MEASURING AND MODELING THE WEB

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Ying Xu
July 2008

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Rajeev Motwani) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Ashish Goel)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Amin Saberi)

Approved for the University Committee on Graduate Studies.

Abstract

The last couple of decades have witnessed a phenomenal growth in the World Wide Web. The Web has now become a ubiquitous channel for information sharing and dissemination. This has created a whole new set of research challenges. This thesis describes several research contributions in an endeavor towards a better understanding of the Web. We focus on two major topics: (1) measuring the size of the Web and indexable web; (2) modeling the Web and social networks using webgraph models.

In the first part of this thesis, we address the problem of estimating the sizes of the Web and indexable web. This problem has drawn keen research interests in recent years. The term "Web size" refers to the number of webpages on the Web. However, part of the Web is rarely accessed or has very low quality, so often researchers are more interested in a quantity called "indexable web" size. Indexable web refers to the subset of webpages that are indexed by major search engines, which defines an important subset of the Web that is most easily accessible to human users. The index size of a search engine is also an important indicator of search engine performance. In recent years there has been much speculation about the sizes of the indexes of major search engines with Google, Yahoo and MSN Search vying for top spot. The question of which search engine provides the most coverage is thus of public interest and there is a need for objective evaluation methods.

We propose methods for estimating the Web and indexable web sizes. We develop the first methods for estimating absolute index sizes of search engines, assuming only access to their public query interface. We validate our methods with synthetic data sets, and then apply them to estimate index sizes for major search engines. After presenting the empirical results, we then study the problem from a different angle:

we map it to a classic theoretical problem of sum estimation, and propose near optimal algorithms by proving almost matching lower and upper bounds for this problem.

After measuring the size of the Web, we are further interested in understanding its structure. The Web can be viewed as a graph where webpages are vertices and hyperlinks between pages are edges; we call it *Webgraph*. The Webgraph is so large that even estimating its size is a hard problem, not to mention more complicated measurement and analysis. To better understand the Webgraph, researchers leverage a powerful theoretical tool called webgraph models, which are stochastic processes that generate random graphs that, with high probability, behave similar to the Webgraph.

In the second half of the thesis, we present several results in analyzing webgraph models and applying the models to real world applications.

First, we study searchability in random graphs (the property that a decentralized routing algorithm can find short paths in the graph): we give a characterization of random graphs that are searchable with deterministic memoryless searching algorithms, and based on this characterization we prove a monotonicity result.

Next, we study a recently proposed webgraph model called Stochastic Kronecker Graph model. We analyze graph properties of this model: we give necessary and sufficient conditions for Kronecker graphs to be connected or to have giant components; we prove that under the parameters that the graph is connected with high probability, it also has a constant diameter with high probability; we also show that Kronecker graphs are not "searchable" even with randomized routing algorithms.

Finally we study link privacy in social networks using webgraph models. Social networks constantly face the dilemma of providing utility and protecting privacy. we formalize a particular attack on link privacy where the adversary breaks in multiple user accounts and stitches together local network information of different users in order to gain global information about the social graph. We quantify how social networks' access control policy affects the complexity of attack under Power Law Random Graph model, and confirm the analytical results with simulation on real social networks.

Acknowledgements

I would like to express my foremost gratitude to my advisor, Rajeev Motwani, for his invaluable guidance and support throughout my Ph.D. years. I want to thank him for giving me freedom to pursue research direction that I am interested in, and at the same time being helpful and supportive whenever I need advice. He has excellent insight at suggesting research problems that are well-motivated and also theoretically challenging.

I owe my sincere thanks to Professor Ashish Goel, Amin Saberi, and students and speakers participating the RAIN seminar (Research on Algorithms for the Internet). A large part of work in this thesis is inspired by presentations and discussion in RAIN. I would like to especially thank Ashish Goel and Amin Saberi for taking the effort to read my thesis.

I want to thank peer students in Rajeev's group, and other professors and students in theory group and infolab. The numerous group meetings and seminars in those group create a wonderful research environment for my Ph.D. study.

I am grateful to my colleagues during my summer internships, especially my mentors: Ravi Kumar from Yahoo! Research, Rakesh Agrawal from Microsoft Search Labs, and Venkatesh Ganti from Microsoft Research. Every internship is a fun and refreshing experience.

Finally I thank my family for giving me unconditional love, support, and encouragement all the time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The last couple of decades have witnessed a phenomenal growth in the World Wide Web. The Web has now become a ubiquitous channel for information sharing and dissemination. This has created a whole new set of research challenges. This thesis describes several research contributions in an endeavor towards a better understanding of the Web.

One of the first questions scientists would like to measure about the Web is its size, i.e. how many webpages there are on the World Wide Web. A closely related problem is the sizes of search engine indexes, as search engine index defines an important subset of the Web that is most easily accessible to human users. Those two problems are also very similar technically. In the last ten years both the scientific literature and the popular press dealt at length with methodologies and estimates for the size of the various public web search engines and the size of the Web. In the first part of the thesis, we propose methods for estimating the Web size; we present both empirical and theoretical results.

After measuring the size of the Web, we are further interested in understanding its structure. The Web can be viewed as a massive graph where webpages are vertices and hyperlinks between pages are edges; we call this graph "Webgraph". The Webgraph is so large that even estimating its size is a hard problem, not to mention more complicated measurement and analysis. To better understand the Webgraph, researchers study random webgraph models, hoping to capture the essence of the

massive Web graph. A random graph model is a stochastic process that generates random graphs, i.e., it defines a probability distribution on graphs. Webgraph models are those random graph models that, with high probability, generate graphs similar to the Webgraph. Webgraph models are extremely useful tools to develop a deep understanding of the evolutionary scheme and important characteristic of the Web, to predict future development and behaviors, and to generate synthetic data sets for testing and simulation. In the second half of the thesis, we present several results in analyzing webgraph models and applying the models to real world applications.

1.1 Estimating the Web Size

The problem of estimating the sizes of the Web and search engine indexes has drawn keen research interests in the recent ten years (see for example [LG98, LG00, GS05, BFJ+06, BYG07]).

We use the term "Web size" to refer to the number of webpages on the Web. But part of the Web is rarely accessed or has very low quality, so sometimes researchers are more interested in a quantity called "indexable web" size. Indexable web refers to the subset of webpages that are indexed by major search engines [Sel99, GS05] [1]. If the Web has changed the way of publishing and sharing information, then search engines have changed the way of accessing information. Today the Web is so large that if a webpage is not indexed by search engines, then it has little chance of ever being visited by users. So indexable web defines an important subset of the Web that is most easily accessible to human users.

The index size of a search engine is not only of scientific interests to understand the Web, but also is an important indicator of search engine performance. While the overall quality of a search engine depends on many factors such as the prowess of the ranking algorithm, the reliability of its infrastructure etc, the calibre of its index is certainly one important factor. In recent years there has been much speculation

[1] I personally feel the term of "indexable web" misleading: nowadays many webpages are not included in the search engine indexes not because those webpages cannot be indexed, but because search engines can only index a limited number of webpages due to the infrastructure constraints. I think a more appropriate name is "indexed web".

about the sizes of the indexes of major search engines with Google, Yahoo and MSN Search vying for top spot. In November 2004, MSN Search claimed to have the largest index with 5 billion pages; immediately after, Google updated its homepage to show 8 billion indexed pages, nearly doubling its earlier figure. Then in August 2005, when Yahoo claimed to be indexing 19.2 billion pages, Google dropped its count from its homepage declaring, however, that its index size was more than three times the size of any other search engine. The question of which search engine provides the most coverage is thus of public interest and there is a need for objective evaluation methods.

1.1.1 Estimating Index Sizes via Queries

In the last ten years both the scientific literature and the popular press dealt at length with methodologies and estimates for the size of the various public web search engines and indirectly, the size of the web.

An approach initiated by Bharat and Broder [BB98] is to produce only relative sizes, that is, to estimate the ratio between the sizes of several engines. Their approach is based on pseudo-uniform sampling from each engine index, first by sampling a query pool of English words, and then by sampling the results of the chosen query, followed by a capture-recapture estimate (that is, sample one engine, test containment in the other); however their method is heavily biased in favor of "content rich" documents and thus the ultimate results are problematic. This has been recently corrected by Bar-Yossef and Gurevich [BYG06], who present a truly uniform sampling method that at least in principle should yield accurate ratios, but their method still needs to test containment, a non-trivial procedure that introduces biases of its own. Note also that both the Broder and Bharat algorithm, as well as the Bar-Yossef and Gurevich algorithm yield only documents that match at least one query from their query pool, so in fact they only estimate the relative sizes of a certain subset of the index.

Estimating absolute index sizes appears to be a more difficult task. Previous work [LG98, BB98] has had to fall back on sources other than public query interfaces, such as `http://searchenginewatch.com/reports/article.php/2156481` and `http://searchengineshowdown.com/stats`, or has had to trust the index size reported by

search engines themselves.

We propose the first methods for estimating absolute index sizes without such additional sources. We assume only access to the public query interface. First, we give a technique to count the size of specific subsets of the index. Second, we show how to use this technique to generate an estimate of the size of the entire index. To verify the effectiveness of our methods, we build a small search engine as a testbed upon a large TREC collection consisting of over 1.2 million documents. We apply our estimation methods on this search engine and find that they can obtain fairly high accuracies. We then apply our methods to estimate the index sizes of three major search engines.

We present the estimation algorithms and the experimental results in Chapter 2. This is joint work with Andrei Broder, Marcus Fontura, Vanja Josifovski, Ravi Kumar, Rajeev Motwani, Shubha Nabar, Rina Panigrahy and Andrew Tomkins, originally published in [BFJ+06].

1.1.2 Estimating Sum by Weighted Sampling

Having seen some empiric results about search engine index sizes, now we switch gear to a theoretical problem arising from the problem of estimating Web size.

Estimating Web size or search engine index sizes can be mapped to the classic problem of estimating the sum of n non-negative variables, which has numerous important applications in various areas of computer science, statistics and engineering. Measuring the exact value of each variable incurs some cost, so we want to get a reasonable estimator of the sum while measure as few variables as possible.

In most traditional applications, only uniform sampling is allowed, i.e. each time we can sample one variable uniformly at random and ask its value. It is simple to estimate the sum from such samples because the average of the samples is an unbiased estimator of the actual average. We can show that any reasonable estimator with uniform sampling requires a linear sample size if the underlying distribution is arbitrary.

We can map the problem of estimating index sizes or Web size to this problem

by partitioning the search index (or the web) into domains (or web servers), with each variable representing the size of a domain (or a web server), and estimating the size is simply estimating the sum of those variables. It is relatively easy to get the total domain (web server) number n (either by uniformly sampling IP space or this number is published periodically). For example in 1999 Lawrence and Giles estimated the number of web servers to be 2.8 million by randomly testing IP addresses; then they exhaustively crawled 2500 web servers and found that the mean number of pages per server was 289, leading to an estimate of the web size of 800 million [LG00]. Lawrence and Giles essentially used uniform sampling to estimate the sum, however, the domain size distribution is known to be highly skewed and uniform sampling has high variance for such inputs.

Now for our problem of estimating the Web size, a new sampling primitive becomes available: we can sample a variable with probability proportional to its value, which we refer to as *linear weighted sampling*. We can implement linear weighted sampling on domains as follows: uniformly sample a page from the web or a search engine index (the technique of uniform sampling a page from the web or an index is an independent research problem and has been studied in for example [HHMN00, BYG06]) and take the domain of the page; it is easy to see that the probability of sampling a domain is proportional to its size.

This new sampling primitive proposes an interesting theoretical question: how can we derive an estimation of the sum from such weighted samples? Does it improve sample complexity over uniform sampling?

We design an algorithm for sum estimation with $\tilde{O}(\sqrt{n})$ samples using only linear weighted sampling. Furthermore, if we use both uniform sampling and linear weighted sampling, we can further reduce the number of samples to $\tilde{O}(\sqrt[3]{n})$ samples. Our algorithm assumes no prior knowledge about the input distribution. We also show the two estimators are almost optimal by proving lower bounds on the sample complexity of $\Omega(\sqrt{n})$ and $\Omega(\sqrt[3]{n})$ respectively.

Finally, we show a negative result that more general weighted sampling methods, where the probability of sampling a variable can be proportional to any function of its value, do not yield better estimators: we prove a lower bound of $\Omega(\sqrt[3]{n})$ samples for

any reasonable sum estimator, using any combination of general weighted sampling methods. This implies that our sum estimator combining uniform and linear weighted sampling is almost optimal (up to a poly-log factor), hence there is no need to pursue fancier sampling methods for the purpose of estimating sum.

We present the sum estimators and hardness results in Chapter 3. This is joint work with Rajeev Motwani and Rina Panigrahy, originally published in [MPX07].

1.2 Webgraph Models

As we see from Part I, the Web is so large that even estimating its size is a hard problem, not to mention more complicated measurement and computation. Webgraph models are succinct ways of representing the Web graph while capturing its essential characteristics. A random graph model is a stochastic process that generates random graphs; in other words, it defines a probability distribution on graphs. And webgraph models are random graph models that generate graphs that with high probability preserve the most important characteristic of the Web graph, such as the power law degree distribution, the transitivity effect, the small world phenomenon.

We want to point out that social networks, a family of fast expanding networks on the Web, are observed to behave very similar characteristics as the Webgraph, so scientists use the same random graph models also for social networks. In this thesis we will use the term "webgraph models" to refer to models for both the Web graph and social networks.

Webgraph models are extremely useful theoretical tools both for researchers and for practitioners. They help scientists to develop a deep understanding of the underlying evolutionary process that shapes the Web and social networks, and to predict future development and behaviors. The models are also useful for practitioners. Due to the massive size of the Web and social networks, it is prohibitively slow and resource-consuming to run any computation or analysis of the entire Web. With webgraph models we can carry out analysis on the models; we can also use the models to generate synthetic data sets with reasonable sizes for testing and simulation.

We very briefly review the history of research on webgraph models.

The first random graph model dates back to 1959 due to Erdős and Rényi [ER59]. The Erdős-Rényi model defines a graph with n vertices, each potential edge existing independently with probability p. The Erdős-Rényi model is mainly of pure mathematical interests. It is far away from the Webgraph (as it is published long before the Web emerges); for example, its degree sequence follows a Poisson distribution rather than the power law distribution as in the Web graph.

The preferential attachment model by Barabási and Albert [BA99] is generally considered as the first model for Webgraph. This is an evolutionary model where vertices and edges are added to the graph one at a time, and the probability that a new edge is attached to an existing vertex depends on the current degree of the vertex. The model is very well received because it defines a dynamic process which gives rise to the power law degree distribution. In the next year, Kumar et al. independently proposed the copying model with similar spirit and presented a rigorous analysis of the degree distribution [KRR$^+$00].

The last decade witnessed a prosperity of research on webgraph models. I feel the research effort in this area can be divided into three main directions:

1. After observing a property of the real world Webgraph or social network graphs, use graph models to explain the underlying process that gives rise to this phenomenon, or characterize graphs that share such property. For example, the preferential attachment model explains the formation of power law degree distribution; Leskovec et al. discover that many real world graphs get denser over time with shrinking diameters, and explain this phenomenon with hierarchical graph models [LKF05]; after Kleinberg asserts that the social networks have a "searchable" property that a decentralized search algorithm can find short routing paths without global knowledge of the graph [Kle00, Kle01], Duchon et al. identify a wide class of graphs with such searchability [DHLS06].

2. Propose new webgraph models or extend existing models, and analyze its properties. After the seminal work of preferential attachment model and copying model, a lot variations and extensions have been proposed to enhance the basic model, such as allowing deletion of edges and nodes, associating quality or

fitness to nodes, changing the function of how the linking probability depends on the degree (see for example [GB01, DEM01, PRU02, CF03]). There are also static models where the vertex set is fixed and edges are generated according to certain distribution, such as configuration model [ACL00], Kronecker graphs [LCKF05, MX07], random dot product graphs [YS07].

3. Apply webgraph models to study real world applications. For example, Berger et al. uses the preferential attachment model to study the spread of viruses in networks [BBCS05]; Motwani and Xu use the preferential attachment with fitness model to study the impact of ranking algorithms on the evolution of webpage popularity [MX06].

My research expands all those three directions. The second part of the thesis includes three projects on webgraph models: in the first direction, we give a simple characterization of graphs that are searchable, and use this characterization to show a monotonicity property (Chapter 4); next, we analyze the properties of a graph model called stochastic Kronecker models (Chapter 5); finally, we use random graph models to study link privacy in social networks (Chapter 6).

1.2.1 Deterministic Decentralized Search in Random Graphs

Since Milgram's famous "small world" experiment [Mil67], it has generally been understood that social networks have the property that a typical node can reach any other node through a short path (the so-called "six degrees of separation"). An implication of this fact is that social networks have small diameter. Many random graph models have been proposed to explain this phenomenon, often by showing that adding a small number of random edges causes a highly structured graph to have a small diameter (e.g., [WS98, BC88]). A stronger implication of Milgram's experiment, as Kleinberg observed [Kle00], is that for most social networks there are decentralized search algorithms that can *find* a short path from a source to a destination without a global knowledge of the graph. As Kleinberg proved, many of the random graph models with small diameter do not have this property (i.e., any decentralized search algorithm in such graphs can take many steps to reach the destination), while in certain

graph models with a delicate balance of parameters, decentralized search is possible. Since Kleinberg's work, there have been many other models that provably exhibit the searchability property [KLNT06, Fra05, Sli05, LNNK+05, Kle01, DHLS06]; however, researchers still lack a good understanding of what contributes to this property in graphs.

We look at a general framework for searchability in random graphs. We consider a general random graph model in which the set of edges leaving a node u is independent of that of any other node $v \neq u$. This framework includes models such as the directed variant of the classical Erdős–Rényi graphs [ER59], random graphs with a given expected degree sequence (e.g., [CL03]), ACL graphs [ACL00], long-range percolation graphs [Kle00], hierarchical network models [Kle01], and graphs based on Kronecker products [LCKF05, ACK+07], but not models such as preferential attachment [BA99] in which the distribution of edges leaving a node is dependent on the other edges of the graph. It is worth noting that, in a random graph model where edges can have arbitrary dependencies, the search problem includes arbitrarily difficult learning problems as special cases, and therefore one cannot expect to have a complete characterization of searchable graphs in such a model.

We restrict the class of decentralized search algorithms that we consider to deterministic memoryless algorithms that succeed in finding a path to the destination with probability 1. This is an important class of search algorithms, and includes the decentralized search algorithms used in Kleinberg's work on long-range percolation graphs and hierarchical network models. For this class, we give a simple characterization of graphs that are searchable in terms of a node ordering property. We will use this characterization to show a monotonicity property for searchability: if a graph is searchable in our model, it stays searchable if the probabilities of edges are increased.

We present the results in detail in Chapter 4. This is joint work with Esteban Arcaute, Ning Chen, Ravi Kumar, David Liben-Nowell, Mohammad Mahdian and Hamid Nazerzadeh, originally published in [ACK+07].

1.2.2 Stochastic Kronecker Graphs

A generative model based on Kronecker matrix multiplication was recently proposed by Leskovec et al. [LCKF05] as a model that captures many properties of real-world networks. In the model, each vertex in the graph is labelled by a vector, and edges between vertices exist with certain probabilities decided by the labels of the two endpoints and an initiator matrix. Leskovec et al. observe that this model exhibits a heavy-tailed degree distribution, and has an average degree that grows as a power law with the size of the graph, leading to a diameter that stays bounded by a constant (the so-called *densification power law* [LKF05]). Furthermore, Leskovec and Faloutsos [LF07] fit the stochastic model to some real world graphs, such as Internet Autonomous Systems graph and Epinion trust graphs, and find that Kronecker graphs with appropriate 2×2 initiator matrices mimic very well many properties of the target graphs.

