamma}}{1-p})$.

Whenever $k < \gamma$ we lose the guarantee that $f_k(j)$ is monotonically decreasing with j. However, for every term $g_k(j)$:

$$\begin{aligned} g_k(j) &= \binom{j}{k} p^{j-k}(1-p)^k c_1 n j^{-\gamma} \\ &\geq \binom{j+\gamma}{k+\gamma} p^{j+\gamma-k-\gamma}(1-p)^{k+\gamma} c_1 n (j+\gamma)^{-\gamma} \\ &\geq g_{k+\gamma}(j+\gamma). \end{aligned}$$

Therefore $E[d_k] \geq E[d_{k+\gamma}]$. So if $E[d_{k+\gamma}] \geq \alpha n(k+\gamma)^{-\gamma}$, then $E[d_k]$ also has a power-law expected degree sequence, since $E[d_k] \geq E[d_{k+\gamma}] \geq \alpha n(k+\gamma)^{-\gamma} \geq \frac{\alpha}{\gamma^\gamma} n k^{-\gamma}$.

For the upper bound on $E[d_k]$ we use the fact from (3.13) that $\forall \sigma$, $g_k(\frac{k \pm \sigma\sqrt{k}}{1-p}) \leq g_k(\frac{k}{1-p}) \cdot e^{p\gamma/(1-p) - \frac{|\sigma|}{8}}$. We group j values based upon which value of σ gives $\frac{k+(2^\sigma-1)\sqrt{k}}{1-p}$ closest to j in the direction towards $\frac{k}{1-p}$. This grouping is evident in the step-like behavior of the upper bound on g_k in Figure 3.15. We then bound a sum over all of these groups of an upper bound on the total weight from each group.

$$\sum_{i=k}^{d_{\max}} g_k(i) \leq \sum_{i=-\infty}^{\infty} g_k(i)$$

$$\leq \sum_{\sigma=0}^{\infty} \sum_{i=0}^{\sqrt{k}(2^\sigma)/(1-p)-1} \left(g_k(\frac{k}{1-p} - \sqrt{k} \cdot \frac{2^\sigma - 1}{1-p} - i) + g_k(\frac{k}{1-p} + \sqrt{k} \cdot \frac{2^\sigma - 1}{1-p} + i) \right) \tag{3.15}$$

$$\leq \sum_{\sigma=0}^{\infty} \sum_{i=0}^{\sqrt{k}(2^\sigma)/(1-p)-1} \left(g_k(\frac{k}{1-p} - \sqrt{k} \cdot \frac{2^\sigma - 1}{1-p}) + g_k(\frac{k}{1-p} + \sqrt{k} \cdot \frac{2^\sigma - 1}{1-p}) \right)$$

$$\leq \sum_{\sigma=0}^{\infty} \left(\sqrt{k} \cdot \frac{2^\sigma}{1-p} \right) \left(g_k(\frac{k}{1-p} - \sqrt{k} \cdot \frac{2^\sigma - 1}{1-p}) + g_k(\frac{k}{1-p} + \sqrt{k} \cdot \frac{2^\sigma - 1}{1-p}) \right)$$

$$\leq \frac{\sqrt{k}}{1-p} \cdot \sum_{\sigma=0}^{\infty} 2^\sigma \left(2g_k(\frac{k}{1-p}) e^{\frac{p\gamma/(1-p) - (2^\sigma - 1)}{8}} \right) \text{ from (3.13)}$$

$$\leq \frac{2g_k(\frac{k}{1-p})\sqrt{k} \cdot e^{p\gamma/(1-p)+1/8}}{1-p} \sum_{\sigma=0}^{\infty} (2^\sigma) \left(e^{\frac{-2^\sigma}{8}} \right)$$

$$= \left(\frac{2e^{p\gamma/(1-p)+1/8}(1-p)^{\gamma-1}}{\sqrt{2\pi p}} \right) c_1 n k^{-\gamma} \sum_{\sigma=0}^{\infty} (2^\sigma) \left(e^{\frac{-2^\sigma}{8}} \right) \cdot (1 - O(1/(12pk))) \tag{3.16}$$

$$= \left(\frac{2e^{p\gamma/(1-p)+1/8}(1-p)^{\gamma-1}}{\sqrt{2\pi p}} \right) c_1 n k^{-\gamma} \sum_{\sigma=0}^{\infty} \left(e^{\sigma \ln 2 - \frac{2^\sigma}{8}} \right)$$

$$= \left(\frac{20e^{p\gamma/(1-p)+1/8}(1-p)^{\gamma-1}}{\sqrt{2\pi p}} \right) c_1 n k^{-\gamma}, \tag{3.17}$$

where the slack from (3.16) comes from Stirling's approximation and (3.17) follows because the preceding summation is less than 10.

Putting the two bounds from (3.14) and (3.17) together completes the proof.

■

3.3.2 Exponential Degree Sequences

We follow up with a similar, though much more straightforward result for graphs with exponential degree sequences. We say a graph has a γ exponential degree sequence if there exist constants c_1, c_2 such that there are between $c_1 n\gamma^k$ and $c_2 n\gamma^k$ nodes of degree k (for the analysis we use a single constant c_1). If each edge is removed with probability p, then the expected number of nodes with final degree k is $E[d_k] = \sum_{j\geq k} c_1 n\gamma^j \binom{j}{k} p^{j-k}(1-p)^k$. In the remainder of this section we prove the following theorem:

Theorem 6 *Given a graph with a γ exponential degree sequence and an edge removal probability p such that $0 \leq p\gamma < 1$, the expected degree sequence for the resulting graph is an $\frac{\gamma(1-p)}{1-p\gamma}$–exponential.*

As with the power-law case, the terms in the summation can be broken into three components: a step function, an exponential function, and a negative binomial. However unlike the power-law case, the exponential function is significant enough that the negative binomial does not dominate it, and the algebra becomes much simpler. We can rewrite $E[d_k]$ as

$$\sum_{j\geq k} c_1 n\gamma^j \binom{j}{k} p^{j-k}(1-p)^k = c_1 n \left(\frac{\gamma(1-p)}{1-p\gamma}\right)^k \sum_{j\geq k} \binom{j}{k} (p\gamma)^{j-k}(1-p\gamma)^k$$
$$= \frac{c_1 n}{1-p\gamma}\left(\frac{\gamma(1-p)}{1-p\gamma}\right)^k.$$

This shows directly that $E[d_k]$ is exponential in k. Specifically the new graph has a $\frac{\gamma(1-p)}{1-p\gamma}$ exponential degree sequence. Note that this result does not require that $\gamma < 1$, but only requires the weaker condition that $p\gamma < 1$. This restriction is in

place because if $p\gamma \geq 1$ then the new negative binomial term is no longer a valid probability distribution.

3.3.3 General Principles

So far we have given two specific examples of what happens to a graph's degree distribution when edges are removed uniformly and independently at random. In this section we expand upon those results by developing general asymptotic principles that apply to almost any monotonically non-increasing degree distribution. We look at any degree distribution given in terms of two parts, a unit step function $s(j)$ which is one for all j between $d_{\min}$ and $d_{\max}$ and a density function $d(j)$ giving the number of nodes of degree j if $s(j) = 1$. These correspond to the first and second components of the terms of g_k, where the third component remains the negative binomial function. For now we assume $d_{\min} = 1$ and $d_{\max} = \infty$. For review, $E[d_k] = c_1 n \sum_{j=k}^{\infty} \binom{j}{k} \left(\frac{1-p}{p}\right)^k p^j s(j) d(j)$.

We classify most such distributions into three classes based upon the asymptotic behavior of $\frac{d(j)}{d(j-1)}$. The first class includes all exponential degree distributions. When $d(j)$ is exponential ($d(j) = \Theta(\gamma^j)$ for some γ) and $\lim_{j\to\infty} \frac{d(j)}{d(j-1)} = \gamma$ for some $0 < \gamma < 1$, we have already shown that the result is an expected distribution that is exponential with $E[d_k] = \frac{c_1 n}{1-p\gamma} \cdot \left(\frac{\gamma(1-p)}{1-p\gamma}\right)^k$. The exponential case is unique in that the p^j term and the $d(j)$ term align to produce a new negative binomial distribution, plus a term exponential in k. The new negative binomial sums to a constant, leaving behind only the new exponential term.

The second class of functions we consider are super-exponential, decreasing functions where $\frac{d(j)}{d(j-1)} = o(\frac{1}{pj})$. Here we have functions that decrease significantly faster than any exponential. If a function $d(j)$ decreases fast enough that asymptotically $\frac{d(j)}{d(j-1)} \leq \frac{\epsilon}{pj}$ for some $\epsilon < 1$, then for sufficiently large k,

$$(1-p)^k d(k) \leq E[d_k] = \sum_{j=k}^{\infty} p^{j-k}(1-p)^k d(j)$$
$$\leq \sum_{j=k}^{\infty} (1-p)^k d(k) \left(\frac{\epsilon}{j}\right)^{j-k} = O((1-p)^k d(k)).$$

The degree distribution decreases so rapidly in this case, that the number of nodes with final degree k is dominated by the number of nodes with initial degree k. Thus the distribution stays asymptotically the same.

Finally we consider the case where $\frac{d(j)}{d(j-1)} = 1 \pm o(\frac{1-p}{\sqrt{j}}))$, which includes all polynomial degree distributions, including power law distributions. In these cases, the maximum term in the $E[d_k]$ summation occurs at the highest j such that $\frac{d(j)}{d(j-1)} \cdot p \cdot \frac{j}{j-k} \geq 1$. Since $\frac{d(j)}{d(j-1)} \approx 1$, this occurs near $j = \frac{k}{1-p}$, which maximizes the negative binomial component function. Recall that the vast majority of the weight in a negative binomial occurs within an $O(\frac{\sqrt{k}}{1-p})$ region around its maximum. For some $x, y \in [\frac{k-\sigma\sqrt{k}}{1-p}, \frac{k+\sigma\sqrt{k}}{1-p}]$, $d(j)$ can change by at most a ratio of

$$\frac{d(x)}{d(y)} \leq \prod_{j=(k-\sigma\sqrt{k})/(1-p)+1}^{(k+\sigma\sqrt{k})/(1-p)} \max\left(\frac{d(j)}{d(j-1)}, \frac{d(j-1)}{d(j)}\right),$$

i.e., at most $\left(1 + o((1-p)/(\sqrt{k}))\right)^{2\sigma\sqrt{k}/(1-p)}$ which for sufficiently large k is at most

$$\left(1 + (1-p)/(2\sigma\sqrt{k})\right)^{2\sigma\sqrt{k}/(1-p)} \leq e^{(1-p)/(2\sigma\sqrt{k})\cdot 2\sigma\sqrt{k}/(1-p)} = e.$$

Since for sufficiently large k, the value of $d(j)$ changes by at most a factor of e within the range for which the negative binomial component is large, we can approximate d as a constant $d(k/(1-p))$ with at most a factor of error e. Thus for this category of $d(j)$ functions, $E[d_k] = cn\Theta(d(\frac{k}{1-p}))$. This result holds for all power-law distributions, as well as any smooth, piece-wise combination of power-law functions, or any other slowly changing function.

In each of these three cases, the effect of the $s(j)$ step function on the summations is that of a range selector. Any k such that $\frac{k}{1-p}$ falls well within the $s(j) = 1$ region will behave as described above. Any k with $\frac{k}{1-p}$ is sufficiently far away from where $s(j) = 1$ will have very small $E[d_k]$, with $E[d_k]$ transitioning slowly between the two extremes.

3.3.4 Bounds on Large Deviations from Expected Degrees

In the previous section, we categorize how the expected degree sequence of a graph changes when its edges are removed at random. In this section we take that one step further and show that with probability at least $1 - \frac{1}{n}$, all relevant degrees k will simultaneously have $\Theta(E[d_k])$ nodes of degree k in the resulting graph as long as a few assumptions are met. The challenge in developing tail bounds comes from the observation that every edge's final degree is correlated with the final degrees of its neighbors. This dependency not only means that Chernoff bounds do not apply, but techniques requiring low degree dependency graphs are not sufficient either. Cooper and Lu [29] use the Azuma-Hoeffding inequality [6, 57] to show for vertex removal

that when $\sum_{v \in V} \deg(v)^2 = O(n^{2-\epsilon})$ for some positive ϵ, the probability that the actual number of nodes with degree k after edge removal deviates from $E[d_k]$ by more than a constant factor is something at most $O(n^{-2})$ (they give a more specific bound). We expand upon their technique, apply it to edge and not vertex removal, and get improved results in several ways. In addition to requiring a bound on the sum of the degrees squared, they reach their result by requiring the power law range to include all degrees (thus not allowing graphs with $d_{\min} \neq 0$ or $d_{\max} \neq \infty$) and the number of nodes of degree k to grow linearly with n. We remove these requirements and replace them with more generally applicable requirements. In this section we will prove the following large deviation theorem:

Theorem 7 *Suppose we are given an initial graph such that for some constants c_1, c_2, and with probability at least $1 - O(n^{-4})$ for all relevant k, the nodes with final degree k come from a set of size at most $E[d_k]c_1\sqrt{k \ln n}$ and with maximum degree $c_2 k(1 + \ln n/\sqrt{k})$. If each edge is removed independently with probability p, then for all such k simultaneously, the final number d_k of nodes of degree k will be within $[\frac{E[d_k]}{2}, \frac{3E[d_k]}{2}]$ with probability at least $1 - O(\frac{1}{n})$. For this theorem, we define all relevant k to mean those $k \in [d_{min}(1-p), d_{\max}(1-p)]$ such that $E[d_k] \geq 72 c_1 c_2 k \ln^2 n(\sqrt{k} + \ln n)$ or both $E[d_k] = \Omega(\ln n)$ and $k = \Omega(\ln^4 n)$.*

Remark. If we set $c_1 = 4/(1-p)$ and $c_2 = 1/(1-p)$, any power law graph satisfies Theorem 7's requirements. This includes power-law graphs that the Cooper-Lu method does not apply to because of the edge density. To demonstrate how our theorem applies to a concrete example, consider a power-law graph where nodes

vary in degree from 1 to $n/10$ with $\gamma = 2+\epsilon$. The expression $\sum_{i=1}^{n/10} \alpha i^{-2+\epsilon}$ for some α gives the number of nodes in the graph. Solving for α gives $\alpha \approx n/1.6$, and thus for k small enough, the expected number of nodes with final degree k is $\Theta(n/k^{2+\epsilon})$. Applying Theorem 7 shows us that the actual number of nodes of degree k will be close to $E[d_k]$ roughly as long as either: (a) $E[d_k] \geq 72c_1c_2k\ln^2 n(\sqrt{k}+\ln n)$ which leads to $k = O(n^{1/3}/\ln n)$; or (b) $k = \Omega(\ln^4 n)$, which for reasonably large n is satisfied if $E[d_k] = \Omega(\ln n)$ and (a)'s conditions are not satisfied. ($\Omega(\ln n) = E[d_k]$ is equivalent to $\ln n = O(n/k^{2+\epsilon})$ and thus $k = O((n/\ln n)^{1/(2+\epsilon)})$.) These two cases combine to show that our theorem covers most of the range of degrees k likely to occur in the random graph. The only degrees likely to occur for which our tail bounds do not apply are those where $E[d_k] < O(\ln n)$. In these cases even if we could directly apply Chernoff bounds to the large deviation probabilities, there are simply not enough expected nodes of degree k to drive the probability of a bad event below $1/n$. Thus no scheme can be expected to improve significantly upon our result in this particular example.

For the first step in our proof we take any arbitrary ordering on the edges and consider the random process where the edges are exposed sequentially, each being removed with probability p. For any vertex v, we can use Chernoff bounds to show that at every step in the random process, if v has had x of its edges exposed, then less than $xp-4\sqrt{x\ln n}$ or more than $xp+4\sqrt{x\ln n}$ of those edges will have been removed with probability at most $2\exp(-xp(\frac{4\sqrt{x\ln n}}{xp})^2/3) = 2\exp(-(\frac{16\ln n}{3p})) = O(n^{-5})$. If we then take the union bound over all vertices and all time steps we have that with probability at least $1-O(n^{-3})$ for every vertex v and number of exposed edges x,

between $xp - 4\sqrt{x \ln n}$ and $xp + 4\sqrt{x \ln n}$ of those edges will have been removed. For the remainder of this section we condition upon this event, which occurs almost certainly, and we concede the remaining $O(n^{-3})$ fraction of instances as having large deviation. This means that for any k, only those vertices for which the initial degree $\deg(v)$ satisfies

$$\deg(v)(1-p) - 4\sqrt{\deg(v) \ln n} \leq k \leq \deg(v)(1-p) + 4\sqrt{\deg(v) \ln n}$$

can have final degree k. Note that this result immediately implies one of the conditions for Theorem 7, namely that with high probability only nodes of initial degree at most $c_2 k(1 + \ln n/\sqrt{k})$ can end up with final degree k.

To prove our theorem, we also condition upon the high probability event that for all relevant k, only nodes from the set (which is a precondition of Theorem 7) of size at most $E[d_k]c_1\sqrt{k \ln n}$ might end up with final degree k. By the union bound over all k individually, with probability at least $1 - O(n^{-3})$ this will happen for all k simultaneously. We condition on this almost certain event throughout the rest of the argument.

For any such k, when $E[d_k] \geq 72c_1c_2k \ln^2 n(\sqrt{k} + \ln n)$, only $E[d_k]c_1\sqrt{k \ln n}$ nodes are conditioned to possibly have final degree k, and each of these has at most $c_2 k(1 + \ln n/\sqrt{k})$ edges to other nodes in this set, for a total of at most $E[d_k]c_1c_2k^{1.5}\sqrt{\ln n}(1 + \ln n/\sqrt{k})/2$ edges internal to this set. Furthermore, each edge we expose in our random process can affect at most two of these nodes (its endpoints). Define the random variable X to be the expected number of nodes with final degree k after exposing these internal edges. We can use Azuma's inequality

which says

$$Pr[|X - E[d_k]| \geq E[d_k]/3] \leq 2\exp(-E[d_k]^2/(18 \sum_{\text{internal edges}} 2^2))$$

$$\leq 2\exp(-E[d_k]^2/(36E[d_k]c_1c_2k^{1.5}\sqrt{\ln n}(1+\ln n/\sqrt{k})))$$

$$\leq 2\exp(-2\ln n) = 2n^{-2}.$$

Therefore, with high probability $X \in [\frac{2}{3} \cdot E[d_k], \frac{4}{3} \cdot E[d_k]]$. After exposing all of the internal edges, those nodes with external edges will not yet have their degrees fixed because their external edges have not been exposed. By definition, each external edge affects at most one relevant node. Therefore these remaining nodes end up with degree k independently from each other, and thus we can apply Chernoff bounds directly to exposing the external edges. Because $X \gg \ln n$, with high probability the final result will be within a small constant factor of X. Specifically, if the final number of nodes of degree k is Y:

$$Pr[|Y - E[d_k]| > E[d_k]/2] \leq 2n^{-2} + Pr[|Y - X| \geq E[d_k]/6 | X > 2E[d_k]/3]$$

$$\leq 2n^{-2} + \exp(-2E[d_k]/3(1/4)^2/3)$$

$$\leq 2n^{-2} + \exp(-c_1c_2k^{1.5}\ln^2 n) = (2+o(1))n^{-2}.$$

For the second case of Theorem 7, when $E[d_k] = \Omega(\ln n)$ and $k = \Omega(\ln^4 n)$, once again for any k we have $E[d_k]c_1\sqrt{k\ln n}$ nodes which might end up with degree k. Each of these nodes has at most a $\binom{k/(1-p)}{k}(1-p)^k p^{k/(1-p)-k} = O(1/\sqrt{k})$ probability of having final degree k. We divide this set of nodes into groups of size at most $k/(2-2p)$ (and at most $k/(4-4p)$ unless there is only one group). We make this

partition fairly uniformly, so that within each group G, the total expected number of nodes of degree k within the group, denoted $E[d_k^G]$ is within a constant factor, say 2, of every other group. The balls-and-bins model shows that even if we assign the nodes to groups randomly, this will be satisfied. Thus each group will have an $E[d_k^G]$ such that

$$E[d_k^G] \geq E[d_k] \cdot \frac{k/(4-4p)}{2E[d_k]c_1\sqrt{k}\ln n} \geq \frac{\sqrt{k}}{2c_1(4-4p)\ln n} \geq \Omega(\ln n).$$

Now consider the process where we arbitrarily order the internal edges, exposing them one at a time, and once finished we expose all of the external edges at once. How much can exposing one of the internal edges change $E[d_k^G]$? For any node v at any point in this process the internal edges already exposed are at most half of its total edges. Thus for each internal edge and each of its two endpoints, for some i: $\deg(v)/2 \leq i \leq \deg(v)$ and $i \leq j$, where i is the number of edges left unexposed and j the number left to keep, the effect on the expectation of exposing a single edge one way or the other per endpoint is at most

$$\begin{aligned}
&\left|\binom{i-1}{j-1}p^{i-j}(1-p)^{j-1} - \binom{i-1}{j}p^{i-j-1}(1-p)^{j}\right| \\
&= \left|\binom{i}{j}p^{i-j}(1-p)^{j}\left(\frac{j}{i(1-p)} - \frac{i-j}{ip}\right)\right| \\
&= O(\frac{1}{\sqrt{i}})\left|\frac{j-i(1-p)}{i(1-p)p}\right|.
\end{aligned}$$

Because of the conditioning that at any point the number of edges removed r is between $xp - 4\sqrt{x\ln n}$ and $xp + 4\sqrt{x\ln n}$, $(j - i(1-p))$ is at most $8\sqrt{\deg(v)\ln n}$.