Most properties of the Kronecker model (such as connectivity and diameter) are only rigorously analyzed in the deterministic case (i.e., when the initiator matrix is a binary matrix, generating a single graph, as opposed to a distribution over graphs), and empirically shown in the general stochastic case [LCKF05].

We analyze some basic graph properties of stochastic Kronecker graphs with an initiator matrix of size 2. This is the case that is shown by Leskovec and Faloutsos [LF07] to provide the best fit to many real-world networks. We give necessary and sufficient conditions for Kronecker graphs to be connected or to have giant components of size $\Theta(n)$ with high probability[2]. Our analysis of the connectivity of Kronecker graphs is based on a general lemma about connectivity in general random graphs that might be of independent interest. We prove that under the parameters that the graph is connected with high probability, it also has a constant diameter with high probability. This unusual property is consistent with the observation of Leskovec et al. [LKF05] that in many real-world graphs the effective diameters do not increase, or even shrink, as the sizes of the graphs increase, which is violated by many other random graph models with increasing diameters. We also show that Kronecker graphs

[2]Throughout the thesis by "with high probability" we mean with probability $1 - o(1)$.

are not "searchable" even with randomized routing algorithms – they do not admit short (poly-logarithmic) routing paths by decentralized routing algorithms based on only local information. Finally, we extend the results to an alternative model where the labels of vertices are chosen randomly.

We present the results in detail in Chapter 5. This is joint work with Mohammad Mahdian, originally published in [MX07].

1.2.3 Link Privacy in Social Networks

Participation in online social networks is becoming ubiquitous in the last couple of years. A major part of the value for a user of participating in an online social network such as LinkedIn, or a web-service with an online community such as LiveJournal, lies in the ability to leverage the structure of the social network graph. For example, one can find potential friends in the local neighborhood – this has become a new feature recently pushed out by Facebook.

However, knowledge of this social graph by parties other than the service provider opens the door for powerful data mining, some of which may not be desirable to the users. The more information a social network reveals to its users, the more vulnerable it is to privacy attacks. Hence, social networks are inevitably facing the dilemma of providing utility and protecting privacy.

There are many ways the privacy of social networks may be tampered. In our work, we focus on a particular type of privacy we call "link privacy", where relationship between users is sensitive to privacy concerns, and the link information is a valuable asset to the user and to the network owner. In such networks, a user is typically permitted only limited access to the link structure. For example, a LinkedIn user can only see the profiles and friends lists of his friends and the profiles of friends of friends. As people are becoming more and more aware of personal privacy, more and more social networks choose to opt in this kind of privacy setting. However, even though each user is given access only to a small part of the social network graph, a resourceful adversarial entity could try to stitch together local network information of different users in order to gain global information about the social graph. We focus on the

case in which an attacker, whose goal is to ascertain a significant fraction of the links in a network, obtains access to parts of the network by gaining access to the accounts of some select users. This is done either maliciously by breaking into user accounts or by offering each user a payment or service in exchange for their permission to view their neighborhood of the social network.

We study that how social networks' access control policy affects the complexity of this attack, i.e., depending on the amount of information visible to each individual user, how many users an adversary needs to subvert to reconstruct a significant fraction of the entire network structure. We use the Power Law Random Graph model [ACL00] to theoretically analyze the dependency. The analysis is confirmed by experiments on both synthetic graphs and real world social networks.

We describe both experimental and theoretical results in Chapter 6. This is joint work with Aleksandra Korolova, Rajeev Motwani and Shubha Nabar, originally published in [KMN08].

Part I

Measuring the Web Size

Chapter 2

Estimating Index Sizes via Queries

The subset of the Web that is indexed by major search engines (so called "indexable web" [Sel99, GS05]) is an important subset of the Web that is most easily accessible to human users, therefore estimating index sizes of search engines is of scientific interests to understand the Web size. Index size is also an important indicator of search engine performance. In recent years there has been much speculation about the sizes of the indexes of major search engines with Google, Yahoo and MSN Search vying for top spot. The question of which search engine provides the most coverage is thus of public interest and there is a need for objective evaluation methods.

The overall quality of a web search engine is determined not only by the prowess of its ranking algorithm, but also by the caliber of its corpus, both in terms of comprehensiveness (e.g. coverage of topics, language, etc) and refinement (e.g. freshness, avoidance of spam, etc). Obviously, comprehensiveness is not the same as size — for instance as suggested in [Bro00] one can easily build a public server for 100 billion pages each representing a random combination of Nostradamus prophecies and a dedicated search engine can as easily index them, in a feat of utter insignificance. See also `http://searchenginewatch.com/searchday/article.php/3527636` for a more serious discussion of comprehensiveness versus size.

Nevertheless, the ability to produce accurate measurements of the size of a web search engine's corpus, and of the size of slices of the corpus such as "all pages in Chinese indexed in Yahoo! from US-registered servers," is an important part of

understanding the overall quality of a corpus. In the last ten years both the scientific literature and the popular press dealt at length with methodologies and estimates for the size of the various public web search engines and indirectly, the "size of the web". (See Section 2.2.)

Perhaps surprisingly, the problem of estimating the size of a corpus slice is also important for the owners of search engines themselves. For example, even the simple problem of efficiently counting the size of a result set raises non-trivial problems. In fact, Anagnostopoulos, Broder, and Carmel [ABC05] show a Google example using the queries `george` and `washington` where the identity $|A \cup B| = |A| + |B| - |A \cap B|$ is off by 25%. (They present an algorithm that enables efficient sampling of search results, but to our knowledge, the required data structures have not been implemented in any web search engine.)

Further, the problems worsen when the goal is to estimate the *effective* size of a corpus slice; that is, the size of the slice discounted for pages that no longer exist, have been redirected, do not actually match the query, etc. In this case, even for a one term query, counting the number of documents in the term posting list gives only an upper bound of the effective size of the matching set.

An approach initiated by Bharat and Broder [BB98] is to produce only relative sizes, that is, to estimate the ratio between the sizes of several engines. Their approach is based on pseudo-uniform sampling from each engine index, first by sampling a query pool of English words, and then by sampling the results of the chosen query, followed by a capture-recapture estimate (that is, sample one engine, test containment in the other); however their method is heavily biased in favor of "content rich" documents and thus the ultimate results are problematic. This has been recently corrected by Bar-Yossef and Gurevich [BYG06], who present a truly uniform sampling method that at least in principle should yield accurate ratios, but their method still needs to test containment, a non trivial procedure that introduces biases of its own. Note also that both the Broder and Bharat algorithm, as well as the Bar-Yossef and Gurevich algorithm yield only documents that match at least one query from their query pool, so in fact they only estimate the relative sizes of a certain subset of the corpus.

Estimating absolute corpus sizes appears to be a more difficult task. Previous work

[LG98, BB98] has had to fall back on sources other than public query interfaces, such as `http://searchenginewatch.com/reports/article.php/2156481` and `http://searchengineshowdown.com/stats`, or has had to trust the corpus size reported by search engines themselves. In this chapter we propose techniques for estimating absolute corpus sizes without such additional sources.

To sum up, the ability to estimate the size of the corpus and of its various slices is important in understanding different characteristics and the overall quality of the corpus. A method for estimating absolute corpus size using a standard query interface can lead to methods for estimating the corpus freshness (i.e., fraction of up-to-date pages in the corpus), identifying over/under-represented topics in the corpus, measuring the prevalence of spam/trojans/viruses in the corpus, studying the comprehensiveness of the crawler that generated the corpus, and measuring the ability of the corpus to ably answer narrow-topic and rare queries [BYG06]. Finally, relative corpus size estimates offers competitive marketing advantage and bragging rights in the context of the web search engines.

This chapter is organized as follows. We give an overview of our methods and results in Section 2.1. Section 2.2 discusses the related work on corpus size estimation and sampling random documents from a corpus. Section 2.3 presents the basic estimator and a variance reduction method. Section 2.4 presents our two approaches for corpus size estimation using the basic estimator. Section 2.5 contains the experimental results for a large TREC collection and Section 2.6 contains the experimental results for three major search engines. Section 2.7 summaries the chapter.

2.1 Methods and Results

Our method has two parts. First, we give a technique to count the size of specific broad subsets of the entire document corpus. Second, we show how to use this technique to generate an estimate of the size of the entire corpus.

2.1.1 Counting Specific Subsets

We begin our discussion of the method with an example. Assume that we wish to count the number of pages that contain an eight-digit number. A natural approach would be to produce a random sample of, say, 1 out of every 100,000 such numbers. One could then submit a query to a search engine for each number in the sample, count up the number of resulting documents, and multiply by 100,000. Unfortunately, some documents could contain many eight-digit numbers, and might therefore be counted multiple times by this procedure.

Let us modify the scheme as follows. Any document containing at least one eight-digit number should contribute 1 to the total count of documents in the set. But if a certain document d contains k different eight-digit numbers, we will allocate its total count of 1 by distributing $1/k$ to each of the eight-digit numbers it contains. In the previous scheme, when d is returned in response to a certain query, it contributes 1 to the overall count; in the new scheme it will contribute $1/k$, which is the reciprocal of the number of eight-digit numbers on the page.

Under the new scheme, we again take a sample of eight-digit numbers and submit each as a query. We then add up for each result document the reciprocal of the number of eight-digit numbers in that result document, and again multiply the final sum by 100,000. This new value is easily seen to be an unbiased estimator of the total number of documents; we provide a formal proof in Section 2.3.

More generally, our basic estimator allows us to count many different subsets D_A of the entire corpus D. The scheme will apply whenever D_A has two properties: first, a *query pool* A can be defined such that D_A is the union of the results of all queries in the pool A. And second, for any document, it is possible to determine efficiently how many queries from the query pool would have produced the document as a result.

We have seen eight-digit numbers as an example query pool. One could also employ queries chosen carefully from a query log. We will also consider approaches to modifying the query pool on the fly in order to reduce the variance of the estimator. Finally, the techniques also allow us to filter the results of each query on the fly, for example, by entirely removing documents that contain more than maximum threshold number of query terms from the pool, or that do not conform to a certain target

length, specified as a range of bytes. The flexibility of the estimator is key to its power, as many issues that arise in the real world may be addressed by modifying the definition of D_A.

2.1.2 Employing the Basic Estimator

We have now defined a basic estimator to count the size of a subset D_A of the total corpus. This estimator may be used in many ways to perform counts of corpus sizes. The first method involves a random sample of the documents in a corpus, and an efficient function to determine whether a particular document belongs to D_A. Given this, we may simply estimate from the random sample the probability p_A that a document from the corpus belongs to D_A, and estimate the overall corpus size as $|D_A|/p_A$. We show in our experiments that this approach can provide very accurate measurements with quite reasonable sample sizes.

Nonetheless, it is quite difficult to generate a random sample of pages on the web, despite the efforts of many authors. Worse yet, when a technique is chosen and applied, there are no clean approaches to estimating the quality of the resulting sample, and so the sample bias cannot be understood. Even worse, because the problem of evaluating the quality of a random sample is so difficult, there has been very little academic work even to understand the extent to which techniques for generating random samples deviate from uniform.

Our basic estimator may also be applied in a second way to estimate the size of a corpus, without the presence of a random sample. We must resort to another assumption about the corpus, but one that is quite different from the assumption that a particular technique generates a uniform random sample. We assume that there are two query pools A and B that produce independent subsets D_A and D_B of the corpus. Note that independence here is different from disjointness. D_A and D_B may share documents, but the fraction of documents that belong to D_A should be the same whether we consider the entire corpus, or just D_B. If this is true, we may estimate the size of the corpus as $|D_A| \cdot |D_B|/|D_A \cap D_B|$.

The accuracy of this method depends heavily on the independence assumption, a

crucial question in all probabilistic IR models. If the terms are correlated then we can only produce bounds based on their correlation. The following example might help build the reader's intuition for this issue. Assume that we apply our method using two sets of perfectly independent English terms and get a very accurate estimate of corpus size. Now the engine owners double its size by adding a large number of Chinese pages. If we repeat our experiments we will report the same number as before (since we will never or seldom see a Chinese page), even though the engine size has doubled. What happened? Well, our term sets used to be independent in the old corpus but now they are correlated: if we choose a page from D_A, it is now *more* likely to belong also to D_B just by dint of being in English.

This might seem as a limitation of our method, but in fact all query based estimation methods proposed so far suffer from this query vocabulary bias, and the contribution of our paper is to give a methodology where this bias can be correctly quantified by relating it to a fundamental concept in probabilistic information retrieval [vR79].

In some ways, the meaning of uncorrelated sets is a philosophical one. The estimator may be viewed as returning an estimate of that part of the corpus in which A and B are uncorrelated. While we do not advocate this perspective, we observe that if the sets are chosen appropriately, this may be the part of the engine of most interest from a measurement standpoint or it can be used to infer the relative sizes of search engines.

2.1.3 Experimental Results

We first apply our methods to a large TREC collection consisting of over 1.2 million documents. Since we know the exact corpus size, these experiments are designed to demonstrate the effectiveness of our methods. We choose the query pool to be the set of all five-digit numbers. For the approach using random document samples, we obtain fairly high accuracies even with small number of samples. The error is significantly reduced when we modify the query pool to discard the most frequent terms in the query pool. We also estimate the corpus size using two uncorrelated

query pools: set of five-digit numbers and a set of medium frequency words.

We then apply our methods to the Web by estimating what we call the "visible" corpus size of three major search engines. We choose the query pool to be the set of all eight-digit numbers and use this to estimate the visible corpus size. We also apply the two uncorrelated query pools approach: set of eight-digit numbers and a set of medium frequency words.

2.2 Related Work

Bharat and Broder [BB98] estimated the relative sizes of search engine indexes by testing the containment of pseudo-uniform samples from engine indexes. They sampled the content of search engines using conjunctive and disjunctive queries composed of terms extracted from pages of the Yahoo! Directory. From a lexicon of 400K words, terms were combined to form around 35K queries. The sample consisted of one random URL chosen from the first 100 results returned by each queries. The fraction of the sampled pages from one search engine that are also present in the index of another search engine gives an estimate of the overlap between the two. The paper also estimates a search engine's coverage of the whole web from overlaps of pairs of search engines. A known problem with this technique is that it is biased toward content-rich pages with high rank.

The same year, Lawrence and Giles [LG98] reported size estimates by comparing query results based on a log of queries submitted to a search engine. They retrieve the entire list of matching documents from all engines and then retrieve all of the individual documents for analysis. To avoid bias toward highly ranked pages, they use only 575 queries for which all search engines retrieve all the results. However such query sampling does not provide a uniform random sample of the pages in the corpus. Later in [LG00], the authors extend the results and provide some other interesting statistics about search engine corpora and the accessible web. A big problem associated with directly comparing query results is that the approach is highly dependent on the underlying IR techniques used by the search engine to answer queries.

Guli and Signorini [GS05] repeated Bharat and Broder's experiments using Open Directory (`www.dmoz.org`) as source of keywords. They also used a modified technique to combine more than 2M terms into queries.

A core primitive in search engine size estimation is a way to obtain a uniform sample from its corpus using the query interface. Several papers report techniques to uniformly sample a search engine corpus using only the public query interface [BYBC+00, HHMN00, LG98, RPLG01, BYG06]. Many of them are based on random walks. A good survey of the methodologies and techniques used for the Web size and search engine size estimation is in [BI06].

Recently, Bar-Yossef and Gurevich [BYG06] propose two new methods to obtain an unbiased sample of a search engine corpus. The first method is based on sampling, where queries are generated as word phrases from a corpus of documents and queries that return too many results are rejected. At a very high level, the analytic ideas in this method are similar to ours. The second method is based on random walks on documents and terms, but suitably unbiased using statistical techniques in order to produce a provably uniform document sample.

An interesting related issue is the study of mirrored hosts or duplicate pages and there has been much work done in this area; see, for instance, [BB99, BBDH00]. Duplication is an important issue affecting the search quality of search engines, but the focus of this paper will be on size estimates.

Liu, Yu, and Meng [LYM02] propose a method for estimating the corpus size based on the idea of two independent sets. Let D_1 and D_2 be two independent random sample of documents and let $D_3 = D_1 \cap D_2$. The search engine size is then estimated to be $|D_1||D_2|/|D_3|$. This idea is somewhat related to our method of uncorrelated query pools.

Callan and Connell [CC01] proposed query-based sampling in the context of distributed information retrieval for acquiring "resource descriptions" that accurately represent the contents of each database without relying on their internals. This work is related to ours since we try to estimate corpus size based only on the search engine's public API. Wu, Gibb, and Crestani [WGC03] propose methods for estimating and maintaining archive size information.

2.3 Basic Methods

In this section we discuss the basic estimator that will be the backbone of all our approaches. The mean of this unbiased estimator can be related to the corpus size. We assume that the index supports the basic query interface: given a query, return *all* the documents that match the query.

The first naive idea would be to use random queries and construct an estimator based on the number of documents returned for such queries. Unfortunately, this does not work, the difficulty being that there is no way of knowing the universe of all queries. Without this knowledge, it may not be possible to obtain an unbiased estimator.

We circumvent this difficulty by working with a *known* and *fixed* set of queries, called a *query pool*. For simplicity of exposition, we assume that each query is just one term. However, our methods will apply to any query pool that satisfies two conditions: first, the size of the pool should be known, and second, it should be possible to determine for any document how many queries in the pool match this document. We show (Lemma 1) how to construct an estimator whose mean is the number of documents in the corpus that match at least one query from this query pool.

Notation. Let D be the set of documents. We treat documents as a set of terms and use the notation "$a \in d$" to indicate that a term of query a occurs in the document d.

A query pool is a set of terms. For a query pool A, let $D_A \subseteq D$ be the set of documents such that every document in D_A contains at least one term in A. Define the *weight of a document* $d \in D_A$ with respect to A to be the inverse of the number of terms in A that occur in the document, i.e.,

$$w_d^A = \frac{1}{|d \cap A|}. \tag{2.1}$$

The definition of D_A guarantees that all weights are finite. The *weight of a query* a with respect to A is simply the weight of all documents containing the query, defined

as follows:

$$w_a^A = \sum_{\substack{d \in D_A: \\ d \ni a}} w_d^A. \tag{2.2}$$

2.3.1 Basic Estimator

Intuitively, if we encounter a document d which contains many query terms, each term should be given only partial credit for the document. Thus, we define our basic estimator as follows:

$$W_{A,D} = E_{a \sim A}[w_a^A], \tag{2.3}$$

i.e., the average weight of a query with respect to A.

We now show that this quantity times $|A|$ is an unbiased estimator of the number of documents containing at least one term in A.

Lemma 1.

$$W_{A,D} = \frac{|D_A|}{|A|}.$$

Proof.

$$\begin{aligned}
|A| \cdot W_{A,D} &\overset{(2.3)}{=} |A| \cdot E_{a \in A}\left[\sum_{\substack{d \in D_A: \\ d \ni a}} w_d^A\right] \\
&= \sum_{a \in A} \sum_{d \in D_A, d \ni a} w_d^A \\
&\overset{\text{swap}}{=} \sum_{d \in D_A} \sum_{a \in A, a \in d} w_d^A \\
&= \sum_{d \in D_A} w_d^A \left(\sum_{a \in A, a \in d} 1\right) \\
&\overset{(2.2)}{=} \sum_{d \in D_A} 1 \\
&= |D_A|.
\end{aligned}$$

□

Thus, Lemma 1 guarantees an unbiased estimator whose mean is $|D_A|/|A|$. By sampling the query pool uniformly at random, $|D_A|$ can be estimated. We now discuss the issues in sampling.

2.3.2 Sampling

All the estimators such as $W_{A,D}$ can be estimated by the usual sampling techniques. We will illustrate the method to estimate $W_{A,D}$. Let X be the random variable given by $\sum_{d \in D_A, d \ni a} w_d^A$, where the query a is chosen uniformly at random from the query pool A. Clearly $E[X] = W_{A,D}$. We pick k terms $a_1, \ldots, a_k$ independently and uniformly at random from A and estimate the quantity $X_i = \sum_{d \in D_A, d \ni a_i} w_d^A$ for each of the a_i's. We then compute an *averaged estimator* $\overline{X}$ by averaging $X_1, \ldots, X_k$. It is easy to see that $E[\overline{X}] = E[X] = W_{A,D}$. Using Chebyshev's inequality, it follows that

$$\Pr[|\overline{X} - E[X]| \geq \epsilon E[X]] \leq \frac{1}{k} \left(\frac{\text{var}[X]}{\epsilon^2 E^2[X]} \right).$$

Using this expression, if $k \geq (10/\epsilon^2)\text{var}[X]/E^2[X]$ for instance, then with probability at least 0.1, the averaged estimator $\overline{X}$ approximates $W_{A,D}$ to within factor $(1 \pm \epsilon)$. To boost the probability of success from 0.1 to $1 - \delta$ for an arbitrary δ, we can compute $O(\log 1/\delta)$ such averaged estimators and take their median value.

Ideally, we wish to make k as small as possible. To be able to do this, we need to make sure that $E[X]$ is not too small. For instance, this means that we cannot pick the query pool A to be terms that occur very rarely, since this will make the estimation of p_A harder. The second point to note is that if the variance var$[X]$ is large, then it implies that k has to be large. We address the second issue in greater detail in the next section.

2.3.3 Variance Reduction

The variance of the random variable X can be very large. As an example, consider the following extreme scenario. The query pool A decomposes into A_1 and A_2 so that $|A_1| \ll |A_2|$ but each $a_1 \in A_1$ occurs in a large number of documents in D_A and each

such document contains only terms in A_1. Consequently, the contribution to $W_{A,D}$ by $a_1 \in A_1$ is large. However, few random samples from A will hit A_1, owing to its small cardinality. Therefore, the number of samples need to be high. In other words, the distribution corresponding to X can have a heavy tail and we need a lot of samples to hit the tail (we give some evidence of this in Figure 2.1).

We now illustrate a generic method to ameliorate the problem: identify the tail and truncate it. Let $A' \subset A$ be those queries whose weights contribute to the tail of X of mass β; random sampling of documents in D, once again, can be used to identify candidate queries in A'. We then redefine a new query pool $\tilde{A}$ such that $\tilde{A} = A \setminus A'$. The hope is that the random variable $W_{\tilde{A},D}$ has lower variance than $W_{A,D}$. We now proceed to formalize this and analyze conditions under which truncation of a random variable causes its variance to reduce.

Notation. Let f be a probability distribution on an ordered domain U and let the random variable $X \sim f$. Consider the conditional random variable $Y = X \mid X < \tau$, i.e., its distribution $f|_{<\tau}$ is given by truncating f at τ and rescaling by the conditional mass $\Pr_{X \sim f}[X < \tau]$. We would like to study var$[Y]$ vs. var$[X]$.

If f can be arbitrary, then there can be no relationship between var$[X]$ and var$[Y]$. It is straightforward to construct an f such that var$[Y] <$ var$[X]$. With little effort, we can also construct an f such that var$[Y] >$ var$[X]$. Let f be a distribution with support of size three given by $f(-\epsilon) = f(\epsilon) = \delta/2$ and $f(1) = 1 - \delta$, for parameters $0 < \epsilon, \delta < 1$ to be specified later. Let $\tau = 1$ be the threshold. Let $X \sim f$ and let $Y = X \mid X < \tau$. Now, it is easy to see that $E[X] = 1 - \delta$ and $E[X^2] = (1 - \delta) + \delta\epsilon^2$ and so

$$\text{var}[X] = (1 - \delta) + \delta\epsilon^2 - (1 - \delta)^2 = \delta(1 - \delta + \epsilon^2).$$

On the other hand, $E[Y] = 0$ and var$[Y] = E[Y^2] = \epsilon^2$. Hence, if $\epsilon > \sqrt{\delta}$, we can achieve var$[Y] >$ var$[X]$.

However, if f is monotonically non-increasing, then we show that var$[Y] \leq$ var$[X]$, i.e., truncation helps to reduce variance. In fact, in the extreme case, truncation can turn infinite variance into finite variance. When the distribution is a power law, we show a quantitative bound of the reduction in variance.

Monotone distributions

For simplicity, let us assume f is a discrete monotonically non-increasing distribution. Without loss of generality, let the support of f be $[n]$ with $f(1) \geq \cdots \geq f(n)$ and without loss of generality, let $\tau = n$. Let $g = f|_{<\tau}$, and $X \sim f, Y \sim g$. Notice that for $i = 1, \ldots, n-1$, $g(i) = f(i)/(1 - f(n))$. Let $\mu = E[f]$.

Lemma 2. $f(n)(n-\mu)^2 \geq \sum_{i=1}^{n-1}(g(i) - f(i))(i-\mu)^2$.

Proof. First, we show that for $1 \leq i \leq n$,

$$(n-\mu)^2 \geq (i-\mu)^2, \tag{2.4}$$

or equivalently, $\mu \leq (1+n)/2$. Without loss of generality, assume n is odd and let $n' = n/2$. Let $a_1 = \min_{i<n'} f(i) = f(\lfloor n' \rfloor)$ and let $a_2 = \max_{i>n'} f(i) = f(\lceil n' \rceil)$. Thus, $a_1 \geq a_2$. Observe that $\mu = n' - \sum_i (n'-i)f(i)$ and

$$\begin{aligned}
\sum_i (n'-i)f(i) &= \sum_{i>n'}(n'-i)f(i) - \sum_{i<n'}(i-n')f(i) \\
&\geq \sum_{i>n'}(n'-i)a_2 - \sum_{i\leq n'}(i-n')a_1 \\
&= (a_2 - a_1)\sum_{i<n'}(n'-i) \\
&\geq 0,
\end{aligned}$$

and thus, $\mu \leq n' = n/2$, establishing (2.4). Finally,

$$\begin{aligned}
&\sum_{i=1}^{n-1}(g(i) - f(i))(i-\mu)^2 \\
&\overset{(2.4)}{\leq} \sum_{i=1}^{n-1}(g(i) - f(i))(n-\mu)^2 \\
&= (n-\mu)^2 \sum_{i=1}^{n-1}(g(i) - f(i)) \\
&= (n-\mu)^2 f(n).
\end{aligned}$$

□

Lemma 3. $\mathrm{var}[Y] \leq \mathrm{var}[X]$.

Proof.

$$\begin{aligned}
\mathrm{var}[X] &= \sum_{i=1}^{n} f(i)(i-\mu)^2 \\
&= \left(\sum_{i=1}^{n-1} f(i)(i-\mu)^2\right) + f(n)(n-\mu)^2 \\
&\overset{\text{Lemma 2}}{\geq} \sum_{i=1}^{n-1} f(i)(i-\mu)^2 + \sum_{i=1}^{n-1}(g(i)-f(i))(i-\mu)^2 \\
&= \sum_{i=1}^{n-1} g(i)(i-\mu)^2 \\
&\overset{(*)}{\geq} \sum_{i=1}^{n-1} g(i)(i-E[g])^2 \\
&= \mathrm{var}[Y],
\end{aligned}$$

where (*) follows since the convex function $\sum_i g(i)(i-y)^2$ is minimized at $y = E[g]$. □

Power law distributions

We consider the important case when f is given by a power law. In this case we obtain a quantitative bound on the reduction of the variance. For simplicity, we assume that f is a continuous distribution defined on $[1, \infty)$. Let $f(x) = \Pr[X = x] = \alpha x^{-\alpha-1}$ for some $\alpha > 1$; the cumulative distribution function of X is then $\Pr[X \leq x] = 1 - x^{-\alpha}$.

Suppose we discard the β fraction of the mass in the tail of of X, i.e., let x_0 be such that

$$\int_{x_0}^{\infty} \alpha x^{-\alpha} dx = \beta.$$

Since $\beta = \Pr[X > x_0] = x_0^{-\alpha}$ by definition, we get

$$x_0 = \beta^{-1/\alpha}. \tag{2.5}$$

Let Y be the random variable truncated at x_0 and rescaled by $1/(1-\beta)$. We first show the following useful inequality.

Lemma 4. $(1-\beta)(1-\beta^{1-2/\alpha}) \leq (1-\beta^{1-1/\alpha})^2$.

Proof.

$$\begin{aligned}(1-\beta)(1-\beta^{1-2/\alpha}) &= 1-\beta-\beta^{1-2/\alpha}+\beta^{2-2/\alpha} \\ &\overset{\text{am-gm}}{\leq} 1-2\sqrt{\beta\cdot\beta^{1-2/\alpha}}+\beta^{2-2/\alpha} \\ &= 1-2\beta^{1-1/\alpha}+\beta^{2-2/\alpha} \\ &= (1-\beta^{1-1/\alpha})^2.\end{aligned}$$

□

Lemma 5. *If* $\alpha \leq 3$, *then* $\text{var}[X] = \infty \gneqq \text{var}[Y]$. *If* $\alpha > 3$, *then* $\text{var}[Y] \leq ((1-\beta^{1-1/\alpha})/(1-\beta))^2\text{var}[X] \leq \text{var}[X]$.

Proof. The easy case is when $\alpha \in (2,3]$, in which case $\text{var}[X] = \infty \gneqq \text{var}[Y]$. For the rest, we will assume $\alpha > 3$.

$E[X] = \alpha/(\alpha-1)$ and $E[X^2] = \alpha/(\alpha-2)$ and so,

$$\text{var}[X] = \frac{\alpha}{(\alpha-2)(\alpha-1)^2}. \tag{2.6}$$

By integrating the pdf of Y from 0 to x_0 and using the value of x_0 from (2.5), we obtain

$$E[Y] = \frac{\alpha}{(1-\beta)(\alpha-1)}(\beta^{1-1/\alpha}-1),$$

and

$$E[Y^2] = \frac{\alpha}{(1-\beta)(\alpha-2)}(\beta^{1-2/\alpha}-1),$$

from which,

$$\begin{aligned}\text{var}[Y] &= \frac{\alpha}{(1-\beta)(\alpha-2)}(\beta^{1-2/\alpha}-1) \\ &\quad - \frac{\alpha^2}{(1-\beta)^2(\alpha-1)^2}(\beta^{1-1/\alpha}-1)^2. \end{aligned}\tag{2.7}$$

Finally, using (2.6) and (2.7),

$$\begin{aligned}
&\frac{\mathrm{var}[Y]}{\mathrm{var}[X]} \\
&= \frac{(\alpha-2)(\alpha-1)^2}{(1-\beta)}\left(\frac{1-\beta^{1-2/\alpha}}{\alpha-2} - \frac{\alpha(1-\beta^{1-1/\alpha})^2}{(1-\beta)(\alpha-1)^2}\right) \\
&= \frac{(\alpha-1)^2(1-\beta)(1-\beta^{1-2/\alpha}) - \alpha(\alpha-2)(1-\beta^{1-1/\alpha})^2}{(1-\beta)^2} \\
&\overset{\text{Lemma 4}}{\leq} \frac{(1-\beta^{1-1/\alpha})^2}{(1-\beta)^2} \cdot ((\alpha-1)^2 - \alpha(\alpha-2)) \\
&= \left(\frac{1-\beta^{1-1/\alpha}}{1-\beta}\right)^2.
\end{aligned}$$

Notice that since $0 < \beta < 1$ and $\alpha > 0$, we have $0 < \beta \leq \beta^{1-1/\alpha} < 1$. □

2.4 Applications of the Basic Estimator

Recall that our goal is to devise a method to estimate the corpus size, i.e., the number of documents in a search engine corpus. In this section we present two algorithms that achieve this goal. The two algorithms are based on different assumptions about the capabilities of the index. In the first algorithm, we assume that a uniform random document can be obtained from the corpus. In the second algorithm, we do away with the uniform document sampleability assumption, but instead use a different assumption, which will be evident from the description below.

2.4.1 Corpus Size via Random Documents

Suppose we can obtain a uniform random sample of documents in D. Then, using such a sample, we can estimate the fraction of documents in D that are also in D_A. Then, using D_A and Lemma 1, we can estimate the corpus size.

Corollary 6. *If* $p_A = |D_A|/|D|$, *then*

$$|D| = \frac{|A|}{p_A} \cdot W_{A,D}.$$

Thus, by estimating p_A via the random sample of documents, we can estimate the corpus size.

2.4.2 Corpus Size via Uncorrelated Query Pools

Second, suppose uniform random sampling of documents in the corpus is not possible. We show that even under this constraint, the corpus size can be estimated provided we make assumptions about the query pool. The core idea is to use an additional query pool B with the property that B is reasonably uncorrelated with respect to A in terms of occurrence in the corpus (Corollary 8). In other words, we use the independence of the query pools A and B.

Let A, B be two query pools. Let $D_{AB} \subseteq D$ be the set of documents that contain at least one term in A and one term in B. Then, from Lemma 1, we have

Corollary 7.

$$\frac{|D_{AB}|}{|A|} = W_{AB,D} = E_{a \in A}\left[\sum_{\substack{d \in D_{AB} \\ d \ni a}} w_d^A\right]$$

Here, notice that the summation is over all documents d that contain a and at least one term in B, i.e., the documents are "filtered" by the query pool B. Thus, Corollary 7 can be used to estimate $|D_{AB}|$.

The significance of Corollary 7 is that it lets us estimate $|D|$ without using any random access to documents in D, modulo appropriate assumptions on A and B. Suppose A and B are uncorrelated, i.e., $\Pr[d \in D_A \mid d \in D_B] = \Pr[d \in D_A] = p_A$, then it is easy to see

Corollary 8. *If* A *and* B *are uncorrelated set of terms, then*

$$|D| = \frac{|D_A| \cdot |D_B|}{|D_{AB}|}.$$

Thus, Corollary 8 can be used to estimate the corpus size without resorting to sampling documents in the corpus uniformly at random.

While Corollary 8 is very attractive if the query pool A and B are perfectly uncorrelated, in practice it may be hard to construct or obtain such sets. However, we observe even if the set of terms A and B are correlated, the measure of correlation directly translates to the quality of approximation of the corpus size. More precisely, let $p_{A|B}$ denote $\Pr_{d \in D}[d \in D_A \mid d \in D_B]$. If $c_1 p_A \leq p_{A|B} \leq c_2 p_A$ for some non-zero constants $c_1 \leq c_2$, then it follows along the lines of Corollary 8 that

$$c_1 \frac{|D_A| \cdot |D_B|}{|D_{AB}|} \leq |D| \leq c_2 \frac{|D_A| \cdot |D_B|}{|D_{AB}|}.$$

2.5 Experiments on TREC

In this section we present our experiments on the TREC collection.

2.5.1 Data and Methodology

The document set D consists of 1,246,390 HTML files from the TREC `.gov` test collection; this crawl is from early 2002 (`ir.dcs.gla.ac.uk/test_collections/govinfo.html`). Our methodology is to first define a query pool, pick sufficiently many sample terms from this query pool, query the index for these sample terms, and then compute the weight of the sample query terms according to (2.3). All our sampling is done with replacement. For all the experiments, we compute 11 averaged estimators as discussed in Section 2.3.2 and the final estimate is the median of these 11 averaged estimators. We preprocess the entire data by applying `lynx` on each of the files to process the HTML page and output a detagged textual version. This is so that the meta-data information in the HTML files is not used in the indexing phase. Moreover, this also serves as a data cleaning phase. We tokenize the documents using whitespace as the separator.

An important point to keep in mind is a consistent interpretation of a term "occurring" in a document. This plays a role in two different cases. First, in the answers

returned by the index — the index should return all and only documents in which the given query term occurs. Second, in the computation of weight of a term in (2.1) — we need to know how many terms from A occur in the document. The first of these cases can be handled easily by having a very "strict" definition of "occurring" and checking to see if each document returned by the index for a given query term actually contains the term according to the definition of "occurring". The second case is trickier, unless we have a reasonable understanding of the way the indexer operates. For sake of this experiment, we adopt the safe approach by hand-constructing a straight-forward indexer.

2.5.2 Corpus Size via Random Documents

We illustrate the performance of our methods using two different query pools and random documents from the corpus. For both the query pools A and B, we estimate p_A and p_B by examining random TREC documents. The experiment in this case has been constructed to remove any systematic bias from the estimate of p_A and p_B in order to evaluate how well the techniques perform when the random samples are good.

A = set of five-digit numbers

We choose the first query pool A to be the set of all five-digit numbers, including numbers with leading zeros. Thus, $|A| = 10^5$. Our notion of a term occurring in a document is governed by the regular expression `/^\d{5}$/`. This will, for instance, ensure that $12,345$ or 12345.67 are not valid matches for the term $12345 \in A$. Under these definitions, we have $|D_A| = 234,014$ and so $p_A = 0.1877$ and 94,918 terms in $|A|$ actually occur in some document in D.

The following table shows the error of results of our experiments with set A. Here, error is measured relative to $|D|$, which we know ($|D| = 1,246,390$), as a function of the number of samples used for each averaged estimator.

Samples	100	500	1000	2000	5000
Error (%)	48.50	37.56	13.31	16.12	6.44

As we see, the error of the method is quite low.

Variance Reduction. We next investigate whether the performance of this method can be improved by applying the variance reduction techniques presented in Section 2.3.3. To understand this further, we compute the weights of all terms in A to see if it indeed has a heavy tail. Figure 2.1 shows the distribution of weights of terms in the set $|A|$. From the figure, it is easy to see that the distribution conforms to a power law (the exponent of the pdf is $\sim$ -1.05). So, there is hope of improving

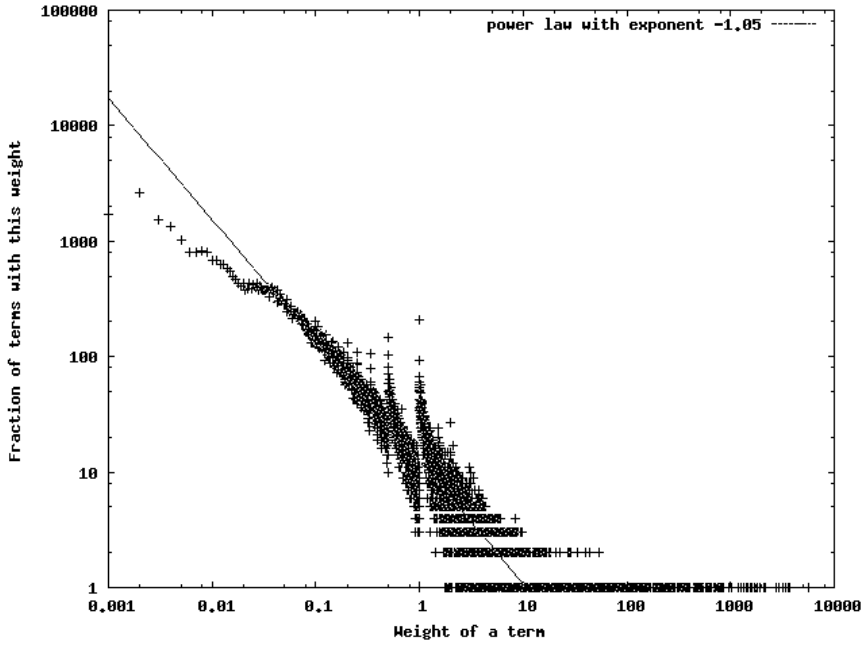


Figure 2.1: Distribution of weights of terms in $|A|$.

the performance of the method by the variance reduction method outlined in Section 2.3.3. To identify the candidate elements in A' — the terms with highest weights — we resort to sampling once again. We randomly sample documents from the corpus, and for each term $a \in A$ that occur in a document d, we maintain a histogram of its weight according to w_d^A as in (2.1); note that these weights are in fact approximations to their actual weights w_a^A as in (2.2). We finally sort the terms in decreasing order of their accumulated weights and declare the terms that contribute to the top 75% of the weights to be present in A'.