This follows because $i = \deg(v) - x$ and $j = k - (x - r)$, and thus

$$\begin{aligned} |j - i(1-p)| &= |k - x + r - (\deg(v) - x)(1-p)| \\ &= |(k - \deg(v)(1-p))| + |(r - xp)| \\ &\leq 4\sqrt{\deg(v)\ln n} + 4\sqrt{x \ln n} \\ &\leq 8\sqrt{\deg(v)\ln n}. \end{aligned}$$

Therefore

$$\left| \binom{i-1}{j-1} p^{i-j}(1-p)^{j-1} - \binom{i-1}{j} p^{i-j-1}(1-p)^{j} \right| = O(\frac{\sqrt{\ln n}}{i}) = O(\frac{\sqrt{\ln n}}{k}). \quad (3.18)$$

Define X^G and Y^G to be the expected number of nodes in G of degree k after exposing the internal and all edges respectively. If we look at any single such group G in isolation and using the bound from (3.18) in applying Azuma's inequality, we get

$$\begin{aligned} Pr[|X^G - E[d_k^G]| \geq E[d_k]/2] &\leq 2\exp(-E[d_k^G]^2/(8 \sum_{\text{internal edges}} O(\sqrt{\ln n}/k)^2)) \\ &\leq 2\exp(-E[d_k^G]^2/O(\ln n)) = n^{-\Omega(1)}. \end{aligned}$$

After exposing the internal edges, we can expose all of the external edges at random, and apply Chernoff bounds to show that the final Y^G values will be very close to X^G. The analysis details are the same as in the first case of the theorem. For n large enough, $n^{-\Omega(1)}$ plus the large deviation bound from the external edges is a small enough per group, that with probability at most n^{-2} *any* group will have large deviation. As long as no group's Y^G has a large deviation from $E[d_k^G]$, the

sum Y cannot have large deviation from $E[d_k]$ either. Since for each k the large deviation probability of Y is at most $O(1/n^2)$, by taking a union bound over all such k from 0 to n we see that the probability that *any* such k has a large deviation is at most $O(1/n)$. This completes the proof of Theorem 7.

3.3.5 Edge Removal on Probabilistic Degree Distributions

For completeness we also mention the effect of random edge removal on stochastic graph models. Two such models are the Chung-Lu [23] and Kronecker [75] models. In the Chung-Lu model, every node v in a bipartite graph is parametrized with an expected degree x_v. The model includes each edge u, v with probability proportional to $x_u \cdot x_v$. The Kronecker graph model starts with a small initiator adjacency matrix with entries between 0 and 1. A Kronecker product of two matrices A and B each of size n by n gives a new n^2 by n^2 matrix where entry i, j is $A_{(\lceil i/n \rceil, \lceil j/n \rceil)} \cdot B_{(i \mod n, j \mod n)}$. A Kronecker power of the matrix gives a probabilistic adjacency matrix from which graphs can be instantiated. In either case, each edge has a probability with which it is independently placed in an instance of the graph. We can sum the probabilities of all the edges incident on a node to get that node's expected degree.

In these, or any other, probabilistic graph models, we cannot used the definition of a degree distribution $d(j)$ for integers j giving the number of nodes of degree j. We modify the definition slightly, to say a stochastic graph specification has distribution $d(j)$ if for all j the number of nodes with expected degrees in the

range $[j, j+1)$ is $d(j)$. Using this definition $E[d_k]$ is approximately

$$\sum_{j=\frac{k}{1-p}}^{\frac{k+1}{1-p}} d(j).$$

For any $d(j)$ such that $\frac{d(j)}{d(j-1)} = 1-o(1)$, this yields $E[d_k] \approx \frac{1}{1-p} \cdot d(\frac{j}{1-p})$. For any exponential distribution with $\frac{d(j)}{d(j-1)} = c$ for some constant c, $E[d_k] = \Theta(d(\frac{j}{1-p}))$. Meanwhile for an exponential distribution, we have $\sum_{j=\frac{k}{1-p}}^{\frac{k+1}{1-p}} d(j) \approx d(\frac{k}{1-p}) \sum_{i=0}^{1/(1-p)} \gamma^{-i} = d(k/(1-p)) \frac{1-\gamma^{-1/(1-p)}}{1-\gamma^{-1}} \approx d(k/(1-p))$.

In the case of an initial power-law distribution, the new distribution is also power-law, and has the same parameter γ. Interestingly, exponential graphs behave differently for stochastic and discrete models. In the discrete model, edge removal changes the exponential base, while the base remains unchanged for the stochastic models.

Chapter 4

Trust Inference

The web has transformed into an interactive environment filled with billions of pages of user-generated content. As with anything available online this content may be accurate and useful, or it may be incorrect, misleading, or even criminally deceptive. To evaluate the likely quality of such content, we naturally look at its author. In this context trust (defined as the confidence that another will take or has taken a beneficial action [48] becomes a critical issue. Note that trust is distinct from reputation (which is often computed using a method similar to PageRank [93, 63]) because trust is personal and therefore relative to whoever is extending trust, while reputation is an attribute relative to the community as a whole. Using trust information we can help reduce risk in a commercial transaction [89], steer the user to correct information [63], or in some cases find others who are similar to oneself [104].

The immediate usefulness of trust is limited since any user only has information about those who he knows directly. We overcome this limitation through *trust inference*: the process of taking local trust judgments made by the individuals in the network and extrapolated them to all pairs of individuals in the network. The most natural method of trust inference involves asking one's friends their opinion. For large networks where most people are not connected by such short paths we require

automated techniques. One of the earliest trust inference algorithms uses a recursive variant on this idea [48]. Some early methods include network flow based models [78], but the most prominent of which are variants on spreading activation [103]. These models compute trusts from a single node at a time. They place an initial amount of trust at the starting node. Then at each time step all nodes that have more than a threshold of free trust on them keep some of it permanently and pass the remainder evenly to its neighbors.

In Section 4.1 we present a novel interpretation of trust inference developed jointly with Golbeck and Srinivasan [32] where direct trust is interpreted as an edge probability. We then assume transitivity of trust and that all edges are independent. It follows naturally that the probability of a path existing from one person to another should correspond to the inferred trust between them. These paths can be viewed as a chain of conditional probabilities. While path probabilities cannot be efficiently computed exactly (the problem is #P-Complete), we approximate them closely by repeatedly sampling the random graph as described in Section 2.

As discussed in detail in Section 2, the major advantage that this method offers over existing trust inference algorithms is the creation of a path probability metric space. This metric space (whether symmetric or asymmetric) provides a basis upon which we can apply any one of the many algorithms designed for metric spaces. Perhaps the most applicable of these are clustering algorithms which can be used to identify groups of high trust within the network. We have shown that these clusterings can be used to improve recommender system accuracy [33]. Another benefit is the ability to quickly find important edges - those whose removal would

make a significant difference in the graph or whose strengthening would greatly increase the connectedness. These are edges with path probabilities very close to the direct edge probability.

Most work on trust has dealt strictly with positive trust, where each new path can only increase trust in the network. The idea of using both positive and negative trust has been studied, and some philosophical ideas behind it have been investigated [62]. However few quantitative methods have been proposed to deal with this combination. Those that do address distrust as well as trust generally propagate them both simultaneously where they can interfere and cancel each other out [105].

Trust inference which incorporates distrust differs from positive-only trust inference in several fundamental ways. Positive trust is naturally transitive – people extend some trust to those whom their friends trust. The transitivity involved with distrust is more complicated – what is my relationship to my enemy's enemy? What about paths with exactly one edge of distrust? Also, when edges are only positive, there can be no disagreement. With distrust, positive paths can work against a negative edge. Any good solution must address these problems.

To address negative trust information, we borrow the idea of a spring embedding simulation from the graph layout literature. We start each node at a random point in a geometric space and let positive edges attract their endpoints while negative edges repel theirs. This simulated system progresses until a cutoff threshold is reached. The spring embedding method treats positive edges in a transitive manner, and also balances conflicting information. Unfortunately, two nodes may end

up very close together by chance if they are unconnected. We resolve this by using both the spring embedding distance and the estimated path probability to infer trust between two nodes. We cover the details of our algorithm in Sections 4.2.

In many applications, we infer trust as an intermediate step towards partitioning a network into clusters. A good clustering has high trust or connectivity within each cluster, and much lower trust between nodes in different clusters. In Section 4.3 we show how our spring embedding algorithm could in some circumstances recover "ground truth" cluster information from a network. Specifically we create some synthetic networks from 2-dimensional points drawn from well defined clusters. Then our algorithm reconstructs the clusters present in the original, embedded data. In Section 4.4 we develop approximation guarantees for a natural linear time clustering algorithm. These guarantees apply to any situation where sample clusterings come from any fixed probability distribution and we wish to find a one which is not too far from the weighted center of this distribution. Thus even on massive networks we can efficiently find a trust-based clustering to help with any number of applications.

4.1 Random Graph Interpretation of Trust

Social networks on the web are a major phenomenon, with hundreds of popular networks that have over a billion user-accounts between them [47]. This large corpus of relationship information has the potential to transform the way intelligent systems on the web are built. Knowing the social connections of a user allows the system to utilize data from all their friends which, in turn, facilitates a better understanding of the user's preferences. The trust relationship is particularly powerful since it speaks directly to the "quality" of a person and what they produce online. Social trust extends beyond the connections between people in social networks; it can represent the quality of a node in a P2P system or the performance of a web service.

In a large network, a given user likely knows only a small fraction of the people with whom he or she will interact; thus, the user has no knowledge of how trustworthy most people are. To use trust in this situation, methods are needed for inferring trust between users who do not know one another directly. We present a novel way of interpreting trust networks that leads to an immediate method for computing implicit trust between all pairs of nodes, including those who have no direct knowledge of each other's trustworthiness. Our goal is to take the direct trust values between individuals – values that the individuals themselves can compute given local information – and use them to infer trust values between all pairs in the population. Our approach also leads rigorously to a metric space among the users, with closer pairs corresponding to higher trust-values; this naturally leads to efficient algorithms for clustering the population.

Our method is particularly interesting because the results of the inference are not simply a best-guess at a trust value, but an implicit composite of trust and confidence. In this context, confidence is a measure of how much certainty we have in an inferred trust value based on factors like how much information the inference is based on, how trustworthy the nodes are along the paths connecting nodes, etc. Many algorithms which compute trust alone do not produce different results whether following one path or multiple identical paths (e.g. [77, 63]). Trust estimates alone do not account for how confident we are in the result of the algorithm. Rather, they use a recommender system-type approach to estimate trust as closely as possible. Our algorithm on the other hand, by requiring the direct trust information to be pessimistic, includes both an estimate of trust and a confidence component. For there to be a high degree of trust between two parties, either there must be a single path with both high trust and high confidence, or many independent paths with lower trust/confidence combinations. Each additional independent path increases our confidence in the strength of an indirect connection, and lets us give a higher rating. For many applications, an estimate that considers both components is very useful; our method is well suited for these types of tasks.

We define $t_{u,v}$ to be the direct trust between u and v (which may or may not be symmetric), and $T_{u,v}$ to be our inferred trust value. While the $t_{u,v}$ may be arbitrary, the inferred trust should obey the axioms in Table 4.1.

The idea that trust networks can be treated as random graphs drives our work. For every pair (u, v), we place an edge between them with some probability that depends on $t_{u,v}$. We then infer trust between two people from the probability that

Axioms of inferred Trust

Local Pessimism	Since $t_{u,v}$ is a pessimistic estimate, indirect information can only increase trust, thus $T_{u,v} \geq t_{u,v}$.
Bottleneck	If all paths from u to v use (a, b), then $T_{u,v} \leq t_{a,b}$, and in general the lower $t_{a,b}$ is, the lower $T_{u,v}$ should be.
Identity	Individuals should completely trust themselves: $T_{u,u} = T_{\max}$.
Complete Trust	If there exists a path $(a_0, a_i, \ldots, a_n)$ such that for all i from 1 to $n : t_{a_{i-1},a_i} = T_{\max}$, then $T_{a_0,a_n} = T_{\max}$.
Monotonicity	For any u, v such that $T_{u,v} < T_{\max}$, augmenting a graph with a new trust path from u to v, or increasing a $t_{a,b}$ value along an existing trust path should increase $T_{u,v}$.
No Trust	For any u, v with no path from u to v, $T_{u,v} = 0$.

Table 4.1: Rules for any pessimistic system that derives inferred trust from direct trust information.

they are connected in the resulting graph. Formally we choose a mapping f from trust value to $[0, 1]$. We then construct a random graph G in which each edge (u, v) exists independently with probability $f(t_{u,v})$. We then use this graph to generate inferred trust values $T_{u,v}$ such that $f(T_{u,v})$ equals the probability that there is a path from u to v in the random graph. This model is one of many that satisfies our trust axioms.

A very intuitive idea motivates this model. Consider the following scenario:

- Alice knows Bob and thinks he has an $f(t_{a,b})$ chance of being trustworthy.
- Bob knows Eve and thinks she has an $f(t_{b,e})$ chance of being trustworthy, and he tells this to Alice if he is trustworthy. If Bob is not trustworthy, he may lie about p_e and give any value to Alice.
- Alice reasons that Eve is trustworthy if Bob is trustworthy and gives her the correct value $f(t_{b,e})$ and Eve is trustworthy with respect to Bob.
- This combination happens with probability $f(t_{a,b})f(t_{b,e}) = f(T_{a,e})$ if Bob's trustworthiness and Eve's trustworthiness are independent.

Thus we view a path through the network as a Bayesian chain. Define X_{Bob} and X_{Eve} to be the respective random events that Bob and Eve are trustworthy from Alice's perspective. This is explained in more detail in Figure 4.1.

The same analysis can be used if trust is a proxy for similarity: Alice and Bob's mutual trust can be a measure of how similar they are. If trust is interpreted as a probability of being in the same category, then Alice's category is the same as

$$
\begin{aligned}
Pr[X_{Eve}] &= Pr[X_{Eve}|X_{Bob}] \cdot Pr[X_{Bob}] + Pr[X_{Eve}|\overline{X_{Bob}}] \cdot Pr[\overline{X_{Bob}}] \\
&\geq Pr[X_{Eve}|X_{Bob}] \cdot Pr[X_{Bob}].
\end{aligned}
$$

Figure 4.1: Here Alice's trust in Eve comes from Alice's trust in Bob and Bob's trust in Eve. The second term drops out because Alice has no information about Eve if Bob is not trustworthy. Furthermore, if Eve and Bob are independent, this probability becomes $Pr[X_{Bob}]Pr[X_{Eve}]$.

Eve's if (but not necessarily only if) Alice and Bob share a category and Bob and Eve share a category.

We employ large-deviation bounds to show how to quickly estimate trust between individuals even in very large, complex networks: those with exponentially many, highly correlated paths between pairs of nodes. In these examples, the Bayesian chain view still applies. If there exists a path from Alice to Eve in a random network constructed from trust values, then that path is a chain of people from Alice to Eve who each trust their successor, and Alice can trust Eve. Therefore Alice trusts Eve with the probability that there is a path from Alice to Eve in the random graph. Since it is inefficient to compute connectivity probabilities exactly, we rely on random sampling. If the true connectivity probability between Alice and Eve is p and we sample the graph k times, then kp of them will contain an Alice to Eve path in expectation. We then apply Chernoff bounds which show that when k is reasonably large, our sampled value will be very close to the actual value kp.

In fact, for any $\epsilon > 0$, if we take $k = \Theta(\frac{\log n}{\epsilon^2})$ samples, then for any pair u, v the probability that our estimate is off by more than ϵ is at most $e^{-\Theta(\epsilon^2 \log n/\epsilon^2)} = n^{-\Omega(1)}$. We then take a union bound over all pairs to bound the probability that any pair deviates by more than ϵ. If we take as few as $5 \log n/\epsilon^2$ samples this probability is at most n^{-3} for each pair. Taking a union bound over all pairs shows that with probability at least $1 - \frac{1}{n}$, $T_{u,v}$ will be within ϵ of the true value for *every* pair u, v simultaneously.

Sampling can also take into account other hard to model factors. In all of our experiments, edges in the random graphs are completely independent, however if Alice has some information that either Bob or Eve is untrustworthy but she does not know which one, she can build that into the random graph model. Thus anytime there is an edge between Alice and Bob, there will not be one between Bob and Eve and vice-versa.

In addition to having an intuitive motivation, our algorithm is also novel within the area of trust inference in the extent to which it allows us to make use of established algorithms in graph and clustering theory. Because of the graph-theoretic nature of the algorithm, we can make use of the probabilistic method as well as theory of random graphs pioneered by Erdös and Rényi [37] and heavily studied since then. Additionally, because our algorithm defines a metric space on the people in a trust network – as demonstrated in Section 4.1.2 – we obtain the flexibility and utility of a variety of metric-clustering algorithms that we can apply.

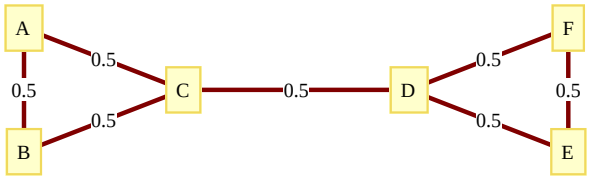


Figure 4.2: This is an example network with a critical edge. No one from the set $\{a, b, c\}$ can trust anyone in $\{d, e, f\}$ except through the mutual trust between c and d.

4.1.1 Illustrative Examples

In this section we introduce a few small example graphs to demonstrate some of the desirable qualities that our path probability formulation exhibits. In these examples trust is symmetric, however it could just as easily be asymmetric.

In our first example, Figure 4.2, the graph consists of two cliques connected by a single edge. Since any path from one clique to the other must include this edge, the trust between any two nodes in different cliques is bounded by this edge's probability. This exemplifies the bottleneck property, which protects against an adversary attempting to build unwarranted trust. Any group of adversaries can only obtain as much trust as people already in the network are willing to extend to them, regardless of how much they trust each other.