This operation defines a new query pool $A \setminus A'$, upon which our methods apply as before. Thus, the correctness of this approach is not dependent on the exact determination of the most frequent terms, or even upon uniform sampling. The method is always correct, but the variance will only benefit from a more accurate determination. The results are shown below.

Samples	100	500	1000	2000	5000
Error (%)	14.39	16.77	19.75	11.68	0.39

We can see that overall this method obtains estimates with significantly lower error, even with few samples.

B = set of medium frequency words

We repeat the same experiments with a different set B of terms. This time we want to choose B with two properties: none of the terms in B matches too many documents and B is reasonably large. The former property will reduce the variance of the sampling steps if the occurrence of terms is correlated positively with the weights. We provide some evidence towards this. See Figure 2.2. We first extract all terms in the document set D using whitespace separated tokenization and then we sort the terms according to their frequency of occurrence. We then pick B to be the terms (that are not purely numbers) that occur from position 100,000 to position 200,000 in this list. Thus, $|B| = 100{,}000$. Under these definitions, we obtain $|D_B| = 396,423$ and so $p_B = 0.3180$.

Samples	100	500	1000	2000	5000
Error (%)	3.51	3.66	1.24	0.95	1.80

We see that the method performs extremely well and this can be attributed to our careful choice of the set B. In fact, our experiments showed no appreciable improvement when we applied the variance reduction method to the set B. This is to be expected, as B is specifically constructed so that no term occurs significantly more often than any other term, and so no term will introduce substantial skew into the measurement.

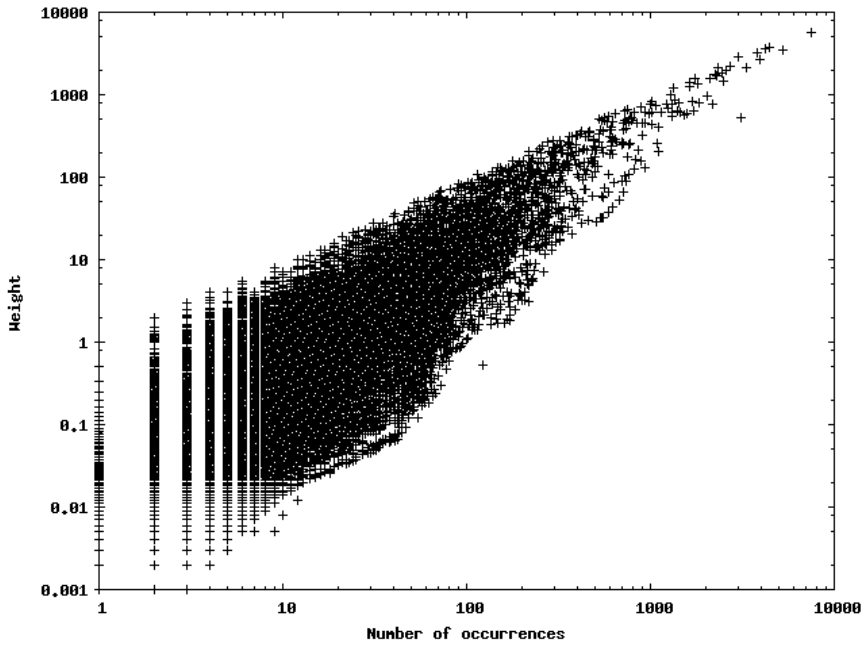


Figure 2.2: Correlation of weights and number of occurrences of terms in A.

2.5.3 Corpus Size via Uncorrelated Query Pools

We need to construct query pools A and B that are reasonably uncorrelated. To do this, we first try A and B as defined before. Since we have the corpus available, we can actually measure the amount of dependence between the sets A and B. We explicitly calculate $p_{A|B}$ and p_A. The values are $p_A = 0.1877$ whereas $p_{A|B} = 0.2628$ indicating some correlation between the term sets. To reduce the correlation, we modify the sets D_A and D_B to be slightly different.

We set D'_A to be the set of documents that contain *exactly one* term from A; D'_B is defined analogously. We do this in order to reduce the potential correlation caused by large documents. Using this, we calculate $p'_A = 0.1219, p'_B = 0.1437$, and $p'_{A|B} = 0.1455$, indicating a significant reduction in correlation.

Modifying Lemma 1, we can estimate $|D'_A|$ and $|D'_B|$. We use Corollary 7 to estimate $|D_{AB}|$. We proceed as before, except that in computing the weight of a sample term from B, we discard documents that do not contain any term from A. The following table shows the error of the method.

Samples	1000	2000	5000
Error (%)	27.64	23.01	21.32

As we see from the results, the method obtain a reasonable estimate of the corpus size, without using any random sample of documents. Obviously, the accuracy of this method can be improved if we work with even less correlated query pools.

2.6 Experiments on the Web

In this section we present our experiments on the Web. We used the public interface of three of the most prominent search engines, which we refer to as SE_1, SE_2, and SE_3. We present three series of results for the Web. First we use the basic estimator defined in Section 2.3.1 to compute the relative sizes of the three engines. We then use the random document approach and the uncorrelated query pool approach defined in Section 2.4 to compute absolute sizes of the "visible" portion of the engines. (See below.)

2.6.1 Method

Our methodology for the Web is similar to the one we used for TREC. We first define a query pool, sample sufficiently many terms from the query pool, and compute the weights as defined in (2.3). We postprocess the results using `lynx` to remove the HTML markup of on each of result pages. Then we compute the weights based on this cleaned version of the result pages, using whitespace tokenization. We use the same definition of a term "occurring" in a document as in the TREC experiment — the cleaned version of the document must contain the queried term. This eliminates documents in which the query term does not appear in the document, including documents in which the query term appears only in anchor text, dead pages (404), and pages that match stemmed versions of the query term. Furthermore, since the capabilities of `lynx` as an HTML parser are limited, many pages are "invisible" to our method, in particular most pages containing Java script, frames, Flash, etc. Thus the absolute sizes that we are reporting, are only estimates for the sets "visible"

to our methodology. Assuming that each search engine carries the same proportion of "visible" pages, we can obtain relative sizes of the search engines, but absolute estimates are still an elusive goal.

2.6.2 Results

We first compute the relative sizes of engines SE_1, SE_2, and SE_3, using the our basic estimator. We define A to be the set of all eight-digit numbers, including numbers with leading zeros, thus, $|A| = 10^8$, and follow the approach detailed for the TREC experiments. The following table shows the results of our experiments with query pool A.

Engine	SE_1	SE_2	SE_3
Samples	3486	3529	3433
$W_{A,D}$	0.29	0.51	0.08

If we assume that p_A is the same for all the three search engines, i.e., that the three engines index the same proportion of pages with eight-digit numbers, the above values provide the relative sizes of the engines corpus. However, p_A varies from engine to engine. We used a random sample of pages provided to us by Bar-Yossef and Gurevich produced according to the random walk approach described in their work [BYG06]. The following table shows p_A for the three engines and the sample sizes.

Engine	SE_1	SE_2	SE_3
Samples	199	137	342
p_A	0.020	0.051	0.008

Using these values, we can easily compute the absolute sizes of the "visible" portion of the three search engines (in billions) as below.

Engine	SE_1	SE_2	SE_3
Size	1.5	1.0	0.95

Next, we used the uncorrelated query pool approach to estimate the absolute corpus sizes. The query pool A is again the set of all eight-digit numbers, and as

for TREC, we chose the query pool B to be the medium frequency words, which was determined from a histogram of term counts from the index of one of the search engines. We assume that these words are also the medium frequency words on the other two engines. Furthermore, we also verified that when queried, all three engines always returned less than 1000 results for all our samples from pool B. However, for the Web there is no straightforward way to verify that pool A and pool B are indeed independent or to estimate their correlation.

The following table shows the resulting estimates for the "visible" portion of the three search engines (in billions) using uncorrelated query pools.

Engine	SE_1	SE_2	SE_3
Size	2.8	1.9	1.1

As can be readily seen, the estimates are now larger although still much less than published values for the entire corpus of these engines while the relative sizes are fairly consistent. The next section explains why this happens.

2.6.3 Discussion and Caveats

Even though our methods are intended to produce absolute estimates for the corpus size, they are still affected by many idiosyncracies that exist in web documents. Here we list some of the caveats while drawing conclusions from the results above.

1. Even though our methods are fairly generic, they end up measuring only a large slice of the corpus (rather than the corpus in its entirety), what we call the "visible" corpus. In particular, our methods exclude documents in the so-called frontier. These documents are not indexed in full, nevertheless the search engine is aware of their existence through anchortext and might return these documents as results to queries (especially when these queries match the anchortext). Our method might end up discarding these documents since the query term may not explicitly occur in the document. Hence, our method will underestimate the corpus size. A similar comment to the many types of documents that are not parsed by our HTML parser (`lynx`). Although `lynx`

has the advantage of being very consistent, in future work we intend to use a more sophisticated parser.

2. Even though we chose our query pools carefully, for about 3% of the queries in the numbers query pool, the search engines return more than 1000 results. For such queries, we do not have access to the result documents after the first 1000. Thus in this cases we end up underestimating the weight of the query, and consequently, the corpus size.

3. Even though the search engine may return fewer than 1000 results for a given query, to promote diversity, it might restrict the number of results from a single host/domain. For the engines we tested this limit is set to two results per host/domain.

4. The number of samples used to estimate p_A is fairly small, since these samples were graciously provided by the authors of [BYG06] and their generation method is fairly laborious. These samples are uniformly random *over a large portion of the underlying corpus restricted to English documents*, namely those pages that match at least one query from the pool used in [BYG06]. Thus what we estimate using these samples, is the number of "visible" pages that match at least one query from the pool used in [BYG06], which is restricted to English only. This explain why these estimates are substantially lower than the second set of estimates, while the relative sizes are fairly consistent.

5. In the uncorrelated query pool approach, our choice of the query pool B has a natural bias against non-English documents. In other words, the pools A and B are presumably uncorrelated only with respect to the English portion of the corpus. Once again, this will result an underestimate of the actual corpus size. For instance the addition of one billion pages in Chinese that do not contain contiguous eight digit strings would be invisible to our approach.

2.7 Summary

In this chapter we addressed the problem of estimating the index size of a search engine given access to only public query interface. We constructed a basic estimator that can estimate the number of documents in the corpus that contain at least one term from a given query pool. Using this estimator, we compute the corpus size using two different algorithms, depending on whether or not it is possible to obtain random documents from the corpus. While the ability to randomly sample the corpus makes the problem easier, we show that by using two query pools that are reasonably uncorrelated, it is possible to obviate the need for random document samples. En route, we also obtain a novel way to provably reduce the variance of a random variable where the distribution of the random variable is monotonically decreasing; this technique may be of independent interest.

We applied our algorithms on the TREC collection to measure their performance. The algorithms that uses random document samples perform quite well as expected. More surprisingly, by carefully constructing query pools that are reasonably uncorrelated, we show that it possible to estimate the corpus size to modest accuracies. We also apply our algorithms to estimate the "visible" corpus size of major web search engines.

Chapter 3

Estimating Sum by Weighted Sampling

Estimating Web size or search engine index sizes can be mapped to the classic problem of estimating the sum of n non-negative variables, which has numerous important applications in various areas of computer science, statistics and engineering. Measuring the exact value of each variable incurs some cost, so we want to get a reasonable estimator of the sum while measure as few variables as possible.

In most traditional applications, only uniform sampling is allowed, i.e. each time we can sample one variable uniformly at random and ask its value. It is simple to estimate the sum from such samples: we can simply take the average of the samples as an estimator of the actual average, which is an unbiased estimator. It is easy to see that any reasonable estimator requires a linear sample size if the underlying distribution is arbitrary. Consider the following two instances of inputs: in the first input all variables are 0, while in the second input all are 0 except one variable x_1 is a large number. Any sampling scheme cannot distinguish the two inputs until it sees x_1, and with uniform sampling it takes linear samples to hit x_1. We defer the formal definition of "reasonable estimator" to Section 3.1, but intuitively we cannot get a good estimator if we cannot distinguish the two inputs.

We can map the problem of estimating index sizes or Web size to this problem by partitioning the search index (or the web) into domains (or web servers), with

each variable representing the size of a domain (or a web server), and estimating the size is simply estimating the sum of those variables. It is relatively easy to get the total domain (web server) number n (either by uniformly sampling IP space or this number is published periodically). For example in 1999 Lawrence and Giles estimated the number of web servers to be 2.8 million by randomly testing IP addresses; then they exhaustively crawled 2500 web servers and found that the mean number of pages per server was 289, leading to an estimate of the web size of 800 million [LG00]. Lawrence and Giles essentially used uniform sampling to estimate the sum, however, the domain size distribution is known to be highly skewed and uniform sampling has high variance for such inputs.

Now for our problem of estimating the Web size, a new sampling primitive becomes available: we can sample a variable with probability proportional to its value, which we refer to as *linear weighted sampling*. We can implement linear weighted sampling on domains as follows: uniformly sample a page from the web or a search engine index (uniform sampling of webpages is a nontrivial research problem by itself, and there has been extensive literature in the techniques of uniform sampling a page from the web [HHMN00, BYBC$^+$00, RPLG01] or a search engine index [BB98, BYG06]) and take the domain of the page; it is easy to see that the probability of sampling a domain is proportional to its size.

This new sampling primitive proposes interesting theoretical questions: how can we derive an estimation of the sum from such weighted samples? Does it improve sample complexity over uniform sampling? For the bad example of uniform sampling, just one sample from linear weighted sampling is sufficient to distinguish the two inputs, so it seems plausible that we can get good sum estimator with less samples using the new sampling method.

In this chapter we present an algorithm for sum estimation with $\tilde{O}(\sqrt{n})$ samples using only linear weighted sampling. Next, if we use both uniform sampling and linear weighted sampling, we can further reduce the number of samples to $\tilde{O}(\sqrt[3]{n})$ samples. Our algorithm assumes no prior knowledge about the input distribution. We also show the two estimators are almost optimal by proving lower bounds on the sample complexity of $\Omega(\sqrt{n})$ and $\Omega(\sqrt[3]{n})$ respectively.

Finally, we show a negative result that more general weighted sampling methods, where the probability of sampling a variable can be proportional to any function of its value, do not yield better estimators: we prove a lower bound of $\Omega(\sqrt[3]{n})$ samples for any reasonable sum estimator, using any combination of general weighted sampling methods. This implies that our sum estimator combining uniform and linear weighted sampling is almost optimal (up to a poly-log factor), hence there is no need to pursue fancier sampling methods for the purpose of estimating sum.

3.1 Definitions and Summary of Results

Let $x_1, x_2, \ldots, x_n$ be n variables. We consider the problem of estimating the sum $S = \sum_i x_i$, given n. We also refer to variables as *buckets* and the value of a variable as its *bucket size*.

In *(general) weighted sampling* we can sample a bucket x_i with probability proportional to a function of its size $f(x_i)$, where f is an arbitrary function of x_i (f independent on n). Two special cases are *uniform sampling* where each bucket is sampled uniformly at random ($f(x) = 1$), and *linear weighted sampling* where the probability of sampling a bucket is proportional to its size ($f(x) = x$). We assume sampling with replacement.

We say an algorithm is (ϵ, δ)-*estimator* ($0 < \epsilon, \delta < 1$), if it outputs an estimated sum S' such that with probability at least $1 - \delta$, $|S' - S| \leq \epsilon S$. The algorithm can take random samples of the buckets using some sampling method and learn the sizes as well as the labels of the sampled buckets. We measure the complexity of the algorithm by the total number of samples it takes. The algorithm has no prior knowledge of the bucket size distribution.

The power of the sum estimator is constrained by the sampling methods it is allowed to use. This paper studies the upper and lower bounds of the complexities of (ϵ, δ)-estimators under various sampling methods. As pointed out in Section 1, using only uniform sampling there is no (ϵ, δ)-estimator with sub-linear samples.

First we show an (ϵ, δ)-estimators using linear weighted sampling with $\tilde{O}(\sqrt{n})$ samples. While linear weighted sampling is a natural sampling method, to derive the

sum from such samples does not seem straightforward. Our scheme first converts the general problem to a special case where all buckets are either empty or of a fixed size; now the problem becomes estimating the number of non-empty buckets and we make use of birthday paradox by examining how many samples are needed to find a repeat. Each step involves some non-trivial construction and the detailed proof is presented in Section 3.3.

In Section 3.4 we consider sum estimators where both uniform and linear weighted sampling are allowed. Section 3.4.1 proposes an algorithm with $\tilde{O}(\sqrt[3]{n})$ samples which builds upon the linear weighted sampling algorithm in Section 3.3. Section 3.4.2 gives a different algorithm with $\tilde{O}(\sqrt{n})$ samples: although it is asymptotically worse than the former algorithm in terms of n, it has better dependency on ϵ and a much smaller hidden constant; this algorithm also is much neater and easier to implement.

Finally we present lower bounds in Section 3.5. We prove that the algorithms in Section 3.3 and 3.4.1 are almost optimal in terms of n up to a poly-log factor. More formally, we prove a lower bound of $\Omega(\sqrt{n})$ samples using only linear weighted sampling (more generally, using any combination of general weighted sampling methods with the constraint $f(0) = 0$); a lower bound of $\Omega(\sqrt[3]{n})$ samples using any combination of general weighted sampling methods.

All algorithms and bounds can be extended to the case where the number of buckets n is only approximately known (with relative error less than ϵ).

3.2 Related Work

Estimating the sum of n variables is a classical statistical problem. For the case where all the variables are between $[0, 1]$, an additive approximation of the mean can be easily computed by taking a random sample of size $O(\frac{1}{\epsilon^2} \lg \frac{1}{\delta})$ and computing the mean of samples; [CEG95] prove a tight lower bound on the sample size. However, uniform sampling works poorly on heavily tailed inputs when the variables are from a large range, and little is known beyond uniform sampling.

Weighted sampling is also known as "importance sampling". General methods of estimating a quantity using importance sampling have been studied in statistics

(see for example [Liu96]), but the methods are either not applicable here or less optimal. To estimate a quantity $h_\pi = \sum \pi(i)h(i)$, importance sampling generates independent samples $i_1, i_2, \ldots, i_N$ from a distribution p. One estimator for h_π is $\hat{\mu} = \frac{1}{N}\sum h(i_k)\pi(i_k)/p(i_k)$. For the sake of estimating sum, $\pi(i) = 1$ and $h(i)$ is the value of ith variable x_i. In linear weighted sampling, $p(i) = x_i/S$, where S is exactly the sum we are trying to estimate, therefore we are not able to compute this estimator $\hat{\mu}$ for sum. Another estimator is

$$\tilde{\mu} = \frac{\sum h(i_k)\pi(i_k)/\tilde{p}(i_k)}{\sum \pi(i_k)/\tilde{p}(i_k)},$$

where $\tilde{p}$ is identical to p up to normalization and thus computable. However, the variance of $\tilde{\mu}$ is even larger than the variance using uniform sampling.

A related topic is priority sampling and threshold sampling for estimating subset sums proposed and analyzed in [DLT05, ADLT05, Sze06]. But their cost model and application are quite different: they aim at building a sketch so that the sum of any subset can be computed (approximately) by only looking at the sketch; in particular their cost is defined as the size of the sketch and they can read all variables for free, so computing the total sum is trivial in their setting.

There is extensive work in estimating other frequency moments $F_k = \sum x_i^k$ (sum is the first moments F_1), in the random sampling model as well as in the streaming model (see for example [AMS99, CCMN00, BYKS01]). The connection between the two models is discussed in [BYKS01]. [BYKS01] also presents lower and upper bounds of the sample size on F_k for $k \geq 2$; note that their sampling primitive is different from ours, and they assume F_1 is known.

3.3 An $\tilde{O}(\sqrt{n})$ Estimator using Linear Weighted Sampling

Linear weighted sampling is a natural sampling method, but to efficiently derive the sum from such samples does not seem straightforward. Our algorithm first converts the general problem to a special case where all buckets are either empty or of a fixed

size, and then tackle the special case making use of the *birthday paradox*, which states that given a group of $\sqrt{365}$ randomly chosen people, there is a good chance that at least two of them have the same birthday.

Let us first consider the special case where all non-zero buckets are of equal sizes. Now linear weighted sampling is equivalent to uniform sampling among non-empty buckets, and our goal becomes estimating the number of non-empty buckets, denoted by B ($B \leq n$). We focus on a quantity we call "*birthday period*", which is the number of buckets sampled until we see a repeated bucket. We denote by $r(B)$ the birthday period of B buckets and its expected value $E[r(B)]$ is $\Theta(\sqrt{B})$ according to the birthday paradox. We will estimate the expected birthday period using linear weighted sampling, and then use it to infer the value of B. Most runs of birthday period take $O(\sqrt{B}) = O(\sqrt{n})$ samples, and we can cut off runs which take too long; $\lg \frac{1}{\delta}$ runs are needed to boost confidence, thus in total we need $O(\sqrt{n})$ samples to estimate B.

Now back to the general problem. We first guess the sum is an and fix a uniform bucket size ϵa. For each bucket in the original problem, we round its size down to $k\epsilon a$ (k being an integer) and break it into k buckets. If our guess of sum is (approximately) right, then the number of new buckets B is approximately n/ϵ; otherwise B is either too small or too large. We can estimate B by examining the birthday period as above using $O(\sqrt{n/\epsilon})$ samples, and check whether our guess is correct. Finally, since we allow a multiplicative error of ϵ, a logarithmic number of guesses suffice.

Before present the algorithm, we first establish some basic properties of birthday period $r(B)$. The following lemma bounds the expectation and variance of $r(B)$; property (3) shows that birthday period is "gap preserving" so that if the number of buckets is off by an ϵ factor, we will notice a difference of $c\epsilon$ in the birthday period. We can write out the exact formula for $E[r(B)]$ and $var(r(B))$, and the rest of the proof is merely algebraic manipulation.

Lemma 9. *(1) $E[r(B)]$ monotonically increases with B;*
(2) $E[r(B)] = \Theta(\sqrt{B})$;
(3) $E[r((1+\epsilon)B)] > (1+c\epsilon)E[r(B)]$, where c is a constant.
(4) $var(r(B)) = O(B)$;

Proof. (1) $r(B) > i$ when there is no repeated buckets in the first i samples.

$$Pr[r(B) > i] = \frac{B}{B}\frac{B-1}{B}\ldots\frac{B-(i-1)}{B} = (1-\frac{1}{B})\ldots(1-\frac{i-1}{B})$$

$$E[r(B)] = \sum_{1<i\leq B+1} Pr[r(B)=i]*i = \sum_{1<i\leq B+1} Pr[r(B)\geq i] = \sum_{1\leq i\leq B} Pr[r(B)>i]$$

$Pr[r(B) > i]$ monotonically increases with B for all i, so $E[r(B)]$ also monotonically increases with B.

(2) First bound $Pr[r(B) > i]$ using the fact $e^{-2x} < 1 - x < e^{-x}$:

$$Pr[r(B) > i] \leq e^{-\frac{1}{B}}e^{-\frac{2}{B}}\ldots e^{-\frac{i-1}{B}} = e^{-\frac{i(i-1)}{2B}}$$

$$Pr[r(B) > i] \geq e^{-\frac{2}{B}}e^{-\frac{4}{B}}\ldots e^{-\frac{2(i-1)}{B}} = e^{-\frac{i(i-1)}{B}}$$

Using the first inequality,

$$\begin{aligned}
E[r(B)] &= \sum_{1\leq i\leq B} Pr[r(B)>i] \leq \sum_{1\leq i\leq B} exp(-\frac{i(i-1)}{2B}) \\
&\leq \int_1^B exp(-\frac{i(i-1)}{2B})di \\
&= \sqrt{\frac{B\pi}{2}}exp(\frac{1}{8B})erf(\frac{2*B-1}{2\sqrt{2B}}) - \sqrt{\frac{B\pi}{2}}exp(\frac{1}{8B})erf(\frac{2*1-1}{2\sqrt{2B}}) \\
&\leq \sqrt{\frac{B\pi}{2}}exp(\frac{1}{8B}) = O(\sqrt{B})
\end{aligned}$$

Similarly, using the second inequality we can prove

$$E[r(B)] \geq \sum_{1\leq i\leq B} exp(-\frac{i(i-1)}{B} = \Omega(\sqrt{B})$$

Therefore $E[r(B)] = \Theta(\sqrt{B})$.

(3) Let $b_i = \frac{B-i}{B}$, $b'_i = \frac{(1+\epsilon)B-i}{(1+\epsilon)B}$; let $a_i = \prod_{j=1..i-1} b_j$, $a'_i = \prod_{j=1..i-1} b'_j$.

It is easy to see $E[r(B)] = \sum_{1\le i\le B} a_i$ and $E[r((1+\epsilon)B)] = \sum_{1\le i\le (1+\epsilon)B} a'_i$, therefore $E[r((1+\epsilon)B)] - E[r(B)] \ge \sum_{1\le i\le B} a'_i - a_i$. We will prove that $\Delta a_i = a'_i - a_i \ge c'\epsilon$ for all $i \in [\sqrt{B}, 2\sqrt{B}]$, which gives a lower bound on $E[r((1+\epsilon)B)] - E[r(B)]$.

Notice that $a_i = a_{i-1}b_{i-1} < a_{i-1}$. Let $\Delta b_i = b'_i - b_i = \frac{\epsilon i}{(1+\epsilon)B} > 0$.

For $i \in [\sqrt{B}, 2\sqrt{B}]$, $a'_i > a_i > exp(-\frac{i(i-1)}{B}) > e^{-4}$, therefore

$$\begin{aligned}
a'_i - a_i &= a'_{i-1}b'_{i-1} - a_{i-1}b_{i-1} = a_{i-1}(b'_{i-1} - b_{i-1}) + b'_{i-1}(a'_{i-1} - a_{i-1}) \\
&> a_{i-1}\Delta b_{i-1} + b_{i-1}\Delta a_{i-1} \\
&> a_{i-1}\Delta b_{i-1} + b_{i-1}(a_{i-2}\Delta b_{i-2} + b_{i-2}\Delta a_{i-2}) \\
&> a_i(\Delta b_{i-1} + \Delta b_{i-2}) + b_{i-1}b_{i-2}\Delta a_{i-2} \\
&\dots \\
&> a_i(\Delta b_{i-1} + \Delta b_{i-2} + \ldots + \Delta b_1) = a_i \frac{\epsilon}{(1+\epsilon)B} * \frac{i(i-1)}{2} \\
&> e^{-4}\frac{\epsilon}{2(1+\epsilon)} = \Theta(\epsilon)
\end{aligned}$$

Finally

$$E[r((1+\epsilon)B)] - E[r(B)] > \sum_{i\in[\sqrt{B},2\sqrt{B}]} \Delta a_i = \Theta(\epsilon\sqrt{B}) = \Theta(\epsilon)E[r(B)]$$

(4)

$$\begin{aligned}
var(r(B)) &= E[r(B)^2] - E[r(B)]^2 \le E[r(B)^2] \\
&= \sum_{2\le i\le B+1} Pr[r(B) = i]i^2 = \sum_{2\le i\le B+1} \frac{B}{B}\frac{B-1}{B}\cdots\frac{B-(i-2)}{B} * \frac{i-1}{B} * i^2 \\
&< \sum_{2\le i\le B+1} \frac{i^3}{B} exp(-\frac{(i-1)(i-2)}{2B}) \\
&\le (\frac{9}{16B}e^{-\frac{5}{2B}}\sqrt{2\pi B}(4B+3)erf(\frac{2x-3}{2\sqrt{2B}}) - \frac{1}{4}e^{-\frac{x^2-3x+2}{2B}}(4x^2+6x+8B+9))|_2^{B+1} \\
&= O(B)
\end{aligned}$$

□

BucketNumber(b, ϵ, δ)
1. Compute $r = E[r(b)]$;
2. for $i = 1$ to $k_1 = c_1 \lg \frac{1}{\delta}$
3. for $j = 1$ to $k_2 = c_2/\epsilon^2$
4. sample until see a repeated bucket; let r_j be the number of samples
5. if $\sum_{j=1}^{k_2} r_j/k_2 \leq (1 + c\epsilon/2)r$ then $s_i = true$, else $s_i = false$
6. if more than half of s_i are $true$ then output "$\leq b$ buckets"
else output "$\geq b(1+\epsilon)$ buckets"

Figure 3.1: Algorithm *BucketNumber*

Lemma 10 tackles the special case, stating that with $\sqrt{b}$ samples we can tell whether the total number of buckets is at most b or at least $b(1+\epsilon)$. The idea is to measure the birthday period and compare with the expected period in the two cases. We use the standard "median of the mean" trick: first get a constant correct probability using Chebyshev inequality, then boost the probability using Chernoff bound. See detail in the algorithm $BucketNumber$. Here c is the constant in Lemma 9; c_1 and c_2 are constants.

Lemma 10. *If each sample returns one of B buckets uniformly at random, then the algorithm BucketNumber tells whether $B \leq b$ or $B \geq b(1+\epsilon)$ correctly with probability at least $1 - \delta$; it uses $\Theta(\sqrt{b} \lg \frac{1}{\delta}/\epsilon^2)$ samples.*

Proof. We say the algorithm does k_1 runs, each run consisting of k_2 iterations. We first analyze the complexity of the algorithm. We need one small trick to avoid long runs: notice that we can cut off a run and set $s_i = false$ if we have already taken $(1 + c\epsilon/2)rk_2$ samples in this run. Therefore the total number of samples is at most

$$(1 + c\epsilon/2)rk_2k_1 = (1 + c\epsilon/2)E[r(b)]\frac{c_2}{\epsilon^2}c_1 \lg \frac{1}{\delta} = \Theta(\frac{\sqrt{b} \lg \frac{1}{\delta}}{\epsilon^2}).$$

The last equation uses Property (2) of Lemma 9.

Below we prove the correctness of the algorithm. Consider one of the k_1 runs. Let r' be the average of the k_2 measured birthday periods r_j. Because each measured period has mean $E[r(B)]$ and variance $var(r(B))$, we have $E[r'] = E[r(B)]$ and

$var(r') = var(r(B))/k_2$.

If $B \leq b$, then $E[r'] = E[r(B)] \leq r$. By Chebyshev inequality [MR95],

$$Pr[r' > (1+\frac{c\epsilon}{2})r] \leq Pr[r' > E[r(B)]+\frac{rc\epsilon}{2}] \leq \frac{var(r(B))/k_2}{(rc\epsilon/2)^2} \leq \frac{O(b)\epsilon^2/c_2}{(\Theta(\sqrt{b})c\epsilon/2)^2} = \frac{O(1)}{c_2}$$

If $B \geq b(1+\epsilon)$, then $E[r'] \geq E[r(b(1+\epsilon))] \geq (1+c\epsilon)r$ by Lemma 9.

$$Pr[r' < (1+\frac{c\epsilon}{2})r] \leq Pr[r' < (1-\frac{c\epsilon}{4})E[r']] \leq \frac{var(r(B))/k_2}{(E[r(B)]c\epsilon/4)^2} = \frac{O(1)}{c_2}$$

We choose the constant c_2 large enough such that both probabilities are no more than 1/3. Now when $B \leq b$, since $Pr[r' > (1+c\epsilon/2)r] \leq 1/3$, each run sets $s_i = false$ with probability at most 1/3. Our algorithm makes wrong judgement only if more than half of the k_1 runs set $s_i = false$, and by Chernoff bound [MR95], this probability is at most $e^{-c'k_1}$. Choose appropriate c_1 so that the error probability is at most δ. Similarly, when $B \geq (1+\epsilon)b$, each run sets $s_i = true$ with probability at most 1/3, and the error probability of the algorithm is at most δ. □

Algorithm *LWSE* (stands for *Linear Weighted Sampling Estimator*) shows how to estimates sum for the general case. The labelling in step 3 is equivalent to the following process: for each original bucket, round its size down to a multiple of $\epsilon_1 a$ and split into several "standard" buckets each of size $\epsilon_1 a$; each time sampling returns a standard bucket uniformly at random. The two processes are equivalent because they have the same number of distinct labels (standard buckets) and each sampling returns a label uniformly at random. Therefore by calling $BucketNumber(n(1+\epsilon_1)/\epsilon_1, \epsilon_1, \delta_1)$ with such samples, we can check whether the number of standard buckets $B \leq n(1+\epsilon_1)/\epsilon_1$ or $B \geq n(1+\epsilon_1)^2/\epsilon_1$, allowing an error probability of δ_1.

Theorem 11. *LWSE is an (ϵ, δ)-estimator with $O(\sqrt{n}(\frac{1}{\epsilon})^{\frac{7}{2}} \log n(\log \frac{1}{\delta} + \log \frac{1}{\epsilon} + \log\log n))$ samples, where n is the number of buckets.*

Proof. We first show that the algorithm terminates with probability at least $1-\delta_1$. S must fall in $[a_0 n, a_0 n(1+\epsilon_1)]$ for some a_0, and we claim that the algorithm will

LWSE(n, ϵ, δ)

1. get a lower bound L of the sum: sample one bucket using linear weighted sampling and let L be the size of the sampled bucket;
2. for $a = L/n, L(1+\epsilon_1)/n, \ldots, L(1+\epsilon_1)^k/n, \ldots$ (let $\epsilon_1 = \epsilon/3$)
3. for each sample returned by linear weighted sampling, create a label as follows: suppose a bucket x_i of size s is sampled and $s = m\epsilon_1 a + r$ (m is an integer and $r < \epsilon_1 a$); discard the sample with probability r/s; with the remaining probability generate a number l from $1..m$ uniformly at random and label the sample as i_l;
4. call $BucketNumber(n(1+\epsilon_1)/\epsilon_1, \epsilon_1, \delta_1)$, using the above samples in step 4 of $BucketNumber$. If $BucketNumber$ outputs "$\leq n(1+\epsilon_1)/\epsilon_1$", then output $S' = an$ as the estimated sum and terminate.

Figure 3.2: Algorithm *LWSE*

terminate at this a_0, if not before: since $S \leq a_0 n(1+\epsilon_1)$, the sum after rounding down is at most $a_0 n(1+\epsilon_1)$ and hence the number of labels $B \leq n(1+\epsilon_1)/\epsilon_1$; by Lemma 10 it will pass the check with probability at least $1-\delta_1$ and terminate the algorithm.

Next we show that given that $LWSE$ terminates by a_0, the estimated sum is within $(1\pm\epsilon)S$ with probability $1-\delta_1$. Since the algorithm has terminated by a_0, the estimated sum cannot be larger than S, so the only error case is $S' = an < (1-\epsilon)S$. The sum loses at most $na\epsilon_1$ after rounding down, so

$$B \geq \frac{S - an\epsilon_1}{a\epsilon_1} \geq \frac{\frac{an}{1-\epsilon} - an\epsilon_1}{a\epsilon_1} = \frac{n}{(1-\epsilon)\epsilon_1} - n \geq n\frac{1-\epsilon_1}{(1-\epsilon)\epsilon_1} \geq n\frac{(1+\epsilon_1)^2}{\epsilon_1}$$

The probability that it can pass the check for a fixed $a < a_0$ is at most δ_1; by union bound, the probability that it passes the check for any $a < a_0$ is at most $\delta_1 \log_{1+\epsilon} \frac{S}{L}$. Combining the two errors, the total error probability is at most $\delta_1(\log_{1+\epsilon} \frac{S}{L} + 1)$. Choose $\delta_1 = \delta/(\log_{1+\epsilon} \frac{S}{L} + 1)$, then with probability at least $1-\delta$ the estimator outputs an estimated sum within $(1\pm\epsilon)S$.

Now we analyze the complexity of $LWSE$. Ignore the discarded samples for now

and count the number of valid samples. By Lemma 10, for each a we need

$$N_1 = O(\frac{\log \frac{1}{\delta_1} * \sqrt{\frac{n(1+\epsilon_1)}{\epsilon_1}}}{\epsilon_1^2}) = O(\sqrt{n}(\frac{1}{\epsilon})^{\frac{5}{2}}(\log \frac{1}{\delta} + \log \frac{1}{\epsilon} + \log\log \frac{S}{L}))$$

samples, and there are $\log_{1+\epsilon} \frac{S}{L} = O(\log \frac{S}{L}/\epsilon)$ as. As for the discarded samples, the total discarded size is at most $an\epsilon_1$, and we always have $S \geq an$ if the algorithm is running correctly, therefore the expected probability of discarded samples is at most $\epsilon_1 = \epsilon/3 \leq 1/3$. By Chernoff bound, with high probability the observed probability of discarded samples is at most half, i.e. the discarded samples at most add a constant factor to the total sample number.

Finally, the complexity of the estimator has the term $\log \frac{S}{L}$. Had we simply started guessing from $L = 1$, the cost would depend on $\log S$. The algorithm chooses L to be the size of a sampled bucket using linear weighted sampling. We claim that with high probability $L \geq S/n^2$: otherwise $L < S/n^2$, then the probability that linear weighted sampling returns any bucket of size no more than L is at most $n * L/S < 1/n$.

Summing up, the total sample number used in $LWSE$ is

$$N_1 * O(\frac{\log n^2}{\epsilon}) = O(\sqrt{n}(\frac{1}{\epsilon})^{\frac{7}{2}} \log n(\log \frac{1}{\delta} + \log \frac{1}{\epsilon} + \log\log n)).\square$$

□

3.4 Combining Uniform and Linear Weighted Sampling

In this section we design sum estimator using both uniform sampling and linear weighted sampling. We present two algorithms. The first uses $LWSE$ in Section 3.3 as a building block and only needs $\tilde{O}(\sqrt[3]{n})$ samples. The second algorithm is self-contained and easier to implement; its complexity is worse than the first algorithm in terms of n but has better dependency on ϵ and a much smaller hidden constant.

CombEst(n, ϵ, δ)

1. find t such that the number of buckets whose size is larger than t is $N_t = \Theta(n^{2/3})$ (we leave the detail of this step later); call a bucket *large* if its size is above t, and *small* otherwise
2. use weighted linear sampling to estimate the total size of large buckets S'_{large}:
   if the fraction of large buckets in the sample is less than $\epsilon_1/2$, let $S'_{large} = 0$;
   otherwise ignore small buckets in the samples and estimate S'_{large} using $LWSE(N_t, \epsilon_1, \delta/2)$, where $\epsilon_1 = \epsilon/4$
3. use uniform sampling to estimate the total size of small buckets S'_{small}:
   divide the small bucket sizes into levels $[1, 1+\epsilon_1), \ldots, [(1+\epsilon_1)^i, (1+\epsilon_1)^{i+1}), \ldots, [(1+\epsilon_1)^{i_0}, t)$; we say a bucket in level i $(0 \le i \le i_0)$ if its size $\in [(1+\epsilon_1)^i, (1+\epsilon_1)^{i+1})$
   make $k = \Theta(n^{1/3} \log n/\epsilon_1^4)$ samples using uniform sampling; let k_i be the number of sampled buckets in level i. Estimate the total number of buckets in level i to be $n'_i = k_i n/k$ and $S'_{small} = \sum_i n'_i(1+\epsilon_1)^i$
4. output $S'_{small} + S'_{large}$ as the estimated sum

Figure 3.3: Algorithm *CombEst*

3.4.1 An Estimator with $\tilde{O}(\sqrt[3]{n})$ Samples

In this algorithm, we split the buckets into two types: $\Theta(\sqrt[3]{n^2})$ *large* buckets and the remaining *small* buckets. We estimate the partial sum of the large buckets using linear weighted sampling as in Section 3.3; we stratify the small buckets into different size ranges and estimate the number of buckets in each range using uniform sampling.