In our second example, Figure 4.3, we have a complete bipartite graph laid out in a planar fashion. The nodes a and b have no direct trust between them, instead they are connected through a sequence of common neighbors. If the trust between neighbors is uniformly p, then each possible path connecting a and b occurs with

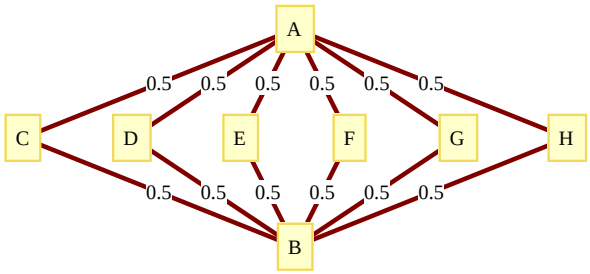


Figure 4.3: This is an example network where many weak, direct connections yield one strong, indirect connection. In this example, the path probability between a and b is $1 - \left(\frac{3}{4}\right)^6 \approx .82$

probability p^2. There are k neighbors between them, so there exists at least one such path with probability $1 - (1 - p^2)^k \approx 1 - e^{-p^2 k}$. The path probability is exponential in $p^2 k$ and thus can be very close to 1 even when p is low. This case demonstrates that a can trust b when there is a lot of independent, low-trust confirmation of b's trustworthiness. Intuitively this corresponds to Alice having lots of acquaintances who also know a little about Eve's trustworthiness. In this case either they all can vouch for Eve a little bit. Thus collectively these paths provide a strong link from a to b. Note that the path probability from a to c is not similarly magnified. Even assuming that $f(T_{a,b}) \approx 1$, the probability connecting a and c at most $2p - p^2$ which does not increase toward 1 as k increases.

4.1.2 Additional Benefits

Recall that $f(T_{u,v})$ is the probability that a path connects u and v. This brings us to our first theorem:

Theorem 8 *The function* $d(u,v) = \log \frac{1}{f(T_{u,v})}$ *defines a metric space (or for asymmetric trust, an asymmetric metric space) on the nodes as long as trust edges are independent.*

Proof:

For a function d to define a metric we require four conditions:

- $d(u,v) \geq 0$
- $d(u,v) = d(v,u)$ (this condition is not necessary for asymmetric metrics).
- $d(u,u) = 0$
- $d(u,v) + d(v,w) \geq d(u,w)$

The first condition holds because logarithms of probabilities are always negative. The second condition holds because T is symmetric. The third holds because every node always has a path of length 0 to itself. The final condition holds because any two paths in a random-graph are positively correlated. The event that there is a path from u to v is monotonically increasing in the set of edges kept in the graph – keeping an addition edge can only create additional paths. Therefore, by the FKG inequality [41], the connectivity between u and v is positively correlated with the connectivity between v and w. Finally a path from u to v and a path from v to w implies a path from u to w. Because the existence of a path is a monotone function of the edges kept, the FKG inequality [41] applies and the such paths are positively correlated. This means that $f(T_{u,v}) \cdot f(T_{v,w}) \leq f(T_{u,w})$ and thus $d(u,v) + d(v,w) \geq d(u,w)$.

Since we have a metric space on the nodes where the further apart two nodes are, the lower the probability of a path between them, we can make use of existing metric clustering algorithms to partition the nodes into groups. A clustering algorithm takes a set of points in a metric space and groups them in a way that tries to optimize some criteria. Examples include, k-centers which finds a set of k points S which minimizes the maximum distance from any point to its closest point in S, k-means which partitions the points into k sets in a way that minimizes the variance within each group, and correlation clustering which partitions the points in a way that minimizes the sum of distances within groups minus the sum of distances across groups. Each of these clustering algorithms have good approximation algorithms when applied to points in a symmetric metric space [56, 64, 8, 1], and some even have good approximations in an asymmetric metric space [3]. Figure 4.4 contains examples of clusterings based upon this trust metric.

Another major analytical benefit of our algorithm involves the ease with which key edges can be identified. A quick visual inspection of Figure 4.2 shows that the edge (c, d) is in some sense critical in that removing it would drastically alter some of the distances in the graph. Meanwhile the edge (a, b) has less importance, because of the path (a, c, b). Our technique gives a simple, algorithmic way of detecting and quantifying the importance of such edges.

For each trust edge, we define the criticality $c_{u,v}$ to be the difference between the inferred trust $T_{u,v}$, and what the inferred trust would be if the edge (u, v) did not exist, which we denote by $T'_{u,v}$. The criticality of an edge tells us how important a direct relationship is, and is parallel to centrality measures for nodes.

If an edge's criticality is small, it means that it is redundant and we can lower it's weight without changing the overall graph distances much because there are other likely paths around it. Conversely if the criticality $c_{u,v}$ is large, most of the paths from node u to node v require the edge u, v.

One way of estimating criticality would involve for each edge, removing the edge and simulating the graph some $\Theta(\frac{\log n}{\epsilon^2})$ additional times. If E is the set of all edges, this generates a total of $\Theta(\frac{|E| \log n}{\epsilon^2})$ random graphs. However if we make use of some probability theory, we only need to acquire one set of estimates on the $T_{u,v}$, and we can directly compute corresponding $T'_{u,v}$ values.

The edge (u, v) is included in the random graph with probability $f(t_{u,v})$, we denote this event by $E_{u,v}$. $E_{u,v}$ is independent of the event that any other path from u to v exists, which we denote by $P_{u \to v}$. We can analytically compute the criticality $c_{u,v}$ by:

$$\begin{aligned} f(T_{u,v}) &= Pr[E_{u,v} \vee P_{u \to v}] \\ &= Pr[E_{u,v}] + Pr[P_{u \to w} \wedge \overline{E_{u,v}}] \\ &= f(t_{u,v}) + f(T'_{u,v})(1 - f(t_{u,v})) \\ T'_{u,v} &= f^{-1}\left(\frac{f(T_{u,v}) - f(t_{u,v})}{1 - f(t_{u,v})}\right) \\ c_{u,v} &= T_{u,v} - f^{-1}\left(\frac{f(T_{u,v}) - f(t_{u,v})}{1 - f(t_{u,v})}\right). \end{aligned}$$

We consider both symmetric and asymmetric trust relationships. Both occur in networks depending on the definition of trust for the application. In social networks where trust is a social relationship between people, it is frequently asymmetric.

This is easy to understand in certain relationships. For example, small children have almost perfect trust in their parents while parents may have very little trust in those children. The asymmetry in trust may also originate from asymmetric knowledge. People may trust an expert while that expert often does not know the people who trust her, and thus she has no trust in them.

However, in other cases, it is reasonable to assume trust is symmetric. This will happen when trust is a measure of the quality of a mutual relationship or similarity, rather than a measure of the quality of an individual node. For example, in a game environment, trust may be a measure of the success two players have as a team. Measured as a rate of success, this value will be symmetric.

4.1.3 Experimental Results

We used two social networks with trust values as test networks: the Trust Project network [46] , a Semantic Web-based social network where users provide general trust ratings for their connections, and the FilmTrust social network [44] where users rate each others ability to recommend movies. In both networks, we selected the the giant component and removed nodes with a degree of 1. This left 330 nodes with 1,059 edges in the FilmTrust network. In the Trust Project network, we selected the 62 people with more than one connection in the largest connected component in the dataset and 177 connections between them.

Both of these networks contain directed edges with asymmetric trust values. For the purpose of our experiments, we worked with both the directed graph and

with a version where we converted the networks to undirected graphs with symmetric trust values. This is discussed further in section 4.1.4.

4.1.4 Symmetric Trust

Our datasets are inherently asymmetric, each trust value comes from one person rating the other, not from some mutually agreed upon value. This poses a problem: how to compute a single, mutual, and direct trust value from conflicting directed trusts that does not distort the meaning of the data too much. In some situations, this may not be possible, however with our datasets trust is similar to an estimation of similarity, which should be approximately symmetric. We resolve the issue of conflicting trust values by taking the average trust over the two directions.

Recall that our algorithm requires a mapping from surveyed trust values to edge probabilities. We try to address two major issues with our choice of this function:

- Our algorithm requires direct trust values to be pessimistic. Any nonzero trust value $t_{u,v}$ should mean that u has a definitive reason to trust v. We suspect that people in our datasets often use a value of 3 or 4 as a minimum rating, and only use lower values when they have specific evidence another person is *untrustworthy*. Conversely people are more inclined to grant high values of trust, even when there is little evidence for it. This is evident from the fact that our datasets contain very few small trust values, and many very high

values. We compensate for this by taking a nonlinear mapping from trust values to edge probabilities.

- Many people rate just a few others, while a small number of people rate many others. Someone could become one of the most trusted nodes in the graph by rating as many others as possible. Since the trust is taken to be symmetric, assigning trust to someone else implicitly assigns trust from them as well. To address this issue and not excessively reward those who rate many others, we capped the amount of outgoing trust for any node at 5 times the maximum amount of trust. The choice of 5 was fairly arbitrary, though the choice of a small constant motivated by the work of Erdös and Rènyi which showed that a random graph with more than one expected edge per node is likely to have a giant component.

We show the largest component of our first dataset in Figure 4.4. We tried many different mappings from trust to probabilities, and most yielded similar results. Looking at the graphs and the metric distance grids, you can pick out some of the natural groups. Specifically, there are three mostly red blocks (indicating high mutual trust) along the diagonal in the grid. The first such block corresponds to the left half of the graph, the next block corresponds to the top right chain in the graph, and the third block to the bottom right grouping. Also note that the node between blocks 2 and 3 is a focal node which connects the nodes in the top right to those in the bottom right of the graph.

Notice the effect that changing the trust to edge probability function has on the distances. With the $t/10$ function, trust values of 10 lead to edges with unit probability, which assures that two nodes are at the same location in the metric space. Furthermore long chains of high trust, like the top right cluster, have a high probability of being fully connected. With the function $t/20$ (where a trust of 10 is less than complete trust) on the other hand, there is a low probability that such a long chain will have its endpoints connected. Hence the top right chain no longer appears as a separate cluster. Only regions with many independent connections have low distance between all of their nodes. This effect is more pronounced at $t/40$ or smaller functions, however our clustering algorithms still pick out the two major clusters. Interestingly enough, the top right chain disappears as a separate cluster when we increase edge probabilities as well. When $p = \sqrt{t/10}$, the probability that it is connected to the cluster below it becomes high enough that they appear as a single cluster.

Next we examine the FilmTrust dataset. Figure 4.5 breaks down our results similarly to the previous dataset. It is dominated by a single, highly connected cluster. Yet our algorithm is still able to identify a few isolated groups, as well as which nodes within the cluster are loosely connected enough to be separate from the core.

4.1.5 Asymmetric Trust

When trust is asymmetric, all of the same fundamentals apply. We can still sample the random graph to estimate the probability that there is a path between two nodes with the same provable error bounds. Taking the log of the multiplicative inverse of these probabilities gives a metric (though now an asymmetric one) which we can use to cluster the nodes. However there are significant differences.

- We need a much richer graph. In the symmetric case, a large connected component is enough to make the problem interesting. However with asymmetric trust, we can have a situation (such as the graph $a \rightarrow b \leftarrow c \rightarrow d \leftarrow e \ldots$) where there are no non-trivial paths. While a dense directed acyclic graph might prove interesting, ideally a graph should be dense and contain multiple, interdependent cycles to be interesting.

- In the symmetric case a person who rates everyone else, but whom no one else has rated can become the most trusted node, so we find it useful to truncate total outgoing trust. In the directed case, this is unnecessary.

Because of the first reasons above, the smaller dataset is not particularly interesting, and we will not examine it in detail. It has only one small strongly connected component as seen in the metric distance grid in Figure 4.6. The larger dataset has much more interesting behavior. Figure 4.7 shows the results. The distance grid shows one large mutually trusting group, as well as several progressively smaller mutually trusting groups. The largest of the groups is trusted by a large portion of

the network. The second largest group is well trusted by this largest group. Beyond that, the plot where $f(t) = t/15$ brings out the most difference within the groups.

4.1.6 Applications of Clustered Networks

A clustering of a network is a partition of the nodes into meaningful groups. Intuitively, a good clustering will identify groups of nodes that are more closely connected than the graph as a whole, and where a node is more similar to the other nodes in its cluster than to the nodes in other clusters. Naturally, the goal of clustering our inferred trust network would be to partition the network such that within each group there is high trust. There are many technical definitions of a "good" clustering and there are many algorithms for each definition. We generally use a correlation clustering algorithm that groups nodes by minimizing sum of the distances withing groups and maximizing sum of the distances between groups. This seems well suited to the trust domain, but, based on the particular needs of an application, any clustering algorithm that works with a metric space can be applied to the results from our algorithm.

Once a network has been clustered, there are a number of interesting applications. First, it is extremely useful for visualization. Visualizing large networks is difficult, as is identification of important groups within them. A quick and efficient clustering algorithm that groups similar, trusted individuals together can be used to effectively display the network and support visual analysis.

Understanding trust relationships in social networks can be used to build more effective teams. In a social network, edges can represent collaborations between people. The trust can come from users who assign values indicating their judgment of their collaborator, or from more objective evaluations of the quality of the collaboration. Once we infer trust and cluster the network, the clusters represent groups of people who have a higher probability of trusting one another and working well together.

Some applications utilize trust as a background for other operations. As one example, consider trust-based recommender systems [44, 82, 92]. We show how path-probability based trust inference can improve recommendation accuracy in [33]. Clusters can also help limit the search space and optimize the list of items shown to users. Instead of considering information from the entire network, the system can focus on ratings from users in the same trusted cluster. Another example is email filtering with trust [49]. Instead of burdening the user with scores or other trust ratings from users, clusters can be a quick way to classify messages as "trusted" when they are from senders in the same cluster as the user. When the goal of the application is to highlight items that are more important, clustering is a straightforward method of classification that makes a quick first cut.

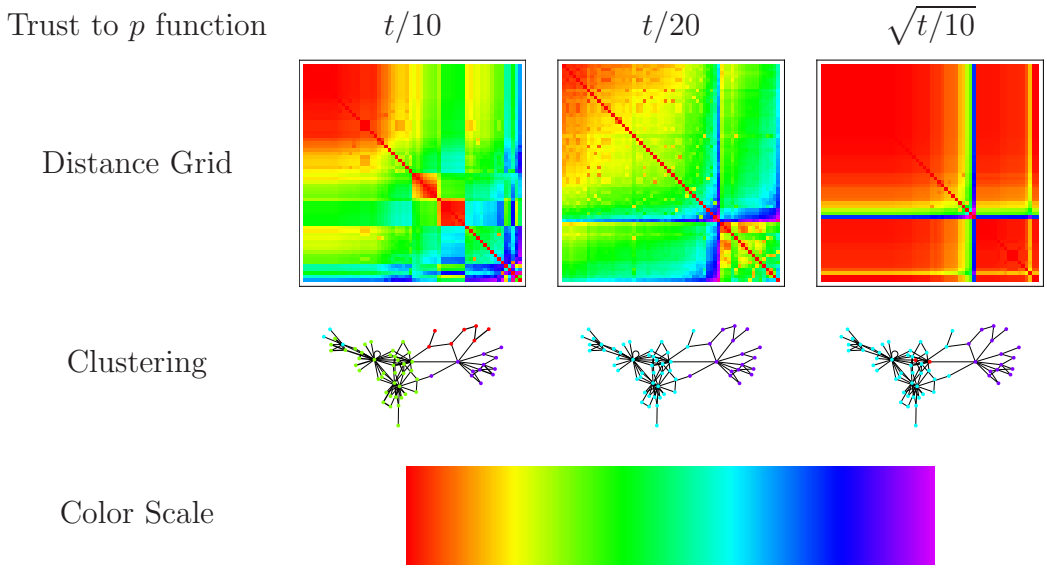


Figure 4.4: In each column, the top row gives the direct trust to edge probability function. The top figures show distances between all pairs of nodes. The distance from node u to node v is given by the color from row u column v. The grid is sorted based upon clustering first, them centrality (the sum of the distances to other nodes). The middle figures show a clustering based on the trust metric. Edge weights are proportional to trust strength., with thicker edges correspond to strong trust. In the rightmost clustering, the red nodes are outliers, and not part of a single cluster. The bottom figure is the key for the grids, distances increase linearly from a distance of 0 at red to 10 or greater at violet.

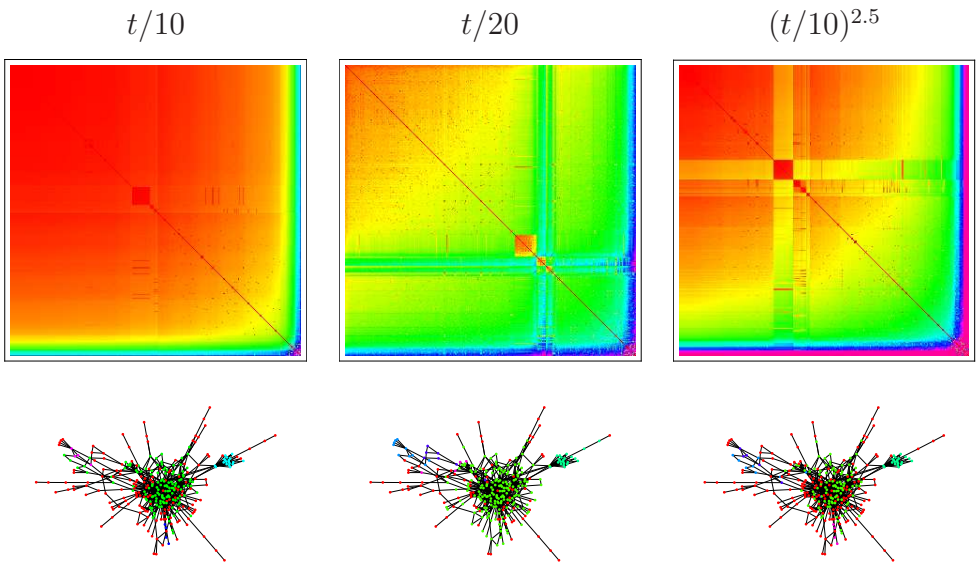


Figure 4.5: Here we present our FilmTrust results in the same format as in Figure 4.4

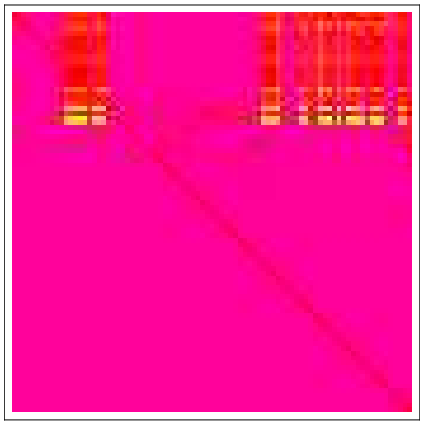


Figure 4.6: The metric distance grid for the asymmetric view of the small dataset.

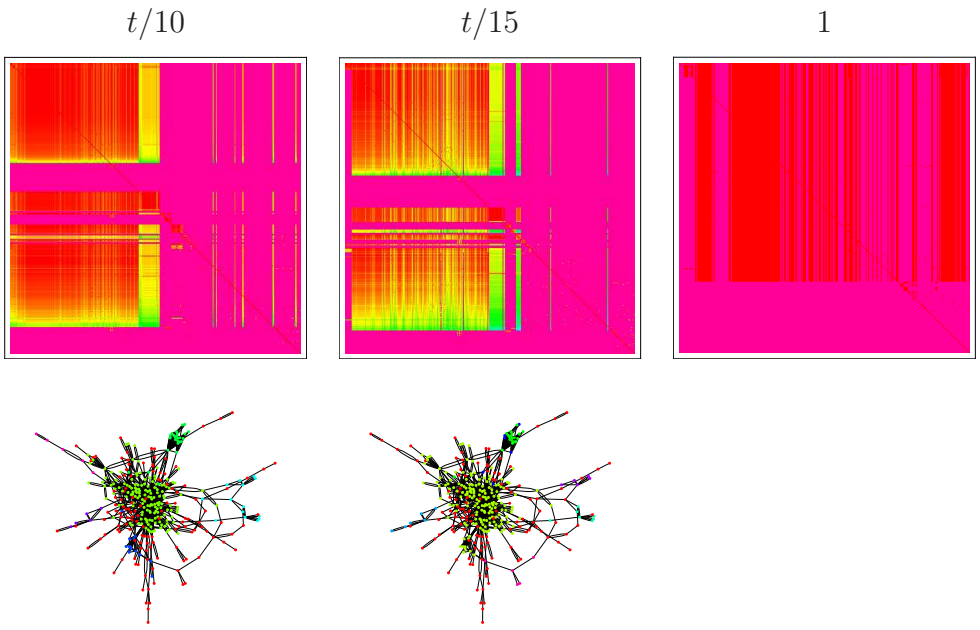


Figure 4.7: Here we show the distance grids along with partitionings for the large dataset with asymmetric trust. The rightmost grid is not probabilistic, but instead shows the transitive closure of the edge set.

4.2 Dealing with Distrust

When interacting with information or other users on the web, knowing who to trust is important, but knowing who to distrust is often more useful. Distrust, however, is much trickier to compute. While intuition and experimental evidence indicates that trust is somewhat transitive (if Alice trusts Bob and Bob trust Chuck, there is a good chance that Alice will trust Chuck, too), distrust is certainly not transitive. If Alice distrusts Bob and Bob distrusts Chuck, Chuck may be closer to Alice than Bob is, or he may be even further away. Thus, when we try to propagate distrust through a network, questions about transitivity and how to deal with conflicting information abound.

While there has been some work on distrust [52, 105], there is a significant gap in the trust inference literature. Distrust information can be useful and plentiful but there are relatively few algorithms to compute it. For example, explicit distrust can distinguish between two factions who do not know each well from those that are antagonistic. It can also expose subtleties in a trust network which are inexpressible with positive trust alone.

In this research, we present a new, efficient algorithm for effectively computing trust and distrust in web-based social networks. With three real world datasets that include trust and distrust ratings from users, we conducted a series of experiments that show we can achieve 80-90% overall accuracy in computing relationships hidden in the original networks.

Recent work on edge sign prediction by Leskovec, Huttenlocher, and Kleinberg [72] is a direct predecessor of our work. They examine the same three networks as we do with a localized view. To predict the sign of an edge they look at the positive and negative edge counts of its endpoints, plus the number and type of triads containing this edge. These local factors form a high dimensional space on which they perform standard machine-learning techniques to determine how to predict the sign of unknown edges. From a theoretical perspective they interpret their results through Heider's balance theory [55], which states that unbalanced triads (those with an odd number of negative edges) are unstable. Experimentally they show good edge sign prediction results for all three datasets (accuracy rates between 80-90% over all edges), with better results on edges with a higher embeddedness - those which are a part of a greater number of triads.

4.2.1 Dataset descriptions

We use three major social network datasets from the Stanford Large Network Dataset Collection [1] to test our methods. These networks have both positive (trust) and negative (distrust) edges which are unweighted, though our methods could easily support weighted trust and distrust. A description of each of the three networks follows:

- Wikipedia moderator elections - Wikipedia, the popular online encyclopedia created by users, has a set of elected moderators who monitor the site for

[1] http://snap.stanford.edu/data/

quality and controversy and who help maintain it. These moderators receive extra administrative privileges, and thus must be trusted by the community. When a user requests administrator access, a public discussion page is set up for users to discuss and vote on whether to admit the moderator. Positive and negative votes are counted as positive and negative trust ratings. Note that in this network, if a user is not ultimately voted in, they will not appear in the graph. Thus, positive trust ratings (or positive votes) will be more common in the graph. The data was pulled from the discussion pages in January 2008 [73, 72]. It contains just over 7,000 nodes and 100,000 edges.

- Slashdot - This is a technology news site where users can rate each other as friend or foe. We treat those as positive and negative trust ratings. The dataset contains over 77,000 nodes and just under 900,000 edges. Use used the version released in February 2009 [74].

- Epinions - This is product review site where users choose whether to trust or not trust one another based on their ratings and reviews of products. The network has over 75,000 nodes and 500,000 edges. The dataset was collected and released in 2003 [95].

4.2.2 Algorithm and Methodology

We previously described a method for computing trust based on path probability in random graphs. For every pair (u, v), we place an edge between them with some probability that depends on the direct trust value between them which we

denote by $t_{u,v}$. We then infer trust between two people from the probability that they are connected in the resulting graphs. Formally we choose a mapping f from trust value to probabilities. We then construct a random graph G in which each edge (u, v) exists independently with probability $f(t_{u,v})$. We then use this graph to generate inferred trust values, $T_{u,v}$, such that $f(T_{u,v})$ equals the probability that there is a path from u to v in the random graph. In addition to having an intuitive appeal, we found this approach to work well in practice.

Distrust, however, is more complex. While trust can be considered transitive, distrust is not. Additionally with only positive trust, there are no mathematical inconsistencies in the data. When we incorporate distrust, there can be paths which disagree. We propose using a modified graph layout algorithm to find a low-dimensional embedding of the graph which tries to reconcile the conflicting information and transitivity. We are inspired by spring embedding graph layout algorithm [34]. This type of algorithm simulates the physics of springs in a 2D or higher dimensional space. In the graph layout version, edges between nodes are treated as springs that pull nodes together together, but reasonable space is maintained between nodes by making them repel one another. Nodes are randomly laid out in an initial configuration, and the system iterates until a stable equilibrium is reached or some short-circuit condition happens (maximum iterations, changes per timestep below a threshold, etc).

Our first attempt to incorporate distrust resolves conflicting trust/distrust information through a nonlinear optimization. We assume that all users' trust estimates are noisy, and we want to find the true ones. In this model, positive trust

corresponds to edge probabilities, while negative trust corresponds to upper bounds on path probabilities. We then apply a cost function for each edge of the deviation between the "true" value and the "measured" value. From there we find a globally minimal cost solution which does not have any conflicting trust/distrust information and infer trust from it. Unfortunately this technique scales very poorly.

This led us to develop a spring-embedding algorithm which we use in conjunction with our path probabilities technique to infer trust. We compute path probabilities using only positive edges. Independently, we use an iterative spring embedding algorithm – where positive edges attract and negatives repel – to resolve competing trust/distrust information. Note that in the face of positive trust only, this results in all nodes very close to each other which provides no meaningful information. A spring-embedding algorithm implicitly has the transitivity and conflict resolution properties we desire as well as the scalability necessary to handle very large datasets.

We adapt the spring embedding algorithms to our trust systems. Instead of having all nodes repel, we add a repelling force between nodes connected with a negative edge. Trust maintains transitivity here because two nodes with a shared friend are both pulled towards that friend. If they share two friends who are co-located, they are pulled with twice as much force. If they have a shared enemy, they are both pushed away (which may or may not move them in the same direction). If one is friends with an enemy of the other, the forces will push them into different locations. This modified spring-embedding algorithm also deals well with conflicting information. If node A has two friends who disagree about node B, the friends will

be pushed apart, and A will be partially pulled towards and partially pushed away from B.

One potential drawback is that two nodes may be placed close together by chance though they have little trust between them. This is why spring embedding alone is not enough - we need to consider path probabilities as well. This combined approach forms the basis of our new method. We can independently compute path probabilities and spring embedding distances for our entire graph. For each edge or potential edge, we record the path probability between its endpoints as well as their embedded distance. Thus each edge corresponds to a two dimensional vector whose position indicates the amount of trust between its endpoints.

To assess our algorithm's quality, we apply it to the edge sign prediction problem. For each of our three datasets, we remove a substantial number of edges (500 in Wikipedia and Slashdot, 1000 in Epinions) chosen uniformly at random. The removed edges make up the testing set and the kept edges make up the training set.

Using the training set with the test edges removed, we perform parameter tuning and compute path probabilities and spring embedding distances. For the path probability algorithm, this tuning consists of choosing a probability p that corresponds to a positive edge. In all three datasets we settled on $p = 0.05$, which gives path probabilities for the edge's endpoints spread across the range $[0, 1]$. For the spring embedding algorithm, tuning means selecting the force functions for both positive and negative edges and choosing the dimensions of the embedding space. We found that an attractive force proportional to r^2 and repelling force proportional

to $1/r^2$ gave good distributions of points, and we chose the four dimensional unit cube for our embedding space. For every edge in the training and test sets we then record its sign as well as its endpoint path probability and embedded distance.

We bucket the list of training edges into intervals based upon their path probability, and for each interval we find the embedded distance which minimizes the maximum of the ratio of mislabeled positive edges and the ratio of mislabeled negative edges.

We then use these values to classify edges in the testing set. For every edge in the testing set, we find the interval that corresponds to their path probability. If they are embedded closer than that interval's cutoff, we guess that they are positively connected, and if they are embedded further than the cutoff, we guess that their edge is negative. Note that all of these test networks are biased with many more positive than negative edges. Therefore our goal is not simply to have the highest ratio of correctly classified edges, but rather to correctly classify both positive and negative edges simultaneously. On such a biased dataset an algorithm which classifies edges randomly could perform quite well overall (by simply always choosing the dominant category), however the better it did on the positive edges, the worse it would do on the negatives.

4.2.3 Results

We ran our algorithm over all three datasets. For each, we computed a separator to classify positive and negative trust relationships. This is shown as a red

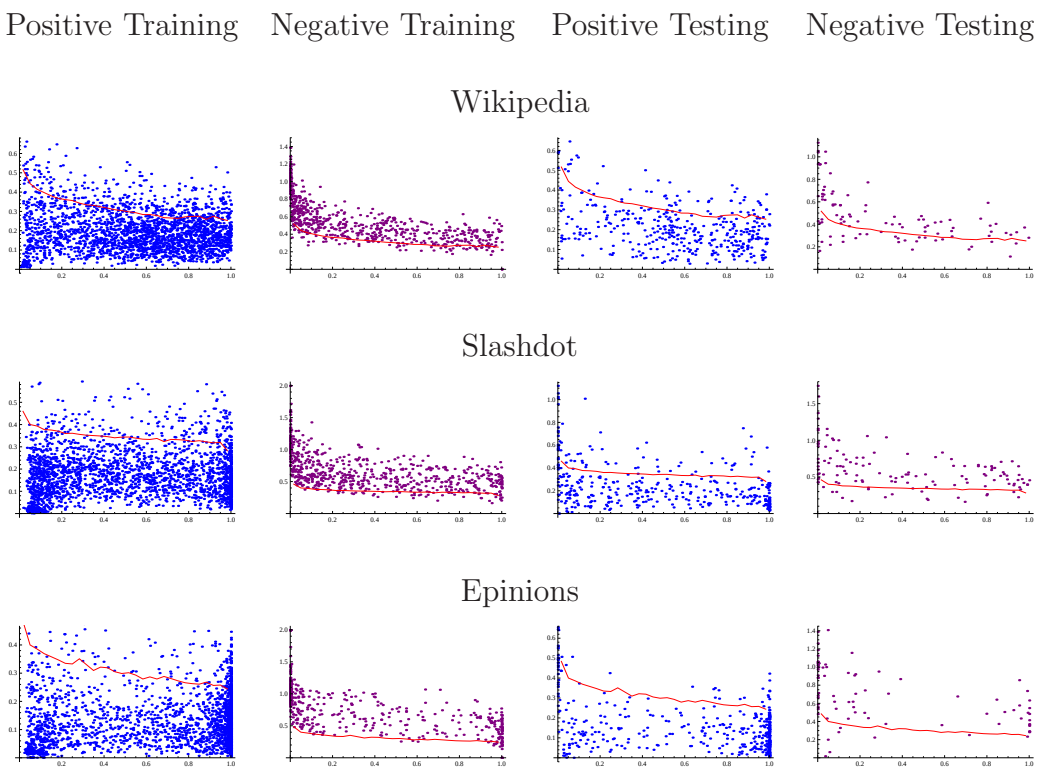


Figure 4.8: The positive and negative edges with the classification line for all three datasets. Each point in the figures corresponds to an edge in the graph. The horizontal axis is the path probability (larger values mean endpoints that are closer) and the vertical axis shows the embedded distance (smaller values mean endpoints that are closer). Points below the classification line are positive edges (or classified as such) and those above are negative. For clarity in viewing the larger datasets, not every point is displayed. Rather we display a random subset of several thousand of the points.

	Wikipedia	Slashdot	Epinions
Positive edges	0.78	0.77	0.85
Negative edges	0.22	0.23	0.15
Training edges correctly classified	0.86	0.92	0.94
Positive test edges correctly classified	0.81	0.81	0.89
Negative test edges correctly classified	0.78	0.84	0.89
Correct positive classifications	0.93	0.94	0.98
Correct negative classifications	0.51	0.60	0.61
Total edges correctly classified	0.80	0.82	0.89

Table 4.2: The fraction of correct classifications for various criteria.

line in the charts of Figure 4.8. Positive edges are correctly classified when they are below the line and negative edges are correct when they are above the line. Note that the noisy nature of the datasets means that we are not able to correctly classify all the edges in our training sets. The first two columns of Figure 4.8 show that there are a number of points that appear on the wrong side of the line. Table 1 provides the actual percentages; we are able to correctly classify between roughly 85-92% of both positive and negative edges in the training set. This gives us a baseline against which we can compare our results from the test set.

The third and fourth columns of Figure 4.8 show the data points for the test set, and the results are detailed in Table 1. We achieve good results in classifying trust edges, with approximately 80% of the edges in each category being predicted correctly.

However, our performance if more skewed when viewed in the context of confidence in our predictions. When our classifier predicts a positive edge, we are quite certain (over 90%) sure of the results. However our confidence in a negative prediction is between 50-60%. This bias comes from the fact that the dataset is unbalanced, with many more positive edges than negative, and may be inherent in the problem.

4.2.4 Discussion

Our results exhibit good self-consistency by performing well with respect to our classifier. Overall, the results are generally quite good, and compare well with

[72] which uses more, but all fairly local, pieces of information to make the same kind of predictions. Our approach achieves a similar level of accuracy with a more holistic approach, one that reduces the complex interdependencies in the network into two related inferred trust values rather than a high dimensional vector of properties. Furthermore, [72] notes poorer results with edges which contribute to few or no triads. While this is expected in general (nodes with less direct information about their relationship should be harder to classify), our approach does a better job of using the information that is available in these cases - those trust paths of length greater than two.

At a higher level, the fact that we are able to achieve these good results speaks to the suitability of trust inference algorithms in general. Trust is a complex social relationship, and we are using realistic datasets with trust edges created by real users. It is not unreasonable to question whether or not trust can be accurately inferred at all, since it is so fuzzy and personal, or to question whether our probabilistic treatment is a proper one. Furthermore, distrust adds a new and difficult element; how to properly treat it is an open question within the trust community. The fact that we were able to classify trust as positive or negative with such a high rate of success means not only that our algorithms work well, but that the underlying data is compatible with our treatment of inferred trust.

4.3 Cluster Reconstruction

Often when we invoke a trust inference algorithm, we want to cluster the network's users in some meaningful way. The fact that we can use inferred trust distances to improve recommendation accuracy [33] and predict the signs of hidden edges suggests that the clusters produced are in some sense "good". However, good results do not necessarily mean that our clusterings correspond to some natural partition of the data. In a real social network, people can self-organize into groups based on organizational memberships, geography, shared interests, political ideology, etc. Can our trust inference algorithms find such clusters, assuming that they even exist?

Unfortunately, as of this writing we have not found any rich social network data with trust scores and well defined clusterings, though we are currently evaluating a candidate network. In order to study how well and under what conditions we can reconstruct a cluster, we generate several synthetic networks with various parameters and attempt to reconstruct their underlying clusters. We do not claim that these networks closely resemble real social networks. Rather our work in this section is a proof-of-concept.

Since we focus on cluster reconstruction, we must start with well defined clusters. We model each cluster as a 2-dimensional Gaussian and randomly sample points from its distribution. This gives us a series of points, each of which belongs to a well defined cluster with a 2-dimensional embedding. We then create edges between these points in two ways. Every node gets some number of positive edges

chosen uniformly at random into its own cluster. We also create edges where two endpoints are chosen uniformly at random, and we randomly assign the edge to be positive or negative. For each of these edges, the further apart its endpoints, the more likely it is to be negative.

In each test in this section, we create three or four partially overlapping Gaussian clusters. We experiment with several different centers and standard deviations, including some clusters with standard deviations much higher than the distance between their centers. For some $c \in [1, 2]$ we give each node c expected outgoing positive edges to other nodes in the same cluster, and 1 expected outgoing edge to a random node in the entire dataset.

We show the results from each of our three synthetic datasets in Figures 4.9, 4.10, 4.11, 4.12, 4.13, and 4.14. These figures show the initial distribution of points, the average number of intra-cluster out edges generated, and a histogram showing edge distances for positive and negative edges (blue and magenta respectively). Then for a few sets of algorithm parameters we show the result of spring embedding, a scatter plot showing how the distances in the original dataset compare to the reconstructed distances, and the correlation between those distances. We obtain correlations and the associated plots by randomly sampling 1000 pairs of points (enough to accurately capture the algorithm's behavior) rather than examining all $\binom{n}{2}$ such pairs.

From a visual inspection we can tell that our algorithm does a good job at finding the clusters from the original datapoints. As long as c is high enough (which appears to be 1 or slightly higher) our spring embedding algorithm separates the

points largely into their original clusters. When $c = 1.4$, it often produces results which are better separated than the clusters in the original graph. We can also see that tuning the spring embedding parameters has a substantial effect on the final points, but a small effect on the resulting clusters. Parameters favoring attractive forces tend to produce fuzzier cluster boundaries. Those that favor repellent forces produce sharper boundaries, but nodes with neighbors in different clusters stand out more.

While our algorithm appears to do well at clustering, it performs only moderately well at distance reconstruction. The correlations between the original distances and the reconstructed distances in our examples varies from less than .1 to almost .5. We do not view this as a drawback. Two points very close together in the original embedding are not much more likely to be connected by a short positive path than two points from the same cluster with are far apart. Since our algorithm takes as input only the graph and not the original points, it has no way to infer that these points should be especially close. In fact, the mean correlation between points from different clusters explains much of the total correlation in distances.

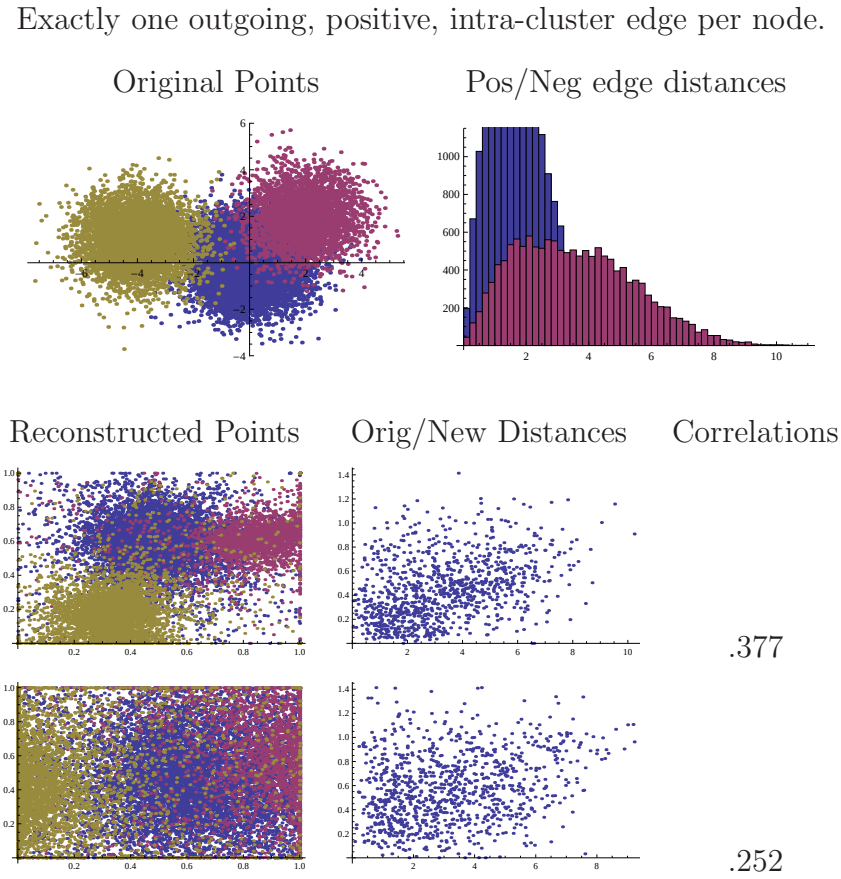


Figure 4.9: Here we show cluster reconstruction results for our first set of generated points. The points come from three Gaussian distributions which overlap slightly with each other. Each cluster has its own color which we use consistently throughout the plots. We show the original points, a histogram of how many positive and negative edges have any given endpoint distance, reconstructed points, a scatter plot showing how original distances relate to reconstructed distances, and the correlation between original and reconstructed distances. In the first row, positive and negative edges produce forces that are $O(d)$ and $O(d^{-1})$ respectively. In the second row they are $O(d)$ and $O(d^{-2})$.