Theorem 12. *CombEst is an (ϵ, δ)-estimator with $O(n^{1/3}(\frac{1}{\epsilon})^{\frac{9}{2}} \log n(\log \frac{1}{\delta} + \log \frac{1}{\epsilon} + \log\log n))$ samples, where n is the number of buckets.*

Proof. We analyze the error in the estimated sum. Denote by $S_{large}(S_{small})$ the actual size of large (small) buckets; by n_i the actual bucket number in level i.

In Step 2, since we are using linear weighted sampling, the expected fraction of large buckets in the samples equals to S_{large}/S. If $S_{large}/S > \epsilon_1$, then by Chernoff bound the observed fraction of large buckets in the sample is larger than $\epsilon_1/2$ with

high probability, and we will get S'_{large} within $(1 \pm \epsilon_1)S_{large}$ with probability at least $1-\delta/2$ according to Theorem 11; otherwise we lose at most $S_{large} = \epsilon_1 S$ by estimating $S'_{large} = 0$. Thus, with probability at least $1 - \delta/2$, the error introduced in Step 2 is at most $\epsilon_1 S$.

In Step 3, it is easy to see that n'_i is an unbiased estimator of n_i. For a fixed i, if $n_i \geq \epsilon_1^2 n^{2/3}$ then by Chernoff bound the probability that n'_i deviates from n_i by more than an ϵ_1 fraction is

$$Pr[|n'_i - n_i| \geq \epsilon_1 n_i] \leq exp(-ck\epsilon_1^2 n_i/n) \leq exp(-c'\frac{n^{1/3}\log n}{\epsilon_1^4}\epsilon_1^2\frac{\epsilon_1^2 n^{2/3}}{n}) = n^{-c'}$$

This means that for all $n_i \geq \epsilon_1 n^{2/3}$, with high probability we estimate n_i almost correctly, introducing a relative error of at most ϵ_1.

We round all bucket sizes of small buckets down to the closest power of $1 + \epsilon_1$; this rounding introduces a relative error of at most ϵ_1.

For all levels with $n_i < \epsilon_1^2 n^{2/3}$, the total bucket size in those levels is at most

$$\sum_{0 \leq i \leq i_0} n_i(1+\epsilon_1)^{i+1} < \epsilon_1^2 n^{2/3} \sum_i (1+\epsilon_1)^{i+1} < \epsilon_1^2 n^{2/3}\frac{t}{\epsilon_1} = \epsilon_1 t n^{2/3} < \epsilon_1 S_{large} < \epsilon_1 S$$

The error introduced by those levels adds up to at most ϵ_1.

Summing up, there are four types of errors in our estimated sum, with probability at least $1 - \delta$ each contributing at most $\epsilon_1 S = \epsilon S/4$, so S' has an error of at most ϵS.

Now we count the total number of samples in $CombEst$. According to Theorem 11, Step 2 needs $O(\sqrt{n^{2/3}}(\frac{1}{\epsilon})^{\frac{7}{2}}\log n^{2/3}(\log\frac{1}{\delta} + \log\frac{1}{\epsilon} + \log\log n^{2/3}))$ samples of large buckets, and by our algorithm the fraction of large buckets is at least $\epsilon_1/2$. Step 3 needs $\Theta(n^{1/3}\log n/\epsilon_1^4)$ samples, which is dominated by the sample number of Step 2. Therefore the total sample number is

$$O(n^{1/3}(\frac{1}{\epsilon})^{\frac{9}{2}}\log n(\log\frac{1}{\delta} + \log\frac{1}{\epsilon} + \log\log n)).\square$$

□

There remains to be addressed the implementation of Step 1. We make $n^{1/3}\log n$

samples using uniform sampling and let t be the size of the $2\log n$-th largest bucket in the samples. Let us first assume all the sampled bucket have different sizes. Let N_t be the number of buckets with size at least t; we claim that with high probability $n^{2/3} \leq N_t \leq 4n^{2/3}$. Otherwise if $N_t < n^{2/3}$, then the probability of sampling a bucket larger than t is $N_t/n < n^{-1/3}$ and the expected number of such buckets in the samples is at most $\log n$; now we have observed $2\log n$ such buckets, by Chernoff bound the probability of such event is negligible. Similarly the probability that $N_t \geq 4n^{2/3}$ is negligible. Hence t satisfies our requirement. Now if there is a tie at position $2\log n$, we may cut off at any position $c\log n$ instead of $2\log n$, and N_t will still be $\Theta(n^{2/3})$ using the same argument. In the worst case where all of them are ties, let t be this size, define those buckets with sizes strictly larger than t as large buckets and those with sizes strictly less than t as small, estimating S_{large} and S_{small} using Steps 2 and 3; estimate separately the number of buckets with size exactly t using uniform sampling – since the number is at least $\Theta(n^{2/3}\log n)$, $O(n^{1/3})$ samples are sufficient. Finally we only know the approximate number of large buckets, denoted by N_t', and have to pass N_t' instead of N_t when call $LWSE$. Fortunately an approximate count of n suffices for $LWSE$, and a constant factor error in n only adds a constant factor in its complexity.

3.4.2 An Estimator with $\tilde{O}(\sqrt{n})$ Samples

Next we present a sum estimator using uniform and weighted sampling with $\tilde{O}(\sqrt{n})$ samples. Recall that uniform sampling works poorly for skewed distributions, especially when there are a few large buckets that we cannot afford to miss. The idea of this algorithm is to use weighted sampling to deal with such heavy tails: if a bucket is large enough it will keep appearing in weighted sampling; after enough samples we can get a fairly accurate estimate of its frequency of being sampled, and then infer the total size by only looking at the size and sampling frequency of this bucket. On the other hand, if no such large bucket exists, the variance cannot be too large and uniform sampling performs well.

Theorem 13. *CombEstSimple is an (ϵ, δ)-estimator with $O(\sqrt{n}/\epsilon^2\delta)$ samples.*

CombEstSimple(n, ϵ, δ)

1. Make $k = c_1\sqrt{n}\log\frac{1}{\delta}/\epsilon^2$ samples using weighted linear sampling. Suppose the most frequently sampled bucket has size t and is sampled k_1 times (breaking ties arbitrarily). If $k_1 \geq k/2\sqrt{n}$, output $S' = tk/k_1$ as estimated sum and terminate
2. Make $l = \sqrt{n}/\delta\epsilon^2$ samples using uniform sampling and let a be the average of sampled bucket sizes. Output $S' = an$ as estimated sum

Figure 3.4: Algorithm *CombEstSimple*

Proof. Obviously *CombEstSimple* uses $k + l = O(\sqrt{n}/\epsilon^2\delta)$ samples. Below we prove the accuracy of the estimator.

We first prove that if Step 1 outputs an estimated sum S', then S' is within $(1\pm\epsilon)S$ with probability $1 - \delta/2$. Consider any bucket with size t whose frequency of being sampled $f' = k_1/k$ is more than $1/2\sqrt{n}$. Its expected frequency of being sampled is $f = t/S$, so we can bound the error $|f' - f|$ using Chernoff bound.

$$Pr[f - f' > \epsilon f] \leq exp(-ckf\epsilon^2) \leq exp(-ckf'\epsilon^2) = exp(\Theta(c1)\log\frac{1}{\delta}) = \delta^{\Theta(c1)}$$

$$Pr[f' - f > \epsilon f] \leq exp(-ckf\epsilon^2) \leq exp(-ck\frac{f'\epsilon^2}{1+\epsilon}) = exp(\Theta(c1)\log\frac{1}{\delta}) = \delta^{\Theta(c1)}$$

Choose c_1 large enough to make $Pr[|f - f'| > \epsilon f]$ less than $\delta/2$, then with probability $1 - \delta/2$, $f' = k_1/k$ is within $(1 \pm \epsilon)t/S$, and it follows that the estimated sum tk/k_1 is within $(1 \pm \epsilon)S$.

We divide the input into two cases, and show that in both cases the estimated sum is close to S.

Case 1, the maximum bucket size is greater than $S/\sqrt{n}$. The probability that the largest bucket is sampled less than $k/2\sqrt{n}$ times is at most $exp(-ck\frac{1}{\sqrt{n}}) < \delta/2$; with the remaining probability, Step 1 outputs an estimated sum, and we have proved it is within $(1 \pm \epsilon)S$.

Case 2, the maximum bucket size is no more than $S/\sqrt{n}$. If Step 1 outputs an

estimated sum, we have proved it is close to S. Otherwise we use the estimator in Step 2. a is an unbiased estimator of the mean bucket size. The statistical variance of x_i is

$$var(x) \leq E[x^2] = \frac{\sum_i x_i^2}{n} \leq \frac{(\frac{S}{\sqrt{n}})^2\sqrt{n}}{n} = \frac{S^2}{n\sqrt{n}}$$

and the variance of a is $var(x)/l$. Using Chebyshev inequality, the probability that a deviates from the actual average S/n by more than an ϵ fraction is at most $var(a)/(\epsilon S/n)^2 = \sqrt{n}/l\epsilon^2 = \delta$. □

3.5 Lower Bounds

Finally we prove some lower bounds on the sample number of sum estimators. Those lower bound results use a special type of input instances where all bucket sizes are either 0 or 1. The results still hold if all bucket sizes are strictly positive, using similar counterexamples with bucket sizes either 1 or a large constant b.

Theorem 14. *There exists no (ϵ, δ)-estimator with $o(\sqrt{n})$ samples using only linear weighted sampling, for any $0 < \epsilon, \delta < 1$.*

Proof. Consider two instances of inputs: in one input all buckets have size 1; in the other, $(1-\epsilon)n/(1+\epsilon)$ buckets have size 1 and the remaining are empty. If we cannot distinguish the two inputs, then the estimated sum deviates from the actual sum by more than an ϵ fraction.

For those two instances, linear weighted sampling is equivalent to uniform sampling among non-empty buckets. If we sample $k = o(\sqrt{n})$ buckets, then the probability that we see a repeated bucket is less than $1 - exp(-k(k-1)/((1-\epsilon)n/(1+\epsilon))) = o(1)$ (see the proof of Lemma 9). Thus in both cases with high probability we see all distinct buckets of the same sizes, so cannot distinguish the two inputs in $o(\sqrt{n})$ samples. □

More generally, there is no estimator with $o(\sqrt{n})$ samples using any combination of general weighted sampling methods with the constraint $f(0) = 0$. Recall that weighted sampling with function f samples a bucket x_i with probability proportional

to a function of its size $f(x_i)$. When $f(0) = 0$, it samples any empty bucket with probability 0 and any bucket of size 1 with the same probability, thus is equivalent to linear weighted sampling for the above counterexample.

Theorem 15. *There exists no (ϵ, δ)-estimator with $o(\sqrt[3]{n})$ samples using any combination of general weighted sampling (the sampling function f independent on n), for any $0 < \epsilon, \delta < 1$.*

Proof. Consider two instances of inputs: in one input $n^{2/3}$ buckets have size 1 and the remaining buckets are empty; in the other, $3n^{2/3}$ buckets have size 1 and the remaining are empty. If we cannot distinguish the two inputs, then the estimated sum deviates from the actual sum by more than $\frac{1}{2}$. We can adjust the constant to prove for any constant ϵ.

We divide weighted sampling into two types:

(1) $f(0) = 0$. It samples any empty bucket with probability 0 and any bucket of size 1 with the same probability, thus it is equivalent to uniform sampling among non-empty buckets. There are at least $n^{2/3}$ non-empty buckets and we only make $o(n^{1/3})$ samples, with high probability we see $o(n^{1/3})$ distinct buckets of size 1 for both inputs.

(2) $f(0) > 0$. The probability that we sample any non-empty buckets is

$$\frac{f(1)cn^{2/3}}{f(1)cn^{2/3} + f(0)(n - cn^{2/3})} = \Theta(n^{-1/3}),$$

so in $o(n^{1/3})$ samples with high probability we only see empty buckets for both inputs, and all these buckets are distinct.

Therefore whatever f we choose, we see the same sampling results for both inputs in the first $o(n^{1/3})$ samples, i.e. we cannot distinguish the two inputs with $o(n^{1/3})$ samples using any combination of weighted sampling methods. □

Part II

Webgraph Models

Chapter 4

Searchability in Random Graphs

Since Milgram's famous "small world" experiment [Mil67], it has generally been understood that social networks have the property that a typical node can reach any other node through a short path (the so-called "six degrees of separation"). An implication of this fact is that social networks have small diameter. Many random graph models have been proposed to explain this phenomenon, often by showing that adding a small number of random edges causes a highly structured graph to have a small diameter (e.g., [WS98, BC88]). A stronger implication of Milgram's experiment, as Kleinberg observed [Kle00], is that for most social networks there are decentralized search algorithms that can *find* a short path from a source to a destination without a global knowledge of the graph. As Kleinberg proved, many of the random graph models with small diameter do not have this property (i.e., any decentralized search algorithm in such graphs can take many steps to reach the destination), while in certain graph models with a delicate balance of parameters, decentralized search is possible. Since Kleinberg's work, there have been many other models that provably exhibit the searchability property [KLNT06, Fra05, Sli05, LNNK$^+$05, Kle01, DHLS06]; however, we still lack a good understanding of what contributes to this property in graphs.

In this chapter, we look at a general framework for searchability in random graphs. We consider a general random graph model in which the set of edges leaving a node u is independent of that of any other node $v \neq u$. This framework includes many existing random graph models such as those proposed in [CL03, ACL00, Kle00, Kle01,

LCKF05, ACK+07]. It is worth noting that, in a random graph model where edges can have arbitrary dependencies, the search problem includes arbitrarily difficult learning problems as special cases, and therefore one cannot expect to have a complete characterization of searchable graphs in such a model.

Throughout most of this chapter, we restrict the class of decentralized search algorithms that we consider to deterministic memoryless algorithms that succeed in finding a path to the destination with probability 1. This is an important class of search algorithms, and includes the decentralized search algorithms used in Kleinberg's work on long-range percolation graphs and hierarchical network models. For this class, we give a simple characterization of graphs that are searchable in terms of a node ordering property (Theorem 16). Then we will use this characterization to show a monotonicity property for searchability: if a graph is searchable in our model, it stays searchable if the probabilities of edges are increased (Theorem 21).

The rest of this chapter is organized as follows: Section 4.1 contains the description of the model. Section 4.2 presents a characterization of searchable random graphs. The monotonicity theorem is presented in Section 4.3. We conclude in Section 4.4 with a discussion of open problems.

4.1 The Model

We define a random graph model parameterized by a a positive integer n (the size of the graph) and n independent distributions $\Omega_1, \Omega_2, \ldots, \Omega_n$. For each $i \in \{1, \ldots, n\}$, Ω_i is a distribution over the collection of all subsets of $\{1, \ldots, n\}$. The random digraph $G(n, \Omega)$ is defined as follows: the vertex set of this graph is $V = \{1, \ldots, n\}$, and for every i, the set of vertices that have an edge from i (i.e., the *out-neighbors* of i) is picked (independently) from the distribution Ω_i. For $i \in V$, let $\Gamma(i)$ denote the set of out-neighbors of i. We denote by $\omega_{i,S}$ the probability that $\Gamma(i) = S$.

This graph model is quite general. It includes models such as the directed variant of the classical Erdős–Rényi graphs [ER59], random graphs with a given expected degree sequence (e.g., [CL03]), ACL graphs [ACL00], long-range percolation graphs [Kle00], hierarchical network models [Kle01], and graphs based on Kronecker

products [LCKF05, ACK+07], but not models such as preferential attachment [BA99] in which the distribution of edges leaving a node is dependent on the other edges of the graph. For example, for the long-range percolation graphs [Kle00], the support of the distribution Ω_i is all possible subsets consisting of all vertices with Hamming distance 1 to i in the lattice and one extra vertex j (denoting this subset by S_j), and the probability of S_j equals to the probability that the long range edge from i lands at j.

A special case of this model that deserves special attention is when all edges of the graph are independent. In this case, given a positive integer n and an $n \times n$ matrix $\mathbf{P}$ with entries $p_{i,j} \in [0, 1]$, we define a directed random graph $G(n, \mathbf{P})$ with the node set $V = \{1, \ldots, n\}$ and with a directed edge connecting node i to node j with probability p_{ij}, independently of all other edges. Note that this is a special case of the $G(n, \Omega)$ random graph model, with $\omega_{i,S} := \prod_{j \in S} p_{ij} \prod_{j \notin S} (1 - p_{ij})$.

We fix two nodes $s, t \in V$ of $G(n, \Omega)$ as the *source* and the *destination*. We investigate the existence of a decentralized search algorithm that finds a path from s to t of at most a given length d in expectation.[1] We restrict our attention to *deterministic memoryless algorithms*. A deterministic memoryless algorithm can be defined as a partial function $A : V \times 2^V \to V$. Such an algorithm A defines a path $v_0, v_1, v_2, \ldots$ on a given graph G as follows: $v_0 = s$, and for every $i \geq 0$, $v_{i+1} = A(v_i, \Gamma(v_i))$. The length of this path is defined as the smallest integer i such that $v_i = t$. If no such i exists, we define the length of the path as infinity.

We are now ready to define the notion of searchability. For a given (n, Ω), source and destination nodes s and t, and a number d, we say that $G(n, \Omega)$ is *d-searchable* using a deterministic memoryless algorithm A if the expected length of the path defined by A on $G(n, \Omega)$ is at most d. Note that this definition requires the algorithm to find a path from s to t with probability 1.

[1] Alternatively, we could ask for which graphs a decentralized search algorithm can find a path between *every* pair of nodes s and t, or between a *random* pair of nodes s and t. Our techniques apply to these alternative formulations of the problem as well. The only point that requires some care is that the orderings in the characterization theorem can depend on s and t.

4.2 A Characterization of Searchable Random Graphs

In this section, we provide a complete characterization of searchable random graphs. We begin by defining a class of deterministic memoryless search algorithms parameterized by two orderings of V, and then prove that if a graph is d-searchable, it is also d-searchable using an algorithm from this narrow class.

Definition 1. *Let σ, π be two orderings (i.e., permutations) of the node set V. We define a deterministic memoryless algorithm $A_{\sigma,\pi}$ corresponding to these orderings as follows: for every $u \in V$, $A_{\sigma,\pi}(u, \Gamma(u))$ is defined as the maximum element according to π of the set $\{v \in \Gamma(u) : \sigma(v) > \sigma(u)\}$.*

In other words, algorithm $A_{\sigma,\pi}$ never goes backwards according to the ordering σ, and, subject to this restriction, makes the maximum possible progress according to π.

Before stating our main result, we comment on why the class of search algorithms we are considering is defined based on two permutations and not just one. Common intuition based on the known results (e.g., on the long-range percolation model [Kle00], or the hierarchical network models [Kle01]) might lead one to conjecture that it is enough to consider decentralized search algorithms that always try to get as close to the destination as possible according to a *single* ordering of the vertices. This, however, is not true, as the following simple example shows.

Example 1. *Consider a graph with the vertex set $s = u_1, u_2, \ldots, u_n = t$. For every $i, j \in \{1, \ldots, n/2\}$, $i \neq j$, there is edge from i to j with probability 1. For $i = n/2, \ldots, n-1$, there is an edge from u_i to u_{i+1} with probability 1. Finally, for $i = 2, \ldots, n/2-1$, there is an edge from u_i to t with probability $1/2$. One can find a path in this graph using a deterministic memoryless search algorithm as follows: the algorithm traverses the path $su_2u_3 \ldots u_{n/2-1}$ in this order until it finds the first vertex that has a direct edge to t. If it finds such a vertex, it goes to t; otherwise, it takes the path $u_{n/2}u_{n/2+1} \ldots u_n$ to t. The expected length of this path is $2+\frac{1}{2}+\frac{1}{4}+\cdots+\frac{1}{2^{n/2-1}}+\frac{n}{2^{n/2}} < 4$. However, a memoryless algorithm that is based on a single ordering cannot visit more than one of the vertices in the set $S = \{u_2, \ldots, u_{n/2-1}\}$. This is because if the first*

vertex in S that the algorithm visits is u_i, u_i must be ahead of all other vertices of S in the ordering. Hence, if the algorithm visits another $u_j \in S$, as its next step it must go back to u_i (since it has the exact same set of choices as it had in the step it first visited u_i), and therefore it will fall into a loop. Therefore, such an algorithm cannot achieve an expected distance smaller than $n/4$.

We are now ready to state our main theorem which characterizes random graphs that are d-searchable.

Theorem 16. *(***Characterization of d-Searchable Random Graphs***) For a given n, collection of out-neighbor distributions Ω, source and destination nodes s and t, and number d, if $G(n, \Omega)$ is d-searchable using a deterministic memoryless algorithm A, then there exist two orderings σ and π of V such that $G(n, \Omega)$ is d-searchable using $A_{\sigma,\pi}$.*

To prove this theorem, we first construct the ordering σ using the structure of the search algorithm A. Next, we define an ordering π using σ. Finally, we use induction with respect to the ordering σ to show that the expected length of the path defined by $A_{\sigma,\pi}$ on $G(n, \Omega)$ is not more than the one defined by A.

We assume, without loss of generality, that for every set $S \subseteq V$, $A(t, S) = t$. In other words, we assume that A never leaves t once it reaches this node.

Define a graph H with the node set V as follows: for every pair $u, v \in V$, the edge (u, v) is in H if and only if this edge is on the path from s to t defined by A on some realization of $G(n, \Omega)$ (i.e., on some graph that has a non-zero probability in the distribution $G(n, \Omega)$). We have the following important lemma.

Lemma 17. *The graph H is acyclic.*

Proof. Assume, for contradiction, that H contains a simple cycle C. Note that by the definition of H, if an edge (u, v) is in H, then u must be reachable from s in H. Therefore, every node of C must be reachable from s in H. Let v^* be a node in C that has the shortest distance from s in H, and $s = v_0, v_1, \ldots, v_\ell = v^*$ be a shortest path from s to v^* in H. Also, let $v^* = v_\ell, v_{\ell+1}, \ldots, v_k, v_{k+1} = v^*$ denote the cycle C.

Therefore, $v_0, v_1, \ldots, v_k$ are all distinct nodes, and for every $i \in \{0, \ldots, k\}$, there is an edge from v_i to v_{i+1} in H.

By the definition of H, for every $i \in \{0, \ldots, k\}$, there is a realization of $G(n, \Omega)$ in which A traverses the edge (v_i, v_{i+1}). This means that there is a realization of $G(n, \Omega)$ in which the set $\Gamma(v_i)$ of out-neighbors of v_i is S_i^*, for some set S_i^* such that $A(v_i, S_i^*) = v_{i+1}$. Recall that by the definition of $G(n, \Omega)$, the random variables $\Gamma(u)$ are all independent. Hence, since v_i's are all distinct and for each i, there is a realization satisfying $\Gamma(v_i) = S_i^*$, there must be a realization in which $\Gamma(v_i) = S_i^*$ for *all* i. In this realization, the algorithm A falls in the cycle C, and therefore will never reach t. Thus the path found by A in this realization is infinitely long, and therefore the expected length of the path found by A is infinite. This is a contradiction. □

By Lemma 17, we can find a topological ordering of the graph H. Furthermore, since by assumption t has no outgoing edge in H, we can find a topological ordering that places t last. Let σ be such an ordering; more precisely, σ is an ordering of V such that

(i) t is the maximum element of V under σ;

(ii) for every edge (u, v) in H, we have $\sigma(v) > \sigma(u)$; and

(iii) all isolated nodes of H are placed at the beginning of σ in an arbitrary order, i.e., $\sigma(u) > \sigma(v)$ for any isolated node v and non-isolated node u.

By the definition of H, these conditions mean that the algorithm A (starting from the node s) never traverses an edge (u, v) with $\sigma(u) > \sigma(v)$.

Given the ordering σ, we define numbers r_u for every $u \in V$ to be the expected time to reach t from u following the best path that does not backtrack with respect to σ. r_u can be computed recursively as follows:

$$r_u = \begin{cases} 0 & \text{if } u = t \\ 1 + \displaystyle\sum_{S \subseteq T_u, S \neq \emptyset} q_{u,S} \cdot \min_{v \in S}\{r_v\} & \text{if } u \neq t \text{ and } q_{u,\emptyset} = 0 \\ \infty & \text{if } u \neq t \text{ and } q_{u,\emptyset} > 0, \end{cases} \tag{4.1}$$

where $T_u := \{v : \sigma(v) > \sigma(u)\}$ and, for a set $S \subseteq T_u$, we write

$$q_{u,S} := \sum_{S' : S' \cap T_u = S} \omega_{u,S'}$$

to denote the probability that the subset of nodes of T_u that are out-neighbors of u is precisely S.[2] Note that the above formula defines r_u in terms of r_v for $\sigma(v) > \sigma(u)$, and therefore the definition is well founded.

We can now define the ordering π as follows: let $\pi(u) > \pi(v)$ if $r_u < r_v$. Pairs u, v with $r_u = r_v$ are ordered arbitrarily by π.

The final step of the proof is the following lemma, which we will prove by induction using the ordering σ. To state the lemma, we need a few pieces of notation. For a search algorithm B, let $d(B, u)$ denote the expected length of the path that the algorithm B, started at node u, finds to t. Also, let V_0 denote the set of non-isolated nodes of H—i.e., V_0 is the set of nodes that the algorithm A (started from s) has a non-zero chance of reaching.

Lemma 18. *Let σ and π be the orderings defined as above. Then for every node $u \in V_0$, we have that $d(A, u) \geq d(A_{\sigma,\pi}, u) = r_u$.*

Proof. We prove this statement by induction on u, according to the ordering σ. The statement is trivial for $u = t$. We now show that for $u \in V_0 \setminus \{t\}$ if the statement holds for every node $v \in V_0$ with $\sigma(v) > \sigma(u)$ (i.e., for every $v \in T_u \cap V_0$), then it also holds for u. Observe that for any deterministic memoryless algorithm B,

$$d(B, u) = 1 + \sum_{S \subset V, S \neq \emptyset} \omega_{u,S} \cdot d(B, B(u, S)). \tag{4.2}$$

This statement follows from the fact that the algorithm B is memoryless, and that

[2] In the special case of $G(n, \mathbf{P})$, we have $q_{u,S} := \left(\prod_{v \in S} p_{uv}\right) \left(\prod_{v \in T_u \setminus S} (1 - p_{uv})\right)$.

$\omega_{u,\emptyset} = 0$ since $u \in V_0$. Applying Equation (4.2) to $A_{\sigma,\pi}$ implies

$$\begin{aligned} d(A_{\sigma,\pi}, u) &= 1 + \sum_{S \subset V, S \neq \emptyset} \omega_{u,S} \cdot d(A_{\sigma,\pi}, A_{\sigma,\pi}(u, S)) \\ &= 1 + \sum_{S' \subset V, S' \neq \emptyset} \omega_{u,S'} \cdot d(A_{\sigma,\pi}, A_{\sigma,\pi}(u, S' \cap T_u)) \qquad (4.3) \\ &= 1 + \sum_{S \subseteq T_u, S \neq \emptyset} \sum_{S' : S' \cap T_u = S} \omega_{u,S'} \cdot d(A_{\sigma,\pi}, A_{\sigma,\pi}(u, S)) \\ &= 1 + \sum_{S \subseteq T_u, S \neq \emptyset} q_{u,S} \cdot d(A_{\sigma,\pi}, A_{\sigma,\pi}(u, S)), \qquad (4.4) \end{aligned}$$

where (4.3) follows from the fact that by definition of $A_{\sigma,\pi}$, $A_{\sigma,\pi}(u, S)$ only depends on u and $S \cap T_u$, and (4.4) follows from the definition of $q_{u,S}$. Since by the definition of $A_{\sigma,\pi}$, $\sigma(A_{\sigma,\pi}(u, S)) > \sigma(u)$, the induction hypothesis implies that $d(A_{\sigma,\pi}, A_{\sigma,\pi}(u, S)) = r_{A_{\sigma,\pi}(u,S)}$. Furthermore, by the definition of $A_{\sigma,\pi}$ and π, we have that $r_{A_{\sigma,\pi}(u,S)} = \min_{v \in S}\{r_v\}$. Combined with Equation (4.4) and the definition of r_u, this shows $d(A_{\sigma,\pi}, u) = r_u$, as desired.

We now prove that $d(A, u) \geq r_u$. By Lemma 17 and the definition of V_0, we have $A(u, S) \in S \cap T_u \cap V_0$. Therefore,

$$d(A, A(u, S)) \geq \min_{v \in S \cap T_u \cap V_0} \{d(A, v)\}. \qquad (4.5)$$

By the induction hypothesis, we have that $d(A, v) \geq r_v$ for every $v \in T_u \cap V_0$. This,

together with equations (4.2) and (4.5), imply

$$\begin{aligned} d(A,u) &\geq 1 + \sum_{S\subset V, S\neq\emptyset} \omega_{u,S} \cdot d(A, A(u,S)) \\ &\geq 1 + \sum_{S\subset V, S\neq\emptyset} \omega_{u,S} \cdot \min_{v\in S\cap T_u\cap V_0}\{d(A,v)\} \\ &\geq 1 + \sum_{S\subset V, S\neq\emptyset} \omega_{u,S} \cdot \min_{v\in S\cap T_u\cap V_0}\{r_v\} \\ &= 1 + \sum_{S\subseteq T_u, S\neq\emptyset} \sum_{S':S'\cap T_u=S} \omega_{u,S'} \cdot \min_{v\in S\cap V_0}\{r_v\} \\ &= 1 + \sum_{S\subseteq T_u, S\neq\emptyset} q_{u,S} \cdot \min_{v\in S}\{r_v\} \qquad (4.6) \\ &= r_u, \end{aligned}$$

where (4.6) follows from the definition of $q_{u,S}$ and the fact that by property (iii) of σ, $S \cap V_0 = S$ for every $S \subseteq T_u$. This completes the proof of the induction step. □

of Theorem 16. Define the graph H, the ordering σ, the values r_u, and the ordering π as above. By Lemma 18, we have that $d(A_{\sigma,\pi}, s) \leq d(A, s)$. Since $G(n, \Omega)$ is d-searchable using A by assumption, we have that $d(A, s) \leq d$. Hence we have $d(A_{\sigma,\pi}, s) \leq d$, as desired. □

Note that in the above proof, the second ordering π was defined in terms of the first ordering σ. Therefore, the condition for the searchability of $G(n, \Omega)$ can be stated in terms of only one ordering σ as follows:

Corollary 19. *$G(n, \Omega)$ is d-searchable if and only if there is an ordering σ on the nodes for which $r_s \leq d$, where r is defined as in (4.1).*

A second corollary of the above characterization is that the following problem belongs in the complexity class **NP**.

> SEARCHABILITY: Given a positive integer n, an $n \times n$ matrix $\mathbf{P}$, two vertices s and t, and a positive number d, decide if $G(n, \mathbf{P})$ is d-searchable.

Corollary 20. SEARCHABILITY *is in* **NP**.

Proof. We use the ordering σ as a polynomial size certificate for membership in SEARCHABILITY. By Corollary 19, it is enough to show that r_u can be computed in polynomial time. We prove this by rewriting Equation 4.1 for vertices u with $u \neq t$ and $q_{u,\emptyset} = 0$. To do this, fix any such u and let $v_1, \ldots, v_t$ denote the vertices of T_u ordered in increasing order of their r_v's, i.e., $r_{v_1} \leq r_{v_2} \leq \cdots \leq r_{v_t}$. We have

$$\begin{aligned}
r_u &= 1 + \sum_{S \subseteq T_u, S \neq \emptyset} \prod_{v \in S} p_{uv} \prod_{v \in T_u \setminus S} (1 - p_{uv}) \cdot \min_{v \in S}\{r_v\} \\
&= 1 + \sum_{i=1}^{t} \sum_{S \subseteq \{1,\ldots,t\}, \min\{S\}=i} \prod_{j \in S} p_{uv_j} \prod_{j \in \{1,\ldots,t\} \setminus S} (1 - p_{uv_j}) \cdot r_{v_i} \\
&= 1 + \sum_{i=1}^{t} r_{v_i} p_{uv_i} \prod_{j=1}^{i-1} (1 - p_{uv_j}) \sum_{S \subseteq \{i+1,\ldots,t\}} \prod_{j \in S} p_{uv_j} \prod_{j \in \{i+1,\ldots,t\} \setminus S} (1 - p_{uv_j}) \\
&= 1 + \sum_{i=1}^{t} r_{v_i} p_{uv_i} \prod_{j=1}^{i-1} (1 - p_{uv_j}).
\end{aligned}$$

Given this equation, one can compute r_u given r_v for all $v \in T_u$ in polynomial time. Therefore, by Corollary 19, membership in SEARCHABILITY can be tested in polynomial time given the certificate σ. □

4.3 The Monotonicity Property

Armed with the characterization theorem of the previous section, we can now prove the following natural monotonicity property for searchability.

Theorem 21. *Let $\mathbf{P}$, $\mathbf{P}'$ be two $n \times n$ probability matrices such that for every i and j, we have $p_{ij} \leq p'_{ij}$. Fix the source and destination nodes s and t. Then, if $G(n, \mathbf{P})$ is d-searchable for some d, so is $G(n, \mathbf{P}')$.*

Proof. By Corollary 19, since $G(n, \mathbf{P})$ is d-searchable, there is an ordering σ such that the value r_s defined using Equation (4.1) is at most d. To show d-searchability of $G(n, \mathbf{P}')$, we apply the same ordering σ. Let $\{r'_u\}$ denote the values computed using Equation (4.1), but with $\mathbf{P}$ replaced by $\mathbf{P}'$. Similarly, we define $q'_{u,S}$'s. By

Corollary 19, it suffices to show that $r'_s \leq d$. To do this, we prove by induction that, for every $u \in V$, we have $r'_u \leq r_u$. This statement is trivial for $u = t$. We assume it is proved for every $v \in V$ with $\sigma(v) > \sigma(u)$, and prove it for u. First, note that if $q'_{u,\emptyset} > 0$, we have $q_{u,\emptyset} = \prod_{v \in T_u}(1 - p_{uv}) \geq \prod_{v \in T_u}(1 - p'_{uv}) = q'_{u,\emptyset} > 0$. Hence, $r_u = \infty$ and the inequality $r'_u \leq r_u$ holds. Therefore, we may assume $q'_{u,\emptyset} = 0$. Thus, we have

$$\begin{aligned} r'_u &= 1 + \sum_{S \subseteq T_u, S \neq \emptyset} \prod_{v \in S} p'_{uv} \prod_{v \in T_u \setminus S} (1 - p'_{uv}) \cdot \min_{v \in S}\{r'_v\} \\ &\leq 1 + \sum_{S \subseteq T_u, S \neq \emptyset} \prod_{v \in S} p'_{uv} \prod_{v \in T_u \setminus S} (1 - p'_{uv}) \cdot \min_{v \in S}\{r_v\} \end{aligned}$$

Let $1, 2, \ldots, k$ denote the nodes of T_u, ordered in such a way that $r_1 \leq r_2 \leq \cdots \leq r_k$. Recall that $\prod_{v \in S} p'_{uv} \prod_{v \in T_u \setminus S}(1 - p'_{uv})$ is the probability that in $G(n, \mathbf{P}')$, $\Gamma(u) \cap T_u = S$. Therefore, we have

$$\begin{aligned} r'_u &\leq 1 + \sum_{S \subseteq T_u, S \neq \emptyset} \mathbf{Pr}_{G(n,\mathbf{P}')}[\Gamma(u) \cap T_u = S] \cdot \min_{v \in S}\{r_v\} \\ &= 1 + \sum_{i=1}^{k} r_i \cdot \mathbf{Pr}_{G(n,\mathbf{P}')}[\min\{\Gamma(u) \cap T_u\} = i] \\ &= 1 + \sum_{i=1}^{k} r_i \left(\mathbf{Pr}_{G(n,\mathbf{P}')}[\min\{\Gamma(u) \cap T_u\} \leq i] - \mathbf{Pr}_{G(n,\mathbf{P}')}[\min\{\Gamma(u) \cap T_u\} \leq i - 1]\right) \\ &= 1 + r_k - \sum_{i=1}^{k-1} \mathbf{Pr}_{G(n,\mathbf{P}')}[\min\{\Gamma(u) \cap T_u\} \leq i](r_{i+1} - r_i). \end{aligned}$$

The coefficient of $(r_{i+1} - r_i)$ in the above expression is the probability of the event that the set of nodes that have an edge from u in $G(n, \mathbf{P}')$ contains at least one of the nodes $1, \ldots, i$. This event is monotone; therefore the probability of this event under

$G(n, \mathbf{P})$ is less than or equal to the probability under $G(n, \mathbf{P}')$. Therefore,

$$\begin{aligned} r'_u &\leq 1 + r_k - \sum_{i=1}^{k-1} \mathbf{Pr}_{G(n,\mathbf{P})}[\min\{\Gamma(u) \cap T_u\} \leq i](r_{i+1} - r_i) \\ &= 1 + \sum_{i=1}^{k} r_i \cdot \mathbf{Pr}_{G(n,\mathbf{P})}[\min\{\Gamma(u) \cap T_u\} = i] \\ &\leq r_u, \end{aligned}$$

where the last inequality follows from the definition of r_u (and holds with equality unless $q_{u,\emptyset} > 0$). This completes the proof of the induction step. □

We want to point out that the monotonicity result may seem surprising especially to those who are familiar with Kleinberg's result on the searchability of long range percolation graphs [Kle00]: in that graph model, there is a critical value for a parameter r that the graph is searchable in poly-logarithmic steps only if r takes on this value; either larger or smaller values for r would render graphs with polynomial time decentralized routing time. Our result is not contradictary with Kleinberg's result: in his model changing the parameter r affects the probabilities of many edges simultaneously; in particular those probabilities sum up to 1, so either increasing or decreasing the value of r will cause the probability of certain edges to decrease. Therefore the condition for our monotonicity result does not hold in his model.

Theorem 21 is proven under deterministic memoryless algorithms. We also prove monotonicity result for randomized algorithms with memory; the proof can be found in Section 5.4. However, we do not know whether a similar statement holds for randomized memoryless algorithms or deterministic algorithms with memory.

4.4 Summary and Open Problems

In this chapter, we defined a general class of random graphs, and gave a simple characterization of random graphs in this class that are searchable using decentralized deterministic memoryless algorithms. Our framework includes many of the previously studied small world networks. Two important corollaries of our characterization are

the monotonicity of the searchability property, and membership of the problem of testing searchability of a given graph in the complexity class **NP**.

Our framework and results lead to many interesting open questions. First, it would be interesting to settle the complexity of SEARCHABILITY. We proved that this problem belongs to **NP**. However, we do not know if this problem can be solved in polynomial time, or if it is **NP**-complete.