In expectation $c = 1.4$ outgoing, positive, intra-cluster edges per node.

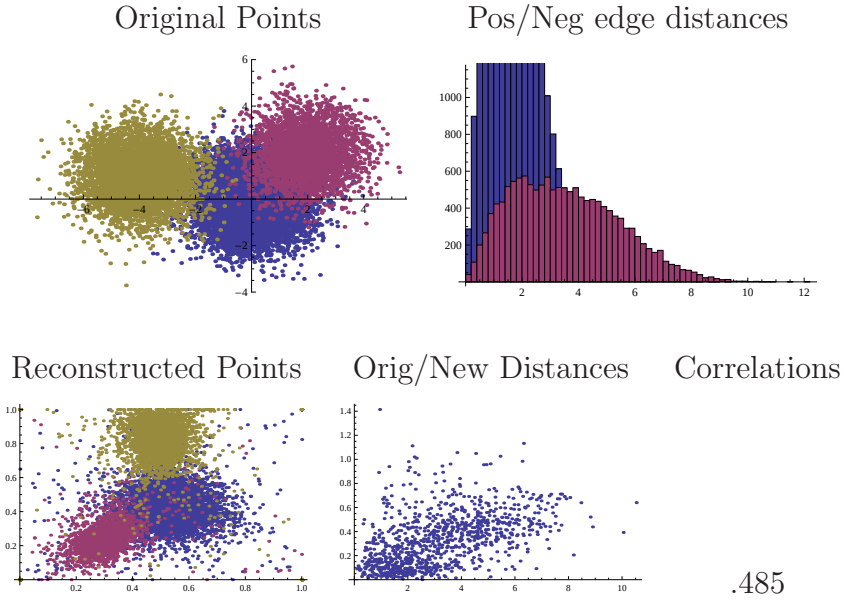


Figure 4.10: Here we show cluster reconstruction results for our first set of generated points, but using more intra-cluster edges than Figure 4.9. Positive and negative edges produce forces that are $O(d)$ and $O(d^{-1})$ respectively.

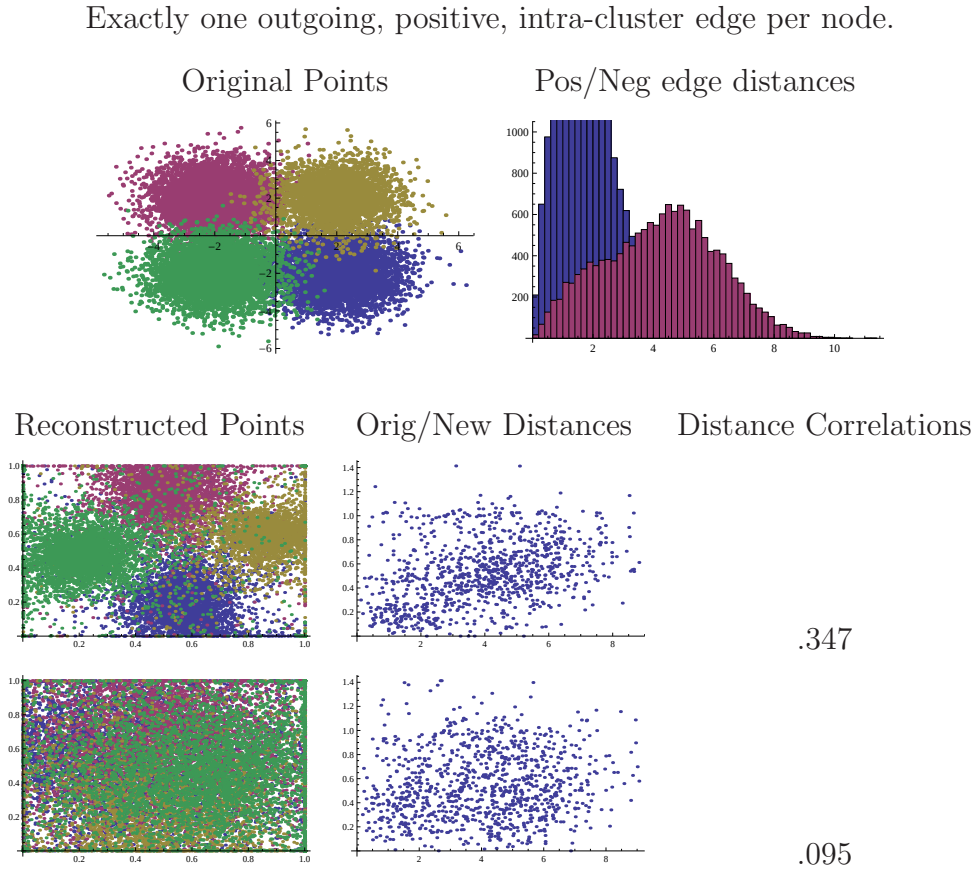


Figure 4.11: Here we show cluster reconstruction results for our second set of generated points. The points come from four equally spaced Gaussian distributions which overlap slightly with each other. In the first row, positive and negative edges produce forces that are $O(d)$ and $O(d^{-1})$ respectively. In the second row they are $O(d^2)$ and $O(d^{-3})$.

In expectation $c = 1.4$ outgoing, positive, intra-cluster edges per node.

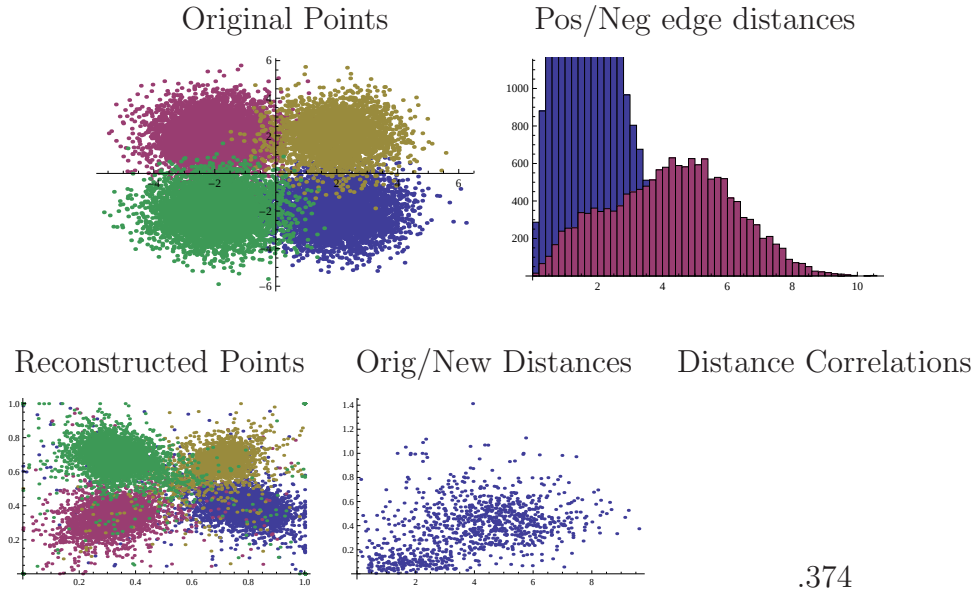


Figure 4.12: Here we show cluster reconstruction results for our second set of generated points, but using more intra-cluster edges than Figure 4.11. Positive and negative edges produce forces that are $O(d)$ and $O(d^{-1})$ respectively.

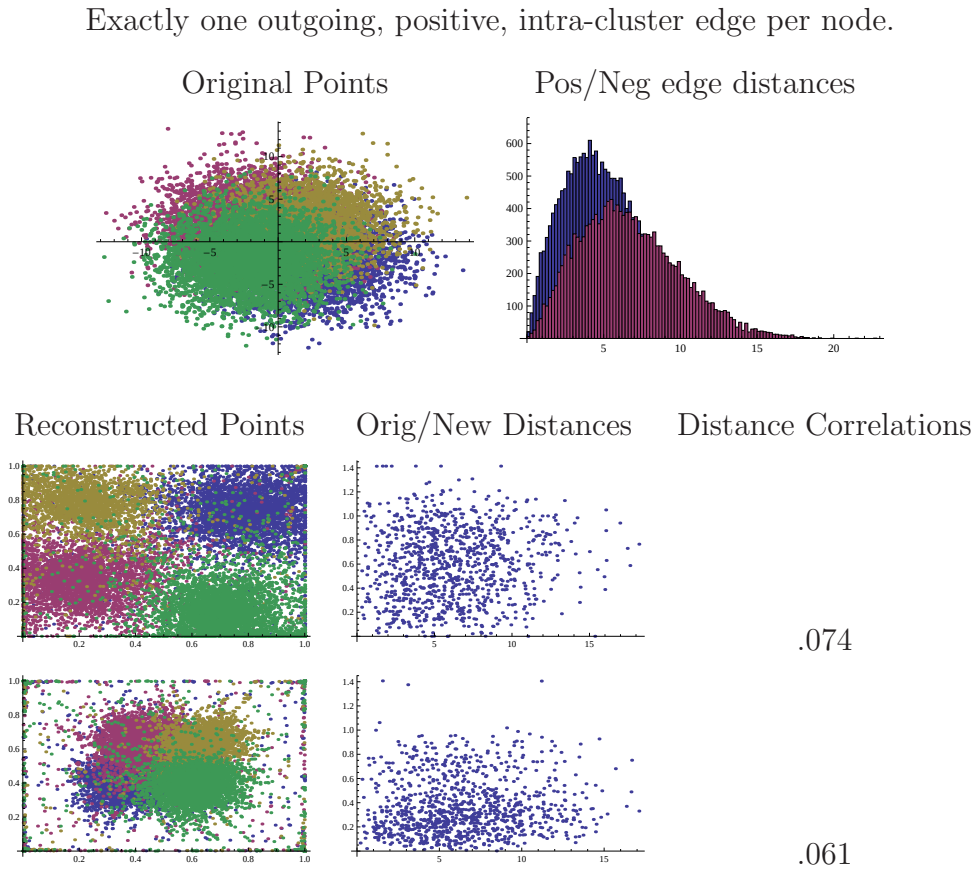


Figure 4.13: Here we show cluster reconstruction results for our third set of generated points. The points come from four very closely spaced Gaussian distributions which overlap significantly. In the first row, positive and negative edges produce forces that are $O(d)$ and $O(d^{-1})$ respectively. In the second row they are $O(d^2)$ and $O(d^{-3})$. Notice that our reconstructed points do not approximate the pairwise distances in the original graph, however it prominently separates the clusters even when no such separation existed originally.

In expectation $c = 1.4$ outgoing, positive, intra-cluster edges per node.

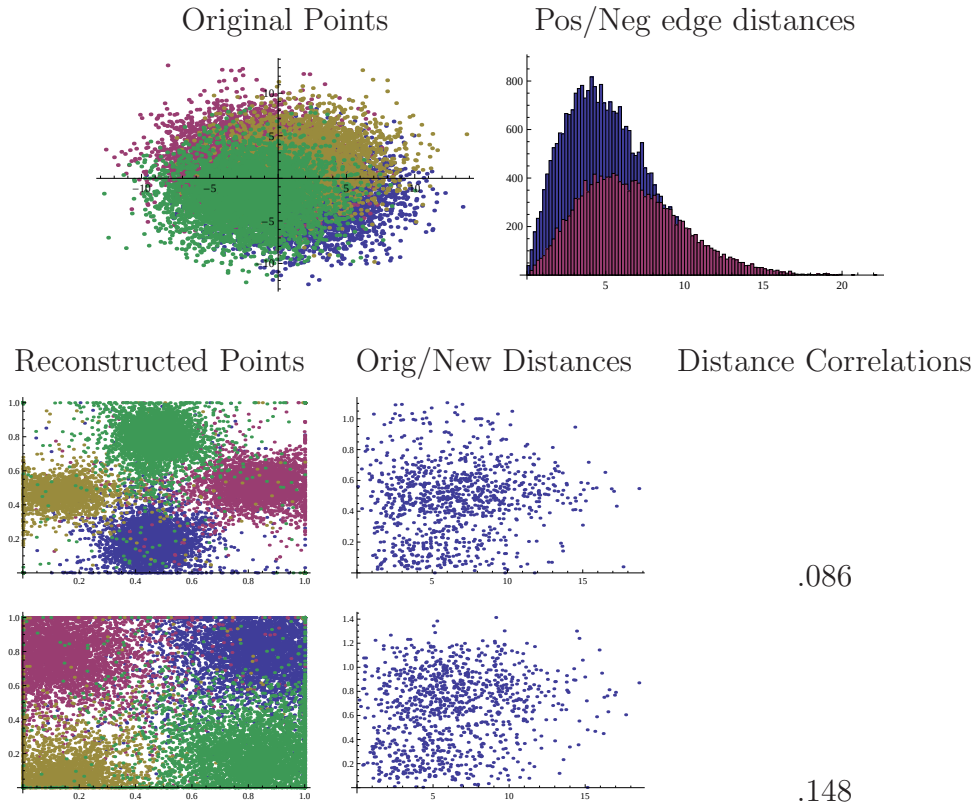


Figure 4.14: Here we show cluster reconstruction results for our third set of generated points, but using more intra-cluster edges than Figure 4.13. Positive and negative edges produce forces that are $O(d)$ and $O(d^{-1})$ respectively in the first row, and $O(d)$ and $O(d^{-2})$ respectively for the second.

4.4 Clustering Using Only Black Box Sampling

Clustering is an important challenge in the context of trust inference. All the applications discussed above: recommender systems, team formation, visualization, and many more benefit from finding groups which are tightly connected internally, and loosely connected to the remaining network. When working with internet-scale networks, even the most efficient clustering algorithms are space and time intensive. Motivated by these applications, as well as those unrelated to trust, our research addresses the issue of clustering the vertices in graphs in the most efficient possible way.

Here we present a new method for graph clustering that clusters any set from which we can sample random partitions. Our algorithm is computationally efficient - it needs only a single pass over the network. This algorithm defines a distance function between any two clusterings and attempts to find a clustering C where the expected distance between C and a randomly sampled clustering is small. We develop an approximation algorithm to find good clusterings in expectation directly by sampling the random graph only once. We then show that repeated sampling improves our confidence in the result. In Section 4.4.1 we formalize the problem and prove that a single random sample gives a 3-approximation in expectation. In Section 4.4.2 we show how to use multiple samples to improve on our probabilistic guarantees. Finally in Section 4.4.3 we apply our new algorithm to trust inference clustering as a demonstration of its usefulness.

Recommender systems are a common application and inferred trust values can be used in place of traditional user similarity measures to compute recommendations (e.g. [92, 5, 50]). In [42], the authors present a technique for using trust to estimate the *truth* of information that is presented, which in turn has applications for assessing information quality, particularly on the Semantic Web. Other applications of that idea include using trust for semantic web service composition [71].

Often these algorithms require, as an intermediate step, finding clusters of people who are more tightly connected to each other than to the remainder of the population [96, 33]. The art of finding useful sets of clusters has been well studied on a wide range of applications. In some cases there is some (unknown) "ground-truth" clustering inherent in the data which we want to find, and the algorithms attempt to find a clustering that is "close" to the true one [30, 7]. Often though, there is no reason to believe that the data has inherently correct clusters, and the goal becomes simply to produce a clustering which works well in practice for a particular application.

When each data point to be clustered consists of a vector of numerical values, one common technique involves choosing a distance function between the elements (Euclidean, L1-norm, etc.) and looking for clusters which minimize some optimization function. Examples of these algorithms include k-means [53] (which minimizes the mean squared-distance of elements from their cluster centers), and k-centers [56] (which minimizes the maximum distance from any point to the center of a cluster). Typically approximation algorithms, which find solutions close to optimal, are used

because it is impractical to compute the optimal clustering for these problems. For a more extensive overview of various clustering algorithms, see [102].

4.4.1 The Algorithm

In our path-probability based trust framework, direct trust corresponds to an edge probability and indirect trust between any two people in the network corresponds to the probability that there is a path between them. Using our path-probability metric space we can apply various well-studied clustering algorithms to the resulting distances. In order to have confidence in our path probability estimates are accurate to within a tolerance of $\pm\delta$, $O(\frac{\log n}{\delta^2})$ samples are needed on a network with n people. This poses a major drawback on internet-scale datasets, which can easily have millions of users.

However there is nothing inherent in clustering which requires computing distances. In the context symmetric trust, we present a computationally efficient clustering algorithm. Our new clustering algorithm algorithm takes a single sample of the random graph and uses its connected component decomposition as the clustering. Sampling the graph and computing the connected components can be done simultaneously in a single pass over the edges of the graph using depth first search, and thus is as fast of an algorithm as we can hope for.

Of course a fast algorithm that produces poor clusters would not be useful. Thus our main result is to derive probabilistic bounds on the quality of the resulting clusterings. We start by defining a distance function between clusterings with the

goal of minimizing the distance between our chosen clustering and a connected component decomposition of an instance of the random graph. We do not expect to be able to find the best such clustering (and in the general case where sample clusters come not from a random graph but from a black box it is not possible), however our single sample algorithm achieves a 3-approximation in expectation. This means that a random clustering will, on average, produce distances no more than 3 times those produced by the best clustering. We also show that using a closely related distance function (one used by Balcan et al. [7]) our algorithm achieves a 2-approximation in expectation. Finally we show how to achieve improved probabilistic bounds by sampling multiple times instead of only once.

Consider the following problem:

- A clustering of a set U is a set C of subsets of U such that $\forall x \in U, |\{s \in C : x \in s\}| = 1$. In other words, a clustering is a set of disjoint subsets whose union is the entire set U. For convenience we let a clustering contain an arbitrary number of distinct empty sets.

- Given a set U, and a probability distribution $\mathcal{P}$ on clusterings of U, we want to find a clustering X which minimizes the expected distances between X and a random clustering drawn from $\mathcal{P}$.

- There are many possible distance functions between clusterings, we will concentrate on the following one: for two clusterings X and Y, define $D_X(Y)$ to be $\min_{f:Y\to X} \sum_{s\in Y} |s \cup f(s)| - |s \cap f(s)|$. In other words, for each set in Y, we match it to the set in X which minimizes the size of the symmetric difference

between the two. We will later consider a similar distance function proposed by Balcan et al. [7]. They let the distance between two clusters be the minimum number of elements not in the matched clusters under any matching of the clusters. For any two clusters, the distance using our metric is at least the distance using theirs, and at most twice the distance using theirs.

Note that while we do not restrict the function f, the optimal choices for f are bijections (when we include an appropriate number of empty sets in the two clusterings). This follows from the observation that for any optimal f, and for all $s \in X$, $s \cap f(s)$ is at least half the size of both s and $f(s)$. Otherwise it would be better to map s to the empty set. Since no two distinct $s_1, s_2 \in X$ can both share more than half of the elements of a single $y \in Y$, f must be one-to-one. By mapping extra, distinct empty sets onto the remaining sets in Y (if any), f becomes a bijections.

For any given probability distribution on clusterings $\mathcal{P}$ (such as the one given by the connected component decomposition of a random graph), define random variable Y to be a clustering drawn from that distribution. Let C be a clustering that minimizes the expectation of $D_C(Y)$. Since the distribution $\mathcal{P}$ is a black box in general, we cannot hope to find the actual clustering C even with arbitrary running-time (because we can not distinguish between two distributions with complete certainty). However the following simple algorithm surprisingly gives a 3-approximation in expectation:

Take a random sample C' from $\mathcal{P}$, and use that as the approximation.

The analysis proceeds as follows:

Let $U, \mathcal{P}$, and C be given. Define $g_X(u)$ to be the set s in the clustering X that contains u and f_X^Y to be the best function mapping clusters in Y to clusters in X. The expected cost of the optimal solution is $E[D_C(Y)]$

$$
\begin{aligned}
&= E[D_C(C')] \\
&= \sum_Y Pr[Y] \cdot \sum_{s \in Y} (|s \cup f_C^Y(s)| - |s \cap f_C^Y(s)|) \\
&= \sum_{u \in U} \sum_Y Pr[Y] \cdot \begin{cases} 0 & f_C^Y(g_Y(u)) = g_C(u) \\ 1 & f_C^Y(\{\}) = g_C(u) \\ 2 & \text{ow.} \end{cases} \\
&= \sum_{u \in U} \left(\begin{array}{c} \sum_{Y : f_C^Y(g_Y(u)) \neq g_C(u) \wedge g_C(u) \neq f_C^Y(\{\})} Pr[Y] \\ + \sum_{Y : f_C^Y(g_Y(u)) \neq g_C(u)} Pr[Y] \end{array} \right). \qquad (4.1)
\end{aligned}
$$

Each element u adds to the total cost only if its set in Y does not map to its set in C. In that case it costs 1 because of the mapping from $g_Y(u)$ to $f_C^Y(g_Y(u))$, and it costs another 1 if some non-empty set in Y maps to $g_C(u)$.

The expected cost, $E[D_{C'}(Y)]$, of our approximated solution is derived as follows:

$$
\begin{aligned}
&E[D_{C'}(Y)] \\
&= \sum_{C'} Pr[C'] \cdot \sum_{Y} Pr[Y] \cdot \sum_{u \in U} \begin{cases} 0 & f^Y_{C'}(g_Y(u)) = g_{C'}(u) \\ 1 & f^Y_{C'}(\{\}) = g_{C'}(u) \\ 2 & \text{ow} \end{cases} \\
&\leq \sum_{C',Y} Pr[C' \wedge Y] \cdot \sum_{u \in U} \begin{cases} 0 & f^Y_C(g_Y(u)) = g_C(u) = f^{C'}_C(g_{C'}(u)) \\ 1 & f^Y_C(\{\}) =? f^{C'}_C(g_{C'}(u)) \\ 2 & \text{ow} \end{cases} \qquad (4.2) \\
&= \sum_{u \in U} \begin{pmatrix} \sum_{C',Y: f^Y_C(g_Y(u)) \neq g_C(u) \vee f^{C'}_C(g_{C'}(u)) \neq g_C(u)} Pr[C' \wedge Y] + \\ \sum_{C',Y: (f^Y_C(g_Y(u)) \neq g_C(u) \vee f^{C'}_C(g_{C'}(u)) \neq g_C(u)) \wedge f^Y_C(\{\}) \neq f^{C'}_C(g_{C'}(u))} Pr[C' \wedge Y] \end{pmatrix} \\
&\leq \sum_{u \in U} \begin{pmatrix} \sum_{C',Y: f^Y_C(g_Y(u)) \neq g_C(u) \vee f^{C'}_C(g_{C'}(u)) \neq g_C(u)} Pr[C' \wedge Y] + \\ \sum_{C',Y: f^Y_C(g_Y(u)) \neq g_C(u) \wedge f^Y_C(\{\}) \neq f^{C'}_C(g_{C'}(u))} Pr[C' \wedge Y] + \\ \sum_{C',Y: f^{C'}_C(g_{C'}(u)) \neq g_C(u) \wedge f^Y_C(\{\}) \neq f^{C'}_C(g_{C'}(u))} Pr[C' \wedge Y] \end{pmatrix} \\
&\leq \sum_{u \in U} \left(\sum_{Y: f^Y_C(g_Y(u)) \neq g_C(u)} 3Pr[Y] + \sum_{Y: f^Y_C(g_Y(u)) \neq g_C(u) \wedge f^Y_C(\{\}) \neq g_{C'}(u)} Pr[Y] \right). \qquad (4.3)
\end{aligned}
$$

Where Equation 4.2 maps $s \in Y$ to $s' \in C'$ if and only if they both map to the same subset in C. This mapping must cost at least as much as the optimal mapping $f^Y_{C'}$. Dividing Equation 4.3 by Equation 4.1 gives $E[D_{C'}(Y)/D_C(Y)] \leq 3$.

We demonstrate that this upper bound is tight with the following distribution on clusterings:

$$Pr[Y = \{\{1\}, \{2\}\}] = (k-1)/k, Pr[Y = \{\{1,2\}\}] = 1/k.$$

The optimal solution simply matches the high probability case, $C = \{\{1\}, \{2\}\}$. The expected cost of this solution is $((k-1) \cdot 0 + 1 \cdot 1)/k$. The expected cost of using a random sample is

$$\frac{(k-1) \cdot \left(\frac{k-1}{k} \cdot 0 + \frac{1}{k} \cdot 1\right) + 1 \cdot \left(\frac{k-1}{k} \cdot 2 + \frac{1}{k} \cdot 0\right)}{k}$$

which reduces to $3 \cdot \frac{k-1}{k^2}$, and thus approaches 3 times optimal as $k \to \infty$.

We briefly consider the case where we change the distance function between clusterings to

$$D_C(Y) = \min_f |\{u : f(g_Y(u)) \neq g_C(u)\}|$$

(as used by [7]) and keep all definitions and notations the same as above. This distance function costs exactly 1 for each element u whose set in Y is not mapped to its set in C. Using this metric, our algorithm yields a 2-approximation in expectation. We show this by rewriting the distance function as

$$E[D_C(Y)] = \sum_u \sum_{Y : f_C^Y(g_Y(u)) = g_C(u)} \Pr[Y].$$

Balcan et al. [7] observe that this function is symmetric and obeys the triangle inequality. Therefore the expected distance $E[D_{C'}(Y)] \leq E[D_C(C') + D_C(Y)] = 2E[D_C(Y)]$.

4.4.2 Multiple Samples

Depending on the application, the guarantee of a 3-approximation in expectation may not be sufficient. An unlikely sample could have arbitrarily bad behavior. For example, the probability that our sample is better than a 5-approximation is not guaranteed to be any higher than 1/2. In this subsection we explore various ways to use multiple samples to achieve better approximation guarantees.

Since our approximation guarantee is in expectation, it is important to limit the probability that we choose a bad clustering C' (one where $E[D_{C'}(Y)]$ is much greater than $3E[D_C(Y)]$). We do this using Markov's inequality. Since the approximation ratio is always at least 1 (no solution can be better than the optimal solution),

$$Pr[E[D_{C'}(Y)] > (3+2\epsilon)E[D_C(Y)] \leq 1/(1+\epsilon).$$

We could attempt to bound the variance in the approximation ratio, however as the above example illustrates (or any example where most of the probability mass lies close to the optimal solution, with a small amount of mass at a large distance), it can be quite bad. Through repeated sampling, we can do much better. Instead of taking only a single sample clustering, let us take samples $C'_1, \ldots, C'_m$ from the distribution. The first quantity of interest is the approximation ratio R achieved by

the best of these samples - $\min_i D_{C_i'}(X)$. Since the sampled C_i''s are independent,

$$R = Pr[\min_i D_{C_i'}(X)/E[D_C(X)] > 3 + 2\epsilon]$$

$$\leq \prod_i Pr[D_{C_i'}(X)/E[D_C(Y)] > 3 + 2\epsilon]$$

$$\leq 1/(1+\epsilon)^m.$$

Thus if we want at most a τ probability of having no samples within this distance, we need $m = \lceil \log_{1+\epsilon} 1/\tau \rceil$ or more samples.

The existence of a sample C_i' which is close to a 3-approximation does not directly imply that we can determine which of the sample(s) are good. If our application allows us to test each of the samples and choose one with the best results, we may not need to find the one with the best approximation ratio. Otherwise, to be certain which of the C_i' gives the best approximation ratio, we would need to know C already (or at least be able to calculate $D_C(X)$ knowing X but not C). We can get around this by taking l additional samples $\{X_1, \ldots, X_l\}$, and computing for each of the C_i', the total distance from the X_j's. We then select the C_i' with the minimum such total distance.

We must now consider that the samples $X_1, \ldots, X_l$ that we draw might give a total distance larger than its expectation for the "good" C_i' and smaller than the expectation for a "bad" C_i'. To address this, we now show that when l is sufficiently large, with high probability even if we don't select the best C_i', we will select one which is close enough. For each C_i', the expected cost $E[D_{C_i'}(X)] \geq E[D_C(X)]$, and with probability at least $1 - \tau$, for at least one such C_i', $E[D_{C_i'}(X)] \leq (3 + 2\epsilon) \cdot$

$E[D_C(X)]$. We define $d_i = \sum_{j=1}^{l} D_{C_i'}(X_j)/|U|$. Since the d_i are a sum of independent random variables taking on values in $[0, 1]$, we can apply Chernoff bounds to their deviation probability. If we take $l = (|U|/\delta^2) \cdot O(\log kp)$, then with probability at least $1 - p$ all candidate's distance totals d_i will be estimated to within $(1 \pm \delta)$.

If there exists a candidate with distance total

$$D_{C_i'}(Y) \leq E[D_C(Y)] \cdot (3 + 2\epsilon) \cdot l,$$

and all such estimates are within $(1 \pm \delta)$ of the true totals, then the candidate with the minimum estimated total has a true total at most $\cdot(3+2\epsilon)E[D_C(Y)] \cdot l \cdot (1+2\delta)$. This candidate gives an approximation ratio of $(3 + 2\epsilon) \cdot (1 + 2\delta)$. Such a candidate is found with probability at least $1 - p - \tau$.

4.4.3 Experimental Results

Having a 3-approximation algorithm is a nice theoretical result, but it does not necessarily imply practical benefits. For example, if an optimal solution has a large expected distance (1/3 of the maximum distance for example), then a 3-approximation is meaningless. The hope is that networks will only have such bad behavior if they are inherently not well clusterable. There is some intuitive reason to believe this is so. If a certain set of nodes often forms the majority of a connected component and they are in the same component c of an optimal clustering, then a random clustering Y will likely have a component y that matches with low cost to c. Meanwhile if the optimal clustering has high cost, that means that few large groups of nodes consistently form the bulk of a component.

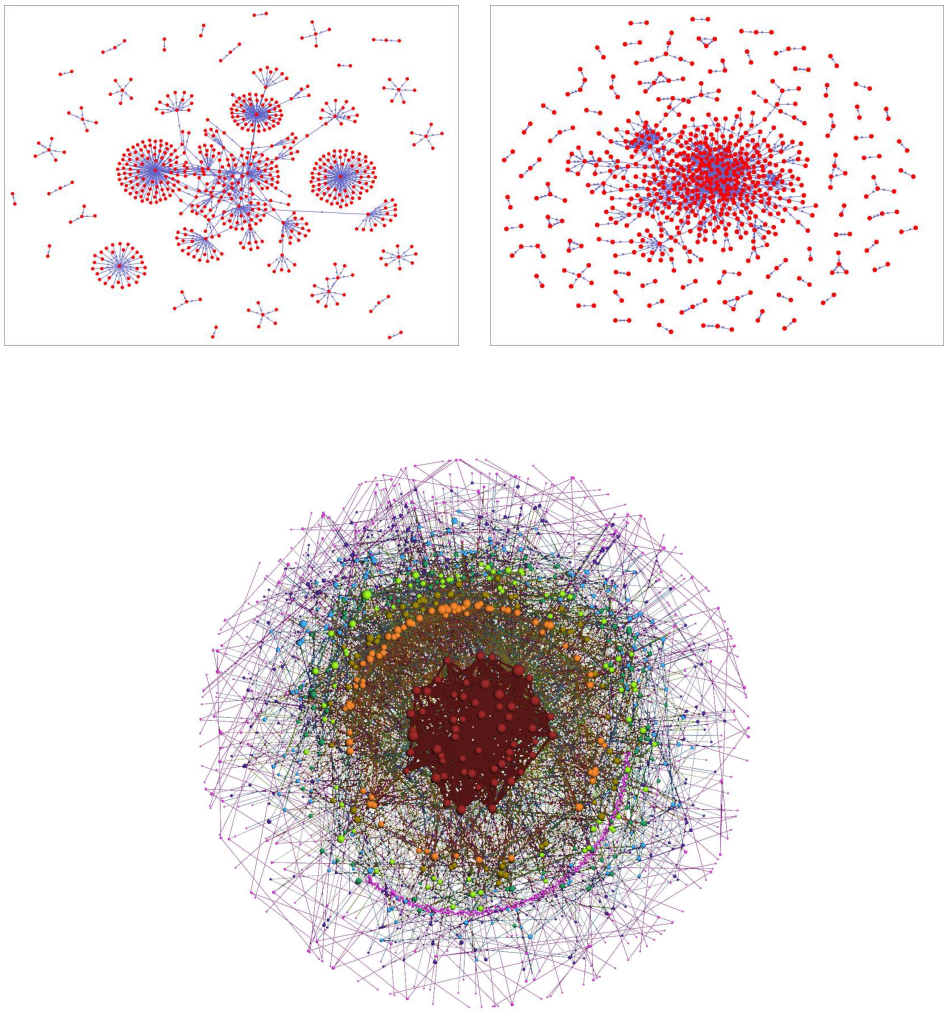


Figure 4.15: The three networks used in our analysis have very different structures. The Trust Project Network (top left) has many star formations which affect the quality of its clusters. The FilmTrust Network (top right) is a more traditionally organized social network. The Epinions Network (bottom) is much larger and harder to succinctly characterize.

In this section we explore what kind of clusterings occur in real trust networks using various parameters. We examine the Trust Project, FilmTrust [45], and Epinions networks. Visualizations of these networks are shown in Figure 4.15 with their sizes shown in Table 4.3. In the first two of these networks, users rate their level of trust (with respect to a specified domain) in their friends. In the Epinions network, users rate whether they like or dislike statements made by others, and these ratings can be used as a proxy for ratings of the statements' authors. In this analysis we address only positive trust, so unfavorable ratings are discarded.

The Trust Project network is derived from an early Semantic Web trust network [46]. As is shown in Figure 4.15, it has many star patterns. This occurs when users make connections to many friends who do not, in turn, participate in the network. Thus, they have no outgoing connections. This affects our ability to cluster the network. The FilmTrust network is built from a social network in which users rate movies and how much they trust their friends in that context. As the visualization shows, it has a more traditional network structure. However, there are a number of small groups that are disconnected from the giant component. These are shown as the small subnetworks, often pairs, floating around the edge of the visualization. Finally, the Epinions network shows social network connections on the product review site. Trust ratings indicate how much they trust one another's reviews.

In all of our networks, ratings form directed trust edges. As a first step we create an undirected and normalized trust graph. We convert every lone directed edge into an undirected edge, and whenever two people rate each other, we average

Network	**# Nodes**	**# Edges**	**Density**
Trust Project	62	105	0.055
FilmTrust	310	774	0.016
Epinions	114,467	717,667	0.0001

Table 4.3: The size and descriptive statistics of our three example networks. Density is calculated as the ratio of edges to possible edges.

their ratings to form a single undirected edge. The edge weights are then normalized to fall between 1 and 10. Since we need probabilities on the edges instead of weights, we introduce a global parameter t. An edge with weight w gets probability $\max(1, w/t)$. Therefore when t is small, edge probabilities tend to be high and connected components will be large, and for large t edge probabilities and connected components will generally be smaller.

For the Trust Project and FilmTrust networks we vary t from 2 to 30, generating 30 sample clusterings for each value. For the Epinions network, we need considerably higher values of t to capture the same behavior, so we use a range of 14 to 50. In Figure 4.16 we show the frequencies of each component size and each component's benefit, where we define the benefit of a cluster to be its size minus its cost (or how much less it costs than its maximum possible cost). Due to our choice of cost function, two clusters each have to share at least half of their nodes to have any benefit at all. The x-axis shows the value of t, the y-axis shows the component size (or benefit), and the circle diameters show the how many components are that

size (or benefit) in our samples. This gives a sense of what size clusters to expect for different values of t. Values of $t < 10$ are included for informational purposes, but may be poor choices in practice, because they give equal weight to all user trust ratings $\geq t$.

Figure 4.17 contains scatter plots of the distances between pairs of randomly sampled clustererings for all 3 datasets. As discussed in Section 4.4.1, the expected distance between two randomly chosen clusterings is at most 3 times optimal. So these plots demonstrate roughly how similar clusterings are, and what range $E[D_C(X)]$ falls into. When $t \to 0$, the random graphs lose their randomness and are always connected. Conversely at $t \to \infty$, the graphs are always disconnected. Therefore at the two extremes distances will be 0. Of interest here is the shape of the curve in between, and specifically for what values of t are there good, representative clusterings.

From Figures 4.16 and 4.17, it is evident that the Trust Project network does not produce particularly stable clusters. Most of the benefits, even for relatively small values of t, are quite small when compared to the larger cluster sizes. This means that there is not much more than 1/2 overlap between matched clusters. Instead most of it's benefit comes from small clusters matching up well. This may be (in part) a product of star shaped connections. Under the right conditions a star graph should form a single cluster. But with our algorithm it will form a random large cluster and many singletons, which will have high distance from another such random clustering.

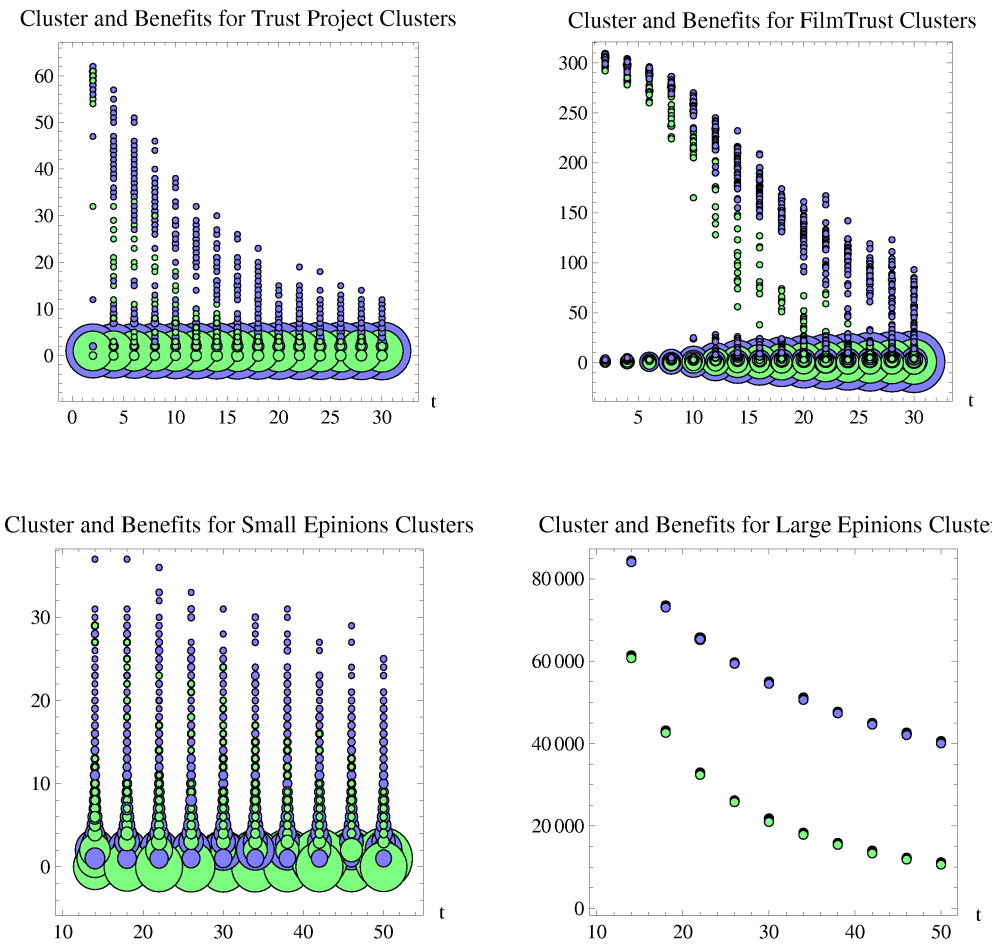


Figure 4.16: The top two plots show component sizes (blue) and benefit sizes (green) within Trust Project (left) and FilmTrust (right) for $t = 2$ to 30 with 30 samples of each. The bottom two plots show the component sizes and benefits for the Epinions dataset, with the left plot showing small clusters and the right plot the largest clusters. A circle centered at (x, y) with radius r indicates that the number of clusters of size (or benefit) y with $t = x$ is proportional to r.