Characterizing searchability with respect to more general classes of decentralized search algorithms is another important open question. The class of search algorithms we considered in this paper can be generalized in three ways: allowing the algorithm to use randomization, allowing the algorithm to have memory, and allowing the algorithm to fail (i.e., not find a path to t) with a small probability ϵ. The following simple example shows any of these generalizations (or in fact, even allowing the algorithm to store one bit of memory) can drastically increase the power of the algorithm.

Example 2. *Consider a graph, consisting of vertices $s = u_0, u_1, \ldots, u_{n-2} = t$, and v. For every $i = 0, \ldots, n-1$, there is an edge from u_i to u_{i+1} with probability 1. Also, there is an edge from s to v and one from v to s with probability 1. Finally, there is an edge from v to t with probability $1 - 1/n$. A deterministic memoryless algorithm on this graph cannot ever visit v, since if it does and the direct edge from v to t is not present, it has to go back to s and then it will fall into a loop, as it has to go back to v. Therefore, any such algorithm reaches t in expected $n-2$ steps. However, if the algorithm is allowed to have one bit of memory, it can first go to v, and if the edge to t is not present, go back to s, remembering that it has already visited v. This achieves an expected path length of $(1-1/n)2 + \frac{1}{n} \cdot n < 3$. Similarly, if the algorithm is allowed to use randomization, it can flip a coin at s, and choose to go to either v or u_1 with probabilities $1 - 1/n$ and $1/n$, respectively. If it goes to v and the edge to t is not present, it returns to s and flips the coin again. It is not hard to prove that the expected number of steps that this algorithm takes to reach t is at most 6.*

The above example suggests that a characterization as simple as the one in Theorem 16 is probably impossible for algorithms that have memory, use randomization, or are allowed to fail. But at least it would be interesting to determine if these problems

belong to **NP**.

Finally, we note that despite the fact that it seems intuitive that the searchability property is monotone, we only know how to prove this property for deterministic memoryless algorithms and for algorithms with memory. Most importantly, proving this property for randomized memoryless algorithms is an intriguing open question.

Chapter 5

Stochastic Kronecker Graphs

A generative model based on Kronecker matrix multiplication was recently proposed by Leskovec et al. [LCKF05] as a model that captures many properties of real-world networks. In particular, they observe that this model exhibits a heavy-tailed degree distribution, and has an average degree that grows as a power law with the size of the graph, leading to a diameter that stays bounded by a constant (the so-called *densification power law* [LKF05]). Furthermore, Leskovec and Faloutsos [LF07] fit the stochastic model to some real world graphs, such as Internet Autonomous Systems graph and Epinion trust graphs, and find that Kronecker graphs with appropriate 2×2 initiator matrices mimic very well many properties of the target graphs.

Most properties of the Kronecker model (such as connectivity and diameter) are only rigorously analyzed in the deterministic case (i.e., when the initiator matrix is a binary matrix, generating a single graph, as opposed to a distribution over graphs), and empirically shown in the general stochastic case [LCKF05]. In this chapter we analyze some basic graph properties of stochastic Kronecker graphs with an initiator matrix of size 2. This is the case that is shown by Leskovec and Faloutsos [LF07] to provide the best fit to many real-world networks. We give necessary and sufficient conditions for Kronecker graphs to be connected or to have giant components of size $\Theta(n)$ with high probability. Our analysis of the connectivity of Kronecker graphs is based on a general lemma about connectivity in random graphs (Theorem 22) that might be of independent interest. We prove that under the parameters that the

graph is connected with high probability, it also has a constant diameter with high probability. This unusual property is consistent with the observation of Leskovec et al. [LKF05] that in many real-world graphs the effective diameters do not increase, or even shrink, as the sizes of the graphs increase, which is violated by many other random graph models with increasing diameters. Finally we show that Kronecker graphs do not admit short (poly-logarithmic) routing paths by decentralized routing algorithms based on only local information.

5.1 Model and Overview of Results

In this chapter we mainly focuses on stochastic Kronecker graphs with an initiator matrix of size 2, as defined below:

Definition 2. *A* (stochastic) Kronecker graph *is defined by*

(i) an integer k, and

(ii) a symmetric 2×2 matrix θ: $\theta[1,1] = \alpha, \theta[1,0] = \theta[0,1] = \beta, \theta[0,0] = \gamma$, where $0 \leq \gamma \leq \beta \leq \alpha \leq 1$. We call θ the base *matrix or the* initiator *matrix.*

The graph has $n = 2^k$ vertices, each vertex labeled by a unique bit vector of length k; given two vertices u with label $u_1 u_2 \dots u_k$ and v with label $v_1 v_2 \dots v_k$, the probability of edge (u, v) existing, denoted by $P[u, v]$, is $\prod_i \theta[u_i, v_i]$, independent on the presence of other edges.

We can think of the label of a vertex as describing k attributes of the vertex, each bit representing one attribute. The initiator matrix specifies how the values of two vertices on one bit influences connection strength (edge probability). We focus on the case where $\gamma \leq \beta \leq \alpha$; intuitively this means that if a vertex is strong in an attribute, then it has larger probability of being linked to. A special case of Kronecker graph model is that when $\alpha = \beta = \gamma$, it becomes the well studied random graph $G(n, p)$ with $p = \alpha^k$.

Leskovec and Faloutsos [LF07] showed that the Kronecker graph model with 2×2 initiator matrices satisfying the above conditions is already very powerful in simulating real world graphs. In fact, their experiment shows that the matrix [.98, .58; .58, .06]

is a good fit for the Internet AS graph. When the base matrix does not satisfy the condition stated in the above definition (i.e., when $\alpha \geq \gamma \geq \beta$ or $\beta \geq \alpha \geq \gamma$), Kronecker graphs appear to have different structural properties, and require different analytic techniques. We prove some of our results in these regimes as well (see Section 5.5); for all other sections, we concentrate on the $\gamma \leq \beta \leq \alpha$ case.

We analyze basic graph properties of the stochastic Kronecker graph model, including connectivity, giant component sizes, diameters and searchability. The main results are summarized below.

Theorem 25 The necessary and sufficient condition for Kronecker graphs to be connected with high probability (for large k) is $\beta + \gamma > 1$ or $\alpha = \beta = 1, \gamma = 0$. (Section 5.2.2)

Theorem 27 The necessary and sufficient condition for Kronecker graphs to have a giant component of size $\Theta(n)$ with high probability is $(\alpha + \beta)(\beta + \gamma) > 1$, or $(\alpha + \beta)(\beta + \gamma) = 1$ and $\alpha + \beta > \beta + \gamma$. (Section 5.2.3)

Theorem 29 When $\beta + \gamma > 1$, the diameters of Kronecker graphs are constant with high probability. (Section 5.3)

Theorem 31 Kronecker graphs are not $n^{(1-\alpha)\log_2 e}$-searchable. (Section 5.4)

To prove some of the results for Kronecker graphs, we sometimes study a general family of random graphs $G(n, P)$ (see also Section 4.1), which generalizes all random graph models where edges are independent, including Kronecker graphs and $G(n, p)$.

Definition 3. *A random graph $G(n, P)$, where n is an integer and P is an $n \times n$ matrix with elements in $[0, 1]$, has n vertices and includes each edge (i, j) independently with probability $P[i, j]$.*

Throughout this chapter we consider undirected $G(n, P)$: P is symmetric and edges are undirected. We prove two useful theorems about connectivity and searchability in this model, which may be of independent interest; namely, we show that if the min-cut size of the weighted graph defined by P is at least $c \ln n$ (c is a sufficiently large constant), then with high probability $G(n, P)$ is connected (Section 5.2.1); we also prove a monotonicity property for searchability in this model (Section 5.4).

In Section 5.6 we consider an alternative model where the labels of vertices are

chosen randomly from all possible k-bit vectors. Most of the results for Kronecker graphs still hold for this random label Kronecker model. Finally, we conclude in Section 5.7 by pointing out open problems and presenting some conjectures.

5.2 Connectivity and Giant Components

We first state a sufficient condition for connectivity of general random graphs $G(n, P)$ (Section 5.2.1), then use this condition to analyze connectivity and giant components of Kronecker graphs (Section 5.2.2, 5.2.3).

5.2.1 Connectivity of $G(n, P)$

We give a sufficient condition of the matrix P for $G(n, P)$ graphs to be connected. Let V be the set of all vertices. For any $S, S' \subseteq V$, define $P(i, S) = \sum_{j \in S} P[i, j]$; $P(S, S') = \sum_{i \in S, j \in S'} P[i, j]$.

Theorem 22. *If the min-cut size of the weighted graph defined by P is $c \ln n$ (c is a sufficiently large constant), i.e. $\forall S \subset V, P(S, V \setminus S) \geq c \ln n$, then with high probability $G(n, P)$ is connected.*

Proof. A *k-minimal cut* is a cut whose size is at most k times the min-cut size. We use the following result about the number of k-minimal cuts due to Karger and Stein [KS96]: In any weighted graph, the number of k-minimal cuts is at most $O((2n)^{2k})$.

Consider the weighted graph defined by P. Denote its min-cut size by t. We say a cut is a k-cut if its size is between kt and $(k + 1)t$. By the above result there are at most $O((2n)^{2k+2})$ k-cuts. Now consider a fixed k-cut in a random realization of $G(n, P)$: the expected size of the cut is at least kt, so by Chernoff bound the probability that the cut has size 0 in the realization is at most $e^{-kt/2}$. Taking the union bound over all k-cuts, for all $k = 1, 2, \ldots, n^2$, the probability that at least one cut has size 0 is bounded by

$$\sum_{k=1,\ldots,n^2} e^{-kt/2} O((2n)^{2k+2})$$

For $t = c \ln n$ where c is a sufficiently large constant, this probability is $o(1)$. Therefore with high probability $G(n, P)$ is connected. □

Note that $G(n, p)$ is known to be disconnected with high probability when $p \leq (1 - \epsilon) \ln n/n$, i.e., when the min-cut size is $(1 - \epsilon) \ln n$. Therefore the condition in the above theorem is tight up to a constant factor. Also, extrapolating from $G(n, p)$, one might hope to prove a result similar to the above for the emergence of the giant component; namely, if the size of the min-cut in the weighted graph defined by P is at least a constant, $G(n, P)$ has a giant component. However, this result is false, as can be seen from this example: n vertices are arranged on a cycle, and P assigns a probability of 0.5 to all pairs that are within distance c (a constant) on the cycle, and 0 to all other pairs. It is not hard to prove that with high probability $G(n, P)$ does not contain any connected component of size larger than $O(\log n)$.

5.2.2 Connectivity of Kronecker Graphs

We define the *weight* of a vertex to be the number of 1's in its label; denote the vertex with weight 0 by $\vec{0}$, and the vertex with weight k by $\vec{1}$. We say a vertex u is *dominated* by vertex u', denoted by $u \leq u'$, if for any bit i, $u_i \leq u'_i$. Recall that $P[u, v]$ is as defined in Definition 2.

The following lemmas state some simple facts about Kronecker graphs. Lemma 23 is trivially true given the condition $\alpha \geq \beta \geq \gamma$.

Lemma 23. *For any vertex u, $\forall v, P[u, v] \geq P[\vec{0}, v]; \forall S, P(u, S) \geq P(\vec{0}, S)$. Generally, for any vertices $u \leq u'$, $\forall v, P[u, v] \leq P[u', v]; \forall S, P(u, S) \leq P(u', S)$.*

Lemma 24. *The expected degree of a vertex u with weight l is $(\alpha + \beta)^l(\beta + \gamma)^{k-l}$.*

Proof. For any vertex v, let i be the number of bits where $u_b = v_b = 1$, and let j be the number of bits where $u_b = 1, v_b = 0$, then $P[u, v] = \alpha^i \beta^{j+l-i} \gamma^{k-l-j}$. Summing $P[u, v]$ over all v, the expected degree of u is

$$\sum_{i=0}^{l} \sum_{j=0}^{k-l} \binom{l}{i} \binom{k-l}{j} \alpha^i \beta^{j+l-i} \gamma^{k-l-j} = \sum_{i=0}^{l} \binom{l}{i} \alpha^i \beta^{l-i} \sum_{j=0}^{k-l} \binom{k-l}{j} \beta^j \gamma^{k-l-j} = (\alpha+\beta)^l(\beta+\gamma)^{k-l}$$

□

Theorem 25. *The necessary and sufficient condition for Kronecker graphs to be connected with high probability (for large k) is* $\beta + \gamma > 1$ *or* $\alpha = \beta = 1, \gamma = 0$.

Proof. We first show that this is a necessary condition for connectivity.
Case 1. If $\beta + \gamma < 1$, the expected degree of vertex $\vec{0}$ is $(\beta + \gamma)^k = o(1)$, with high probability vertex $\vec{0}$ is isolated and the graph is thus disconnected.
Case 2. If $\beta + \gamma = 1$ but $\beta < 1$, we again prove that with constant probability vertex $\vec{0}$ is isolated:

$$\begin{aligned} Pr[\vec{0} \text{ has no edge}] &= \prod_v (1 - P[\vec{0}, v]) = \prod_{w=0}^{k} (1 - \beta^w \gamma^{k-w})^{\binom{k}{w}} \geq \prod_{w=0}^{k} e^{-2\binom{k}{w}\beta^w \gamma^{k-w}} \\ &= e^{-2\sum_{w=0}^{k}\binom{k}{w}\beta^w \gamma^{k-w}} = e^{-2(\beta+\gamma)^k} = e^{-2} \end{aligned}$$

Now we prove it is also a sufficient condition. When $\alpha = \beta = 1, \gamma = 0$, the graph embeds a deterministic star centered at vertex $\vec{1}$, and is hence connected. To prove $\beta + \gamma > 1$ implies connectivity, we only need to show the min-cut has size at least $c \ln n$ and apply Theorem 22. The expected degree of vertex $\vec{0}$ excluding self-loop is $(\beta + \gamma)^k - \gamma^k > 2ck = 2c \ln n$ given that β and γ are constants independent on k satisfying $\beta + \gamma > 1$, therefore the cut $(\{\vec{0}\}, V \setminus \{\vec{0}\})$ has size at least $2c \ln n$. Remove $\vec{0}$ and consider any cut $(S, V \setminus S)$ of the remaining graph, at least one side of the cut gets at least half of the expected degree of vertex $\vec{0}$; without loss of generality assume it is S i.e. $P(\vec{0}, S) > c \ln n$. Take any node u in $V \setminus S$, by Lemma 23, $P(u, S) \geq P(\vec{0}, S) > c \ln n$. Therefore the cut size $P(S, V \setminus S) \geq P(u, S) > c \ln n$. □

5.2.3 Giant Components

Lemma 26. *Let H denote the set of vertices with weight at least* $k/2$*, then for any vertex u,* $P(u, H) \geq P(u, V)/4$.

Proof. Given u, let l be the weight of u. For a vertex v let $i(v)$ be the number of bits where $u_b = v_b = 1$, and let $j(v)$ be the number of bits where $u_b = 0, v_b = 1$.

we partition the vertices in $V \setminus H$ into 3 subsets: $S_1 = \{v : i(v) \geq l/2, j(v) < (k - l)/2\}, S_2 = \{v : i(v) < l/2, j(v) \geq (k - l)/2\}, S_3 = \{v : i(v) < l/2, j(v) < (k - l)/2\}$.

First consider S_1. For a vertex $v \in S_1$, we flip the bits of v where the corresponding bits of u is 0 to get v'. Then $i(v') = i(v)$ and $j(v') \geq (k - l)/2 > j(v)$. It is easy to check that $P[u, v'] \geq P[u, v]$, $v' \in H$, and different $v \in S_1$ maps to different v'. Therefore $P(u, H) \geq P(u, S_1)$.

Similarly we can prove $P(u, H) \geq P(u, S_2)$ by flipping the bits corresponding to 1s in u, and $P(u, H) \geq P(u, S_3)$ by flipping all the bits. Adding up the three subsets, we get $P(u, V \setminus H) \leq 3P(u, H)$. Thus, $P(u, H) \geq P(u, V)/4$. □

Theorem 27. *The necessary and sufficient condition for Kronecker graphs to have a giant component of size $\Theta(n)$ with high probability is $(\alpha + \beta)(\beta + \gamma) > 1$, or $(\alpha + \beta)(\beta + \gamma) = 1$ and $\alpha + \beta > \beta + \gamma$.*

Proof. When $(\alpha + \beta)(\beta + \gamma) < 1$, we prove that the expected number of non-isolated nodes are $o(n)$. Let $(\alpha + \beta)(\beta + \gamma) = 1 - \epsilon$. Consider vertices with weight at least $k/2 + k^{2/3}$, by Chernoff bound the fraction of such vertices is at most $\exp(-ck^{4/3}/k) = \exp(-ck^{1/3}) = o(1)$, therefore the number of non-isolated vertices in this category is $o(n)$; on the other hand, for a vertex with weight less than $k/2 + k^{2/3}$, by Lemma 24 its expected degree is at most

$$(\alpha + \beta)^{k/2+k^{2/3}}(\beta + \gamma)^{k/2-k^{2/3}} = (1 - \epsilon)^{k/2}(\frac{\alpha + \beta}{\beta + \gamma})^{k^{2/3}} = n^{-\epsilon'}c^{o(\log n)} = o(1)$$

Therefore overall there are $o(n)$ non-isolated vertices.

When $\alpha + \beta = \beta + \gamma = 1$, i.e. $\alpha = \beta = \gamma = 1/2$, the Kronecker graph is equivalent to $G(n, 1/n)$, which has no giant component of size $\Theta(n)$ [ER59].

When $(\alpha + \beta)(\beta + \gamma) > 1$, we prove that the subgraph induced by $H = \{v : \mathit{weight}(v) \geq k/2\}$ is connected with high probability, hence forms a giant connected component of size at least $n/2$. Again we prove that the min-cut size of H is $c \ln n$ and apply Theorem 22. For any vertex u in H, its expected degree is at least $((\alpha + \beta)(\beta + \gamma))^{k/2} = \omega(\ln n)$; by Lemma 26 $P(u, H) \geq P(u, V)/4 > 2c \ln n$. Now given any cut $(S, H \setminus S)$ of H, we prove $P(S, H \setminus S) > c \ln n$. Without loss of generality assume vertex $\vec{1}$ is in S. For any vertex $u \in H$, either $P(u, S)$ or $P(u, H \setminus S)$ is at

least $c \ln n$. If $\exists u$ such that $P(u, H \setminus S) > c \ln n$, then since $u \leq \vec{1}$, by Lemma 23 $P(S, H \setminus S) \geq P(\vec{1}, H \setminus S) \geq P(u, H \setminus S) > c \ln n$; otherwise $\forall u \in H, P(u, S) > c \ln n$, since at least one of the vertex is in $H \setminus S$, $P(S, H \setminus S) > c \ln n$.

Finally, when $(\alpha + \beta)(\beta + \gamma) = 1$ and $\alpha + \beta > \beta + \gamma$, let $H_1 = \{v : weight(v) \geq k/2 + k^{1/6}\}$, and we will prove that the subgraph induced by H_1 is connected with high probability by proving its min-cut size is at least $c \ln n$ (Claim 1), and that $|H_1| = \Theta(n)$ (Claim 2), therefore with high probability H_1 forms a giant connected component of size $\Theta(n)$.

Claim 1. *For any cut $(S, H_1 \setminus S)$ of H_1, $P(S, H_1 \setminus S) > c \ln n$.*

Proof of Claim 1. First, for any $u \in H_1$,

$$P(u, V) \geq (\alpha + \beta)^{k/2+k^{1/6}} (\beta + \gamma)^{k/2-k^{1/6}} = ((\alpha + \beta)/(\beta + \gamma))^{k^{1/6}} = \omega(\ln