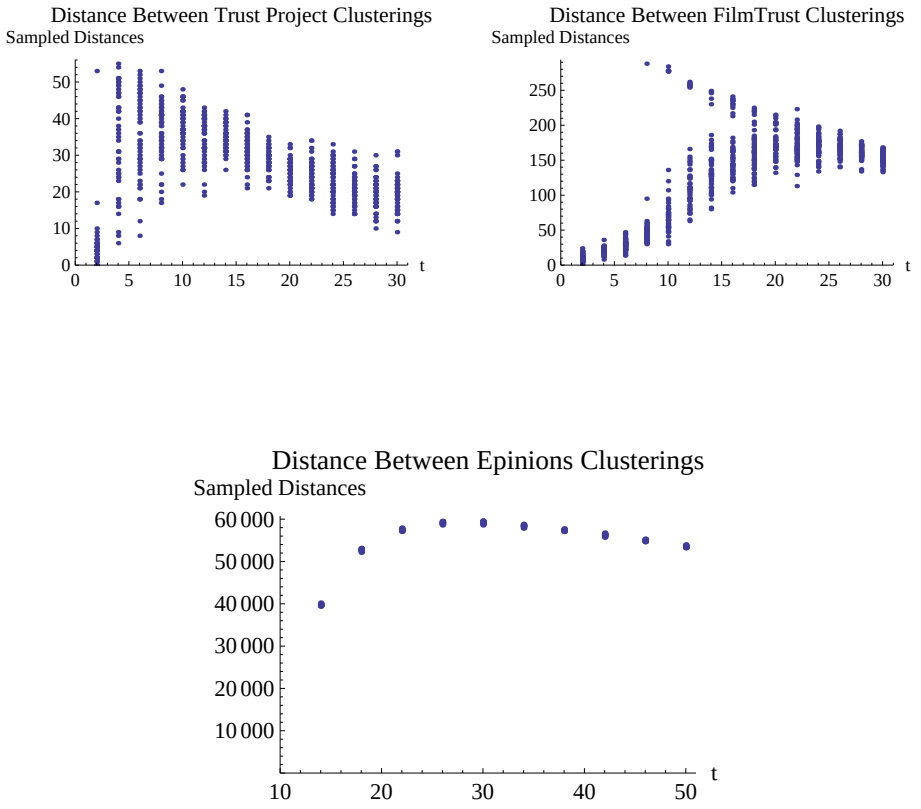


Figure 4.17: This figure shows costs between randomly sampled clusterings for Trust Project (top left), FilmTrust (top right), and Epinions (bottom) networks. The maximum distance between samples in the smaller two networks is approximately the size of the network, whereas the maximum distance in the Epinions network is roughly half of its network size.

The FilmTrust network creates considerably more consistent clusterings. Much more of the benefit comes from a large, fairly consistent cluster, but considerable benefit comes from smaller clusters as well. Even with t as high as 20 (which corresponds to maximum trust giving only a 1/2 edge probability), there are still large clusters that share a considerable core component, indicating a very stable cluster. For this network as well as Trust Project, the shape of the cost curves largely depends on the giant component. If it exists and is stable for a given t, the costs are low. If it exists but changes significantly with different samples, then some pairs have low cost, and some have high cost.

In this Epinions dataset, our algorithm consistently identifies a stable giant component and a number of smaller components of widely varying stability. This clustering behavior is particularly useful for applications that use the trust values to boost performance. Trust values can be used in recommender systems to generate predictive ratings based on a user's social connections [44]. Integrating information about trust clusters into traditional recommender systems can significantly improve the accuracy of recommendations [33]. The smaller groups of size < 30 that identified may reflect the types of niche interest groups that benefit most from the trust clustering recommendation techniques.

Chapter 5

Special-Purpose Software

In order to do empirical studies of random graphs we require appropriate software tools. These tools must be flexible enough to thoroughly explore the data and a wide range of algorithms, and they must efficiently scale up to graphs of hundred thousand nodes or more. For smaller graphs, as well as for visualization and data analysis we use industry-standard platforms such as Mathematica and Matlab. These tools provide all of the flexibility we need, but they suffer from critical scalability issues. In this chapter we discuss the special purpose tools which aid our research. For fast and flexible epidemic simulations we use EpiFast which is developed and hosted by the Network Dynamics and Simulation Science Laboratory which is part of the Virginia Bioinformatics Institute at Virginia Tech. Additionally, as a major component in our work on protective sequestering, we developed a fast and space efficient implementation of our optimal placement algorithm. Finally, we develop a random graph analysis package. This package performs our common tasks in an efficient manner and exports the relevant results for visualization and statistical analysis.

5.1 EpiFast

For our large-scale epidemic simulations, we use EpiFast [14]. EpiFast provides a highly flexible framework for modeling epidemics on an individual level. Once invoked, EpiFast launches parallel processes on a large cluster in order to produce results quickly. It quantizes time into days, so that locations can be sent to different machines for processing without incurring much communication overhead.

As input it takes a contact network in person-location format and a configuration file. The configuration file specifies (among other things), the transmission rate, a distribution on initial infections, how an individual's state changes once they contact the disease, and what interventions to use and when. These factors can be specified on a global level, or specific individuals may be given specific properties. We use some fine-grained control by specifying which individuals should be initially infected, and who receives a vaccination on what day. However the interventions need not be completely determined before running, but can be triggered by the epidemic's progression. For example, mass vaccinations may be triggered by a fixed percentage of the population becoming diagnosed. On an individual level, people can alter their daily routine if too many of their friends fall ill. This heterogeneity is a particularly important feature for accurately modeling the complexity of epidemics in the real-world.

By default, each run of EpiFast (or each iteration within a single run) simulates a unique, random progression of the epidemic. Its output consists of a list of infected individuals, along with the day they became infected, and a who transmitted the

epidemic to them. From this output we construct the vulnerabilities and epidemic trees necessary for our algorithms.

5.2 Sequestering

We implement our optimal partitioning algorithm in C in order to conduct experiments. Figure 3.7 describes the core of the algorithm. In order to handle very large datasets, we chose the memory efficient version which periodically checkpoints rows in the dynamic programming matrix. The program takes a configuration file which specifies the person-to-person transmission rate, the group types, how many groups of each type, and the external infection probabilities of each individual. It outputs, for every group, the people assigned to that group, and the total number of expected infections.

5.2.1 Time and Space Complexity

In this section we analyze our implementation's efficiency. To do so we break it into two parts, one part specific to the subroutine g and concerning mostly initialization, and a second part which concerns the main function Sequester.

In initializing the function g (which gives the expected number of infections if a given set of people are placed into a group as described in Figure 3.8), we compute and store for each group, and every possible number of initially infected people in the group, how many people become finally infected in expectation. If we let m_i denote the size of the i^{th} group when sorted by group sizes, then the permanent

space used to after initialization is $\sum_i m_i$. When computing these values, we need an m_i^3 sized temporary array for the intermediate computation. The j, x, y entry in this array records the probability that there are x people infected by a path of distance at most j, and y people infected at distance exactly j. We achieve a significant efficiency improvement by truncating any array entry whose magnitude is below some threshold. The threshold gives an upper bound on the total error introduced into the final calculation.

Once the initialization is completed, the expected total infection size is the sum for $j = m_i, y = 0$ and over all x of x times the array entry at j, x, y. Each entry in these arrays takes at most m_i time to compute, for a total initialization running time order of the sum over all i from 1 to r of m_i matrices times m_i^3 array entries times m_i time per entry gives a total time complexity of $O(\sum_i m_i^5)$ and a space complexity of order $O(\max_i m_i^3)$.

Instead of computing the stored values exactly we can use a natural Monte-Carlo simulation process to estimate them to within a factor of ϵ as follows. For each ℓ from 1 to k and i from 1 to m_ℓ, generate a random graph $G(m_\ell, p)$. For each simulation j, let the random variable $X_{\ell,i,j}$ be the sum of the sizes of the connected components containing nodes 1 through i – the nodes that become infected. For each ℓ and i, $E[X_{\ell,i,j}]$ is the value we want to compute. Let $X_{\ell,i}$ be the average of our samples j. Using Chernoff bounds, we see that the probability of more than an ϵ relative error after c samples is

$$Pr[|X_{\ell,j} - E[X_{\ell,j}]| \geq \epsilon E[X_{\ell,j}] \leq \exp(-cE[X_{\ell,j}]\epsilon^2/(3m_\ell) \leq \exp(-c\epsilon^2/(3m_\ell)).$$

If we want an error probability of at most $1/n$, we can take $c = \frac{3m_\ell \ln n}{\epsilon^2}$. We can compute the $X_{\ell,i,j}$ values in time m_ℓ^2 using BFS, which gives a total initialization running time of $O(\sum_\ell \frac{3m_\ell^3 \ln n}{\epsilon^2})$ and space of $O(\max_\ell m_\ell^2)$.

Once the initialization is complete, we create an array OPT of size $n \cdot \prod_i u_i$. For each entry in OPT we create and store a matrix A once taking time and space $O(\max_i m_i^2)$ and we make $\sum_i m_i$ calls to g, each of which does $O(m_i)$ arithmetic operations for a total running time of $O(n \cdot (\prod_i u_i \cdot \sum_i m_i^2))$.

In the uniform case where all groups are of size m and $m \cdot u_1 \approx n$ (meaning the total space available is approximately the number of people), this yields a space complexity of $O(n^2/m + m^2)$ and a time complexity of $O(m^5 + n^2 m)$. The naive way of backtracking through the array to produce the partition which gives an optimal result takes no more time than creating the array to begin with, so this is the total complexity of this phase of the algorithm.

We can greatly improve space efficiency with a factor of 2 cost in running time using the following observation: to compute the array after person i, all we need to have stored are the $\max_t m_t$ rows from $OPT(i - \max_t m_t, *)$ to $OPT(i-1, *)$. If we only want to compute the optimal expected outbreak size, we can reuse the space from $OPT(i \mod m + 1, j)$ for every entry $OPT(i, j)$ for a space savings factor of n/m. If we want to compute the optimal partition however, we need to be able to backtrack through the OPT array, which we cannot do efficiently if we have to repeatedly recompute it from scratch for every m people. Instead we have to do something more clever.

Theorem 9 *Suppose we are given an instance of the Sequestering problem with a single room type and total capacity $O(n)$. We can compute an optimal partition in time $O(m^5 + n^2 m)$ and space $O(m^3 + \frac{n^{1.5}}{\sqrt{m}})$.*

The time complexity and initialization space complexity are taken from above. Here we prove the improved space bounds for the main phase. For the single group type case, after computing OPT for c rows (for a value of c to be specified later), we store the entries for the last m of them, and reuse the space for OPT in computing the next c people. This takes space $mu_1 \cdot \frac{n}{c}$ for the saved blocks and $c \cdot u_1$ for the frequently overwritten memory. Combined they sum to $O(\frac{mu_1 n}{c} + cu_1)$ which is minimized when the two terms are equal or $mu_1 n/c = cu_1$ and therefore $c = \sqrt{mn}$. Since $u_1 = O(n/m)$ this gives a total space complexity of $O(n^{1.5}/\sqrt{m})$ plus $O(m^2)$ for the array A. As the algorithm backtracks to find the optimal assignment, each section of c people must be recomputed exactly once from the saved m people before it. This method results in a factor of 2 increase in running time, which is a reasonable trade-off.

5.2.2 Resource Usage

During our experimental evaluation, we generated performance statistics of the algorithm Sequester as implemented in C and running on a reasonably fast Linux machine. We implemented the basic algorithm, thus we compute g exactly using dynamic programming and our space complexity is $O(n^2/m)$ and not $O(n^{1.5}/\sqrt{m})$ as described by Theorem 9. Figure 5.2.2 summarizes our results. We achieve a

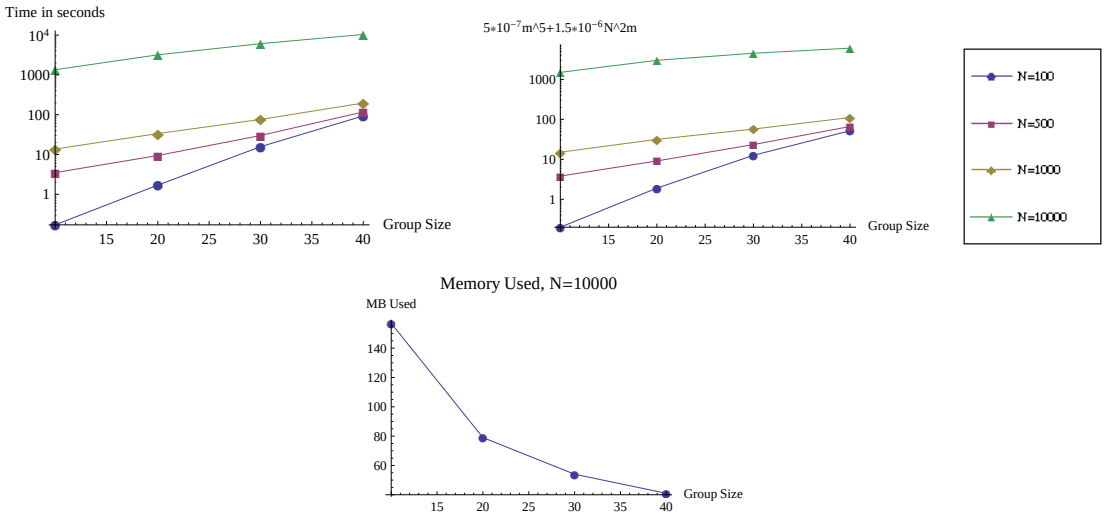


Figure 5.1: Resource usage information for our algorithm's implementation.

running time that closely matches the formula $5 * 10^{-7}m^5 + 1.5 * 10^{-6}n^2m$ seconds. Roughly this means problem sizes like $n = 100, m = 20$ complete nearly instantly and our largest example of $n = 10,000$ and $m = 40$ taking almost three hours. With regard to memory, the program uses roughly $16n^2/m$ bytes plus 3MB overhead. For the majority of our runs, the overhead dominates the algorithm's memory usage, so we only show detailed results for when $n = 10000$. On a 32-bit machine, this nearly reaches the limit of what the basic algorithm can fit into memory. However at $n = 10000$ and $m = 10$, we would expect roughly a factor of $\sqrt{n/m} \approx 30$ space savings using the space efficient version.

5.3 Graph Analysis Package

Since much of our work deals with s, t connectivity probabilities and other properties of large random graphs, we require a tool which can manipulate them as

needed on a commodity personal computer. For smaller graphs, Mathematica and Matlab perform excellently. They provide the flexibility to carry out any calculation or algorithm plus the visualization and statistical tools necessary for fully exploring the results. Being general-purpose means they are not optimized for working with large graphs in terms of time or space, and they have trouble scaling up above tens of thousands of nodes. After exploring some of the existing graph analysis packages, we determined that making our own was simpler than finding, learning, and customizing an existing one. In this section we explain our package as well as some of the trade-offs that went into its design.

Our graph analysis package operates as a persistent command loop. We first made that decision because graph files hundreds of megabytes long take a significant amount of time to load. A command loop design allows us to load a graph once and operate on it many times. Using a command loop we can also string operations together in an arbitrary order, including a reset back to the original graph, plus we can execute scripts. While our command language is very simple, we can execute Turing complete operations by having a scripting language wrapper which pipes commands to the loop and parses its outputs on the fly.

Because social networks are sparse graphs, and we want common graph operations like connected component decomposition and iterating over all edges to be fast, we use an adjacency list graph format. For maximally efficient node accesses, node ids must be consecutive integers starting with 0. Any graph not in this format can be pre-precessed once and subsequently loaded quickly. When loading a graph into memory, we require it have one edge per line, with each edge consisting of two

node ids and an edge weight. The weights can be arbitrary doubles as the loop keeps a global "scale factor" variable to map the range onto $[0, 1]$.

We maintain one working graph at a time, which is freed only when a new graph is loaded or the program exits. However this graph is not static. We can augment the graph by adding new edges from a file. These new edges remain part of the graph until it is explicitly reset. We can also generate a random instance of the graph where each edge's probability of being kept is its weight times the scale factor. The removed edges are not freed, but saved so that they can be returned when we generate a new random instance or reset the graph. We can also compute a max $s - t$ flow (either on the full graph or a random instance), in which case the resulting graph contains both the flow and residual graph information. Reseting the graph, generating a random instance, and adding edges all take linear time. We use the push-relabel method [51] to find a maximum flow where each vertex with excess flow tries to send it closer to the sink. This method has worst case complexity of $O(V^2 \cdot E)$, however appears to run significantly faster in practice.

Our next set of functionality directly addresses connectedness and pairwise connectivity probabilities. We use depth first search (DFS) for finding the connected components in linear-time. For estimating probabilities, we have a command which generates many independent random samples and finds the connected components in each of them. For each sample we record every node's component id. These records are kept active and can be queried until we reset the graph or create a new set of random samples. To estimate the connectivity between two nodes we compare their component ids and return the fraction of samples where they are in the same

component. The samples can be printed directly, or we can print a pairwise path probability or all path probabilities from a given node. In order to quickly visualize the variation in path probabilities, we also support quickly giving histogram-like information for the distances from a specific node. It takes $O(k \cdot (V + E))$ time to generate k samples and their connected component records and $O(k)$ time per path probability query.

The functionality described so far handles positive edges only which is sufficient for our work in Section 4.1 but not for Section 4.2. We implement one feature to return the fraction of negative edges between the same connected component, but our main work for mixing positive and negative edges is the spring embedding algorithm. We use this algorithm to find a global k-dimensional embedding of nodes in the graph where the endpoints of positive edges are biased towards each other and those of negative edges are biased apart. This type of algorithm takes as input a network, the force functions for positive and negative edges, a dimension for the embedding, an a termination condition. After simulating many iterative updates of the system state, it outputs the geometric location of every node.

The spring embedding parameters are all straightforward except the force functions. The attractive force between the endpoints of a positive edge should increase with distance while a negative edge's repulsive force should decrease. There are no other rules, so we leave a great deal of flexibility. We do not want too much flexibility though, since we compute these forces once per edge on every iteration and need to do so quickly. Even interpreting a simple mathematical string would be too inefficient. We choose instead to allow four parameters to completely define

the forces. The positive force functions take the form $a \cdot d^b$ where d is the distance between nodes, a is a real-valued parameter, and b is one, two, or three. The negative force function takes the form $a' \cdot d^{-b'}$ where a' and b' have the same constraints as a and b. This allows us to try out different combinations of functions at runtime.

When a user invokes the spring embedding command for dimension k, we start by initializing a $|V|$ by k array so that each vertex has a uniformly random point in $[0, 1]^k$. We also initialize a variable Δ which is the timestep simulated by the current iteration. The higher the value of Δ, the further a given force will push a node in that iteration. We then calculate every edge's force on its endpoints using the parameterized force functions. Nodes which are closer together than some threshold have their forces decreased to prevent arbitrarily large forces. Once every vertex has its net force calculated, we multiply that force by Δ, divide the force by the square root of the vertice's degree, and move the vertex. We return any vertex pushed outside of the $[0, 1]^k$ boundary back within the valid space. At each iteration we sum the force magnitudes over all of the vertices, and we think of this as the embedding's "balance".

Our algorithm terminates when it reaches either a parameterized maximum number of iterations, or a small enough sum of forces on the vertices. We added the second threshold because the smaller the net forces, the more balanced the embedding. If the network is perfectly balanced, there are no net forces and the network stabilizes. Borrowing an idea from simulated annealing [97], we also use the current balance to adjust Δ. If the network moves towards greater balance, we assume it may be converging towards an equilibrium that we do not want to

overshoot. In this case we decrease Δ slightly. On the other hand, if the network moves away from balance, we assume that it is far from equilibrium and would benefit by changing faster, so we increase Δ.

Once we compute the embedding, it remains in memory until the graph is reset of a new embedding is computed. We provide commands to dump the embedding to a file, or query the distance between a pair of points. Additionally, we support batch processing of all of the graph's edges. One command will output to a file one line per edge showing the ids of its endpoints, its sign and magnitude, the estimated path probability between its endpoints, and their embedded distance. We designed this last feature specifically to test our methods in the edge sign prediction problem.

Chapter 6

Conclusion

Even long before the first computers, social connections have profoundly affected everyday life from our areas of disease transmission and extending trust to many others like economic activity, justice, and politics. Recently, they have become both more important (due to expanded communication networks and the ability to travel to a different city or country quickly and easily) and easier to capture (online networks are recorded exactly and population networks can be approximated). In addition, ever increasing computational speed and storage allows us to study large social networks in detail mathematically. Our work focuses on one particularly important aspect of these networks – the affects of numerous, interdependent paths of connectivity within these networks. Understanding how to use and manipulate the collective strength of these paths leads us to interesting algorithmic solutions to important problems. Within this space we focus on two specific applications: epidemic mitigation and social trust inference.

In the area of epidemic mitigation we develop two new algorithms. For protecting the general population we introduce the heuristic to target vaccinations based upon individuals' epidemic subtree sizes. We derive this heuristic from a novel graph-theoretic proof about vertex cuts in a random graph. We show empirically that this method performs similarly to two other heuristics for minimizing total

infections on a particular, realistic contact network. We conclude by arguing why that network's properties cause our results, and what to look for in a network which would produce different results.

Our second epidemic mitigation algorithm protects a small but important subpopulation from an at-large epidemic by removing them from the general population and carefully isolating them into groups, usually of some maximum size. We model the structure of the resulting groups as a random graph and show through a probabilistic argument how to optimally partition people into these groups. We show that in a realistic setting an optimal solution can lead to considerably fewer final infections than a naive one. Because protective sequestering involves much more than how to place individuals into isolated groups, we use our placement algorithm as a basis for exploring the other trade-offs involved and how sensitive the infections rates are to real-world problems like estimation error and logistics.

Both of these sides of epidemic mitigation have significant open problems to consider. Some major directions for further work include:

- What properties do much larger (state or country-wide) contact networks have, and how do those properties effect various vaccination strategies?

- More information about an epidemic arises as it runs its course – taking some transmission paths by not others. Can we use this information as it becomes available to make better decision about how to counter the epidemic?

- If the vaccination strategy is public, could some individuals alter their behavior in a way that makes the epidemic worse in order to increase their likelihood of receiving a vaccination?

- A simulated population may resemble a real population, but there is no mapping from individuals in one to the other. Can we use a targeted vaccination strategy in a simulated network to aid in targeted vaccinations in a similar real one?

- In a real protective sequestering scenario we cannot perfectly isolate a large number of individuals. What types of connections exist, and how do they effect the results?

We also bring a graph-theoretic approach to the problem of social network trust inference. We develop several important algorithms: using path probabilities in a random-graph as an indicator for inferred trust, using spring embedding to infer trust based upon both trust and distrust information, and linear-time trust clustering.

Our first contribution – the idea to relate trust in a social network to edge and path probabilities in a random-graph – directly brings graph theory to a practical problem. Because of the mathematical basis of our approach, we immediately realize a number of benefits, including a trust metric space for clustering or other applications. We explore the parameter space of our algorithm, and show it to be fairly robust to the specific trust to probability mapping, though reasoning analytically about a given network shows how to tune the function to get the most information

out of the results. A wide range of dataset sizes motivates our development of several variants of the algorithm which trade-off space and running-time in one phase or another. Experimental results on real trust networks provides validation for our approach.

We proceed by addressing how distrust fits into our social trust framework. Because there is no distrust analog for our Bayesian chain view of trust paths, we require a new approach – inferred trust as the results of a spring embedding process. Our spring embedding algorithm does not have the nice graph-theoretic structure as the path probability approach, but we find that it adds important information on top of the algorithms which use positive trust only. As usual, we validate the combined trust inference algorithm empirically, using it for hidden edge sign prediction and cluster reconstruction.

Finally we show that when clustering is the goal of trust inference, a simple and very fast algorithm can bypass the computation of trust distances and find a clustering directly. Our main contribution here is not the algorithm itself but the analysis behind it. We show that a sample clustering from a fixed distribution gives an approximately central clustering for that distribution under a particular distance metric on clusterings. As is often the case, this algorithm is meaningful on some networks but not on others. We examine its behavior on several real datasets and discuss when it will likely give useful results.

Within trust inference, major open problems include:

- Can we quantify and adjust our algorithms for the non-uniformilty in how users rate each other?

- The new edge prediction problem: how well can a trust inference algorithm predict which edges will be added to a social network and which edges will change their sign/magnitude?

- Do social networks have "ground truth" clusterings (or overlapping clusterings) and can we detect them using trust-based methods?

- Can organizations make use of social trust clustering to form teams with high cohesion and productivity?

- Alternatively, can organizations or users cultivate new trust links carefully to maximize their effect on the trust network?

In all of the above cases, when working on complex network problems we recognize the importance of developing both a deep, mathematical understanding of the structures involved and also empirical results which demonstrate how they apply in a realistic setting. This interplay between theory and data-driven results goes well beyond the obvious goal of demonstrating an algorithm's effectiveness. We also work to understand when and why an algorithm performs better than its guarantees, and how to use knowledge of the structure of a particular dataset to further refine the algorithm: such as choosing a vaccination strategy based upon how embedded a contact network is or tuning the trust to edge probability function to maximize the information content of connectivity probabilities.

Bibliography

[1] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *Journal of the ACM*, 55(5):1–27, 2008.

[2] N. Alon, I. Benjamini, and A. Stacey. Percolation on finite graphs and isoperimetric inequalities. In *Annals of Probability*, volume 32, pages 1727–1745, July 2004.

[3] Aaron Archer. Two $O(\log k)$-approximation algorithms for the asymmetric k-center problem. In *Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization*, pages 1–14. Springer-Verlag, 2001.

[4] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *J. ACM*, 45:70–122, January 1998.

[5] Paolo Avesani, Paolo Massa, and Roberto Tiella. A trust-enhanced recommender system application: Moleskiing. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1589–1593, New York, NY, USA, 2005. ACM.

[6] Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 19(3):357–367, 1967.

[7] Maria F. Balcan, Avrim Blum, and Anupam Gupta. Approximate clustering without the approximation. In *SODA '09: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1068–1077, Philadelphia, PA, USA, 2009. Society for Industrial and Applied Mathematics.

[8] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *Machine Learning*, pages 238–247, 2002.

[9] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A*, 311:3, 2002.

[10] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999.

[11] Albert-László Barabási, Réka Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the World-Wide Web. *Physica A*, 281:69–77(9), 2000.

[12] M. Barthelemy, A. Barrat, R. Pastorsatorras, and A. Vespignani. Dynamical patterns of epidemic outbreaks in complex heterogeneous networks. *Journal of Theoretical Biology*, 235(2):275–288, July 2005.

[13] David M Bell, World Health Organization Working Group on International, and Community Transmission of SARS. Public health interventions and sars spread, 2003. *Emerg Infect Dis*, 10(11):1900–1906, Nov 2004.

[14] Keith R. Bisset, Jiangzhuo Chen, Xizhou Feng, V.S. Anil Kumar, and Madhav V. Marathe. Epifast: a fast algorithm for large scale realistic epidemic simulations on distributed memory systems. In *ICS '09: Proceedings of the 23rd international conference on Supercomputing*, pages 430–439, New York, NY, USA, 2009. ACM.

[15] R Beckman K Bisset J Chen T DuBois S Eubank B Lewis A Kumar M Marathe A Srinivasan P Stretz C Barrett. Protective sequestering of socially essential subpopulations. Technical report, Virginia Tech, 2010.

[16] Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In Matteo Fischetti and David Williamson, editors, *Integer Programming and Combinatorial Optimization*, volume 4513 of *Lecture Notes in Computer Science*, pages 182–196. Springer Berlin / Heidelberg, 2007.

[17] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Network robustness and fragility: Percolation on random graphs. *Physical Review Letters*, 85:5468, 2000.

[18] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Math. Stat.*, 23:493–509, 1952.

[19] F. Chung. *Spectral Graph Theory*. AMS Publications, 1997.

[20] F. Chung and L. Lu. *Complex Graphs and Networks*. AMS Publications, 2006.

[21] Fan Chung, Paul Horn, and Linyuan Lu. The giant component in a random subgraph of a given graph. In *WAW '09: Proceedings of the 6th International Workshop on Algorithms and Models for the Web-Graph*, pages 38–49, Berlin, Heidelberg, 2009. Springer-Verlag.

[22] Fan Chung and Linyuan Lu. The average distance in random graphs with given expected degrees. In *Proceedings of National Academy of Science*, volume 99, pages 15879–15882, 2002.

[23] Fan Chung and Linyuan Lu. Connected components in random graphs with given expected degree sequences. *Annals of Combinatorics*, 6:125–145, November 2002.

[24] Fan Chung and Linyuan Lu. The volume of the giant component for a random graph with given expected degrees. *SIAM J. Discrete Math.*, 20(2):395–411, 2006.

[25] Fan Chung, Linyuan Lu, and Van Vu. The spectra of random graphs with given expected degrees. In *Proceedings of National Academy of Science*, volume 100, pages 6313–6318, 2003.

[26] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Rev.*, 51(4):661–703, 2009.

[27] J. Cohen and M. Enserink. After delays, WHO agrees: the 2009 pandemic has begun. *Science*, 324:1496–1497, June 2009.

[28] Reuven Cohen, Keren Erez, Daniel Ben-Avraham, and Shlomo Havlin. Resilience of the Internet to Random Breakdowns. *Physical Review Letters*, 85(21):4626–4628, November 2000.

[29] Joshua N. Cooper and Lincoln Lu. Where do power laws come from?, Feb 2007.

[30] Sanjoy Dasgupta. Learning mixtures of gaussians. In *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 634, Washington, DC, USA, 1999. IEEE Computer Society.

[31] Dimitri do B. DeFigueiredo and Earl T. Barr. Trustdavis: A non-exploitable online reputation system. In *Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*, pages 274–283, Washington, DC, USA, 2005. IEEE Computer Society.

[32] T. DuBois, J. Golbeck, and A. Srinivasan. Rigorous probabilistic trust-inference with applications to clustering. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 655–658. IEEE Computer Society, 2009.

[33] Thomas DuBois, Jennifer Golbeck, John Kleint, and Aravind Srinivasan. Improving recommendation accuracy by clustering social neworks with trust. In *Proceedings of the ACM RecSys 2009 Workshop on Recommender Systems and the Social Web*, October 2009.

[34] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[35] K. Earhart, C. Beadle, L. Miller, M. Pruss, G. Gray, E. Ledbetter, and M. Wallace. Outbreak of influenza in highly vaccinated crew of U.S. Navy ship. *Emerging Infectious Diseases*, 7:463–465, 2001.

[36] Erdös and Rényi. On random graphs. *Publ. Math. Debrecen*, pages 290–297, 1959.

[37] P. Erdös and A. Rényi. On the evolution of random graphs. In *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[38] Stephen Eubank, Hasan Guclu, V. S. Anil Kumar, Madhav V. Marathe, Aravind Srinivasan, Zoltan Toroczkai, and Nan Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429(6988):180–184, May 2004.

[39] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '99, pages 251–262, New York, NY, USA, 1999. ACM.

[40] Neil M. Ferguson, Derek A. T. Cummings, Christophe Fraser, James C. Cajka, Philip C. Cooley, and Donald S. Burke. Strategies for mitigating an influenza pandemic. *Nature*, 442(7101):448–452, April 2006.

[41] C. M. Fortuin, P. W. Kasteleyn, and J. Ginibre. Correlation inequalities on some partially ordered sets. *Comm. Math. Phys*, 22:89–103, 1971.

[42] Alban Galland, Serge Abiteboul, Amélie Marian, and Pierre Senellart. Corroborating information from disagreeing views. In *WSDM '10: Proceedings of the third ACM international conference on Web search and data mining*, pages 131–140, New York, NY, USA, 2010. ACM.

[43] Timothy C. Germann, Kai Kadau, Ira M. Longini, and Catherine A. Macken. Mitigation strategies for pandemic influenza in the united states. *PNAS*, 103(15):5935–5940, April 2006.

[44] J. Golbeck. Generating predictive movie recommendations from trust in social networks. *Trust Management*, pages 93–104, 2006.

[45] J. Golbeck and J. Hendler. Filmtrust: movie recommendations using trust in web-based social networks. In *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, volume 1, pages 282–286, 2006.

[46] Jennifer Golbeck. *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, University of Maryland, College Park, MD, April 2005.

[47] Jennifer Golbeck. The dynamics of web-based social networks: Membership, relationships, and change. *First Monday*, 12(11), 2007.

[48] Jennifer Golbeck and James Hendler. Accuracy of metrics for inferring trust and reputation in semantic web-based social networks. In *Engineering Knowledge in the Age of the SemanticWeb*, volume 3257 of *Lecture Notes in Computer Science*, pages 116–131. Springer Berlin, 2004.

[49] Jennifer Golbeck and James Hendler. Reputation network analysis for email filtering. In *Proceedings of the First Conference on Email and Anti-Spam*, 2004.

[50] Jennifer Golbeck and James Hendler. Inferring binary trust relationships in web-based social networks. *ACM Trans. Internet Technol.*, 6(4):497–529, 2006.

[51] Andrew V. Goldberg and Robert E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35:921–940, October 1988.

[52] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*, pages 403–412. ACM, 2004.

[53] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[54] Ara Hayrapetyan, David Kempe, Martin Pl, and Zoya Svitkina. Unbalanced graph cuts. In Gerth Brodal and Stefano Leonardi, editors, *Algorithms ESA 2005*, volume 3669 of *Lecture Notes in Computer Science*, pages 191–202. Springer Berlin / Heidelberg, 2005.

[55] F. Heider. Attitudes and cognitive organization. *Journal of Psychology*, 21(2):107–112, 1946.

[56] Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, May 1985.

[57] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *American Statistical Association*, 58(301):13–30, Mar 1963.

[58] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc. (N.S*, 43:439–561, 2006.

[59] Chaug-Ing Hsu and Hsien-Hung Shih. Transmission and control of an emerging influenza pandemic in a small-world airline network. *Accident Analysis & Prevention*, July 2009.

[60] Tara LaForce James M Hyman. *Bioterrorism: mathematical modeling applications in homeland security*, chapter 10, Modeling the spread of influenza among cities, pages 211–234. Society for Industrial Mathematics, 2003.

[61] Svante Janson. New versions of Suen's correlation inequality. In *proceedings of the eighth international conference on Random structures and algorithms*, pages 467–483, New York, NY, USA, 1998. John Wiley & Sons, Inc.

[62] Audun Jøsang, Elizabeth Gray, and Michael Kinateder. Analysing Topologies of Transitive Trust. In Theo Dimitrakos and Fabio Martinelli, editors, *Proceedings of the First International Workshop on Formal Aspects in Security & Trust (FAST2003)*, pages 9–22, Pisa, Italy, September 2003.

[63] Sepandar D. Kamvar, Mario T. Schlosser, and Hector G. Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651, New York, NY, USA, 2003. ACM.

[64] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. A local search approximation algorithm for k-means clustering. In *SCG '02: Proceedings of the eighteenth annual symposium on Computational geometry*, pages 10–18, New York, NY, USA, 2002. ACM.

[65] W. O. Kermack and A. G. McKendrick. A Contribution to the Mathematical Theory of Epidemics. *Proceedings of the Royal Society of London. Series A*, 115(772):700–721, 1927.

[66] S. Kilic and G. Gray. Nonpharmaceutical interventions for military populations during pandemic influenza. *TAF Prev Med Bulletin*, 6(4):285–290, 2007.

[67] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *in Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–170, 2000.

[68] Jon M. Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew S. Tomkins. The web as a graph: Measurements, models, and methods. In T. Asano, editor, *International Conference on Combinatorics and Computing*, Lecture Notes in Computer Science, pages 1–17. Springer Berlin / Heidelberg, 1999.

[69] Ravi Kumar, Jasmine Novak, and Andrew Tomkins. Structure and evolution of online social networks. In Philip S. S. Yu, Jiawei Han, and Christos Faloutsos, editors, *Link Mining: Models, Algorithms, and Applications*, pages 337–357. Springer New York, 2010.

[70] Ugur Kuter and Jennifer Golbeck. Sunny: a new algorithm for trust inference in social networks using probabilistic confidence models. In *AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence*, pages 1377–1382. AAAI Press, 2007.

[71] Ugur Kuter and Jennifer Golbeck. Semantic web service composition in social environments. In *8th International Semantic Web Conference (ISWC2009)*, October 2009.

[72] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *Proceedings of the 19th international conference on World wide web*, pages 641–650. ACM, 2010.

[73] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 1361–1370. ACM, 2010.

[74] J. Leskovec, K.J. Lang, A. Dasgupta, and M.W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[75] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *J. Mach. Learn. Res.*, 11:985–1042, March 2010.

[76] Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 695–704, New York, NY, USA, 2008. ACM.

[77] Raph Levien and Alex Aiken. Attack-resistant trust metrics for public key certification. In *7th USENIX Security Symposium*, pages 229–242, 1998.

[78] Raphael L. Levien. *Attack Resistant Trust Metrics*. PhD thesis, University of California at Berkeley, 2002.

[79] G Macdonald. *The epidemiology and control of malaria*. Oxford University Press, 1957.

[80] Y. Malevergne, V. Pisarenko, and D. Sornette. Empirical distributions of stock returns: between the stretched exponential and the power law? *Quantitative Finance*, 5(4):379–401, 2005.

[81] S. Martin, R. D. Carr, and J.-L. Faulon. Random removal of edges from scale free graphs. *Physica A Statistical Mechanics and its Applications*, 371:870–876, November 2006.

[82] P. Massa and B. Bhattacharjee. Using trust in recommender systems: an experimental analysis. In *Proc. of 2nd Int. Conference on Trust Management*, 2004.

[83] Jan Medlock and Alison P Galvani. Optimizing influenza vaccine distribution. *Science*, 325(5948):1705–1708, Sep 2009.

[84] L A Meyers. Contact network epidemiology: bond percolation applied to infectious disease prediction and control. *Bulletin of the American Mathematical Society*, 44:63–86, 2007.

[85] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 29–42, New York, NY, USA, 2007. ACM.

[86] M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet Mathematics*, 1(2):226–251, 2004.

[87] Cristopher Moore and M. E. J. Newman. Epidemics and percolation in small-world networks. *Physical Review E*, 61(5):5678–5682, May 2000.

[88] J. Müller. Optimal vaccination strategies–for whom? *Math Biosci*, 139(2):133–154, Jan 1997.

[89] Diana C. Mutz. Social Trust and E-Commerce: Experimental Evidence for the Effects of Social Trust on Individuals' Economic Behavior. *Public Opin Q*, 69(3):393–416, 2005.

[90] M. Newman. The spread of epidemic disease on networks. *Physical Review E*, 66:016128, 2002.

[91] M. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[92] J. O'Donovan and B. Smyth. Trust in recommender systems. In *Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174. ACM, 2005.

[93] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[94] J. Scott Provan. The complexity of reliability computations in planar and acyclic graphs. *SIAM Journal on Computing*, 15(3):694–702, 1986.

[95] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. *The SemanticWeb-ISWC 2003*, pages 351–368, 2003.

[96] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the Fifth International Conference on Computer and Information Technology*, 2002.

[97] Christopher C. Skiścim and Bruce L. Golden. Optimization by simulated annealing: A preliminary computational study for the tsp. In *Proceedings of the 15th conference on Winter Simulation*, volume 2 of *WSC '83*, pages 523–535, Piscataway, NJ, USA, 1983. IEEE Press.

[98] A. Srinivasan, S. Banerjee, S. Lee, and B. Bhattacharjee. Resilient multicast using overlays. In *IEEE/ACM Transactions on Networking*, pages 237–248, 2006.

[99] Matthew W Tanner, Lisa Sattenspiel, and Lewis Ntaimo. Finding optimal vaccination strategies under parameter uncertainty using stochastic programming. *Math Biosci*, 215(2):144–151, Oct 2008.

[100] Virginia Tech. Factbook: Student overview, 2010-2011.

[101] Andreas Wagner. How the global structure of protein interaction networks evolves. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270(1514):457–466, 2003.

[102] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.

[103] C. N. Ziegler and G. Lausen. Spreading activation models for trust propagation. In *e-Technology, e-Commerce and e-Service, 2004. EEE '04. 2004 IEEE International Conference on*, pages 83–97, 2004.

[104] Cai-Nicolas Ziegler and Georg Lausen. Analyzing correlation between trust and user similarity in online communities. *: Trust Management*, pages 251–265, 2004.

[105] Cai-Nicolas Ziegler and Georg Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4-5):337–358, December 2005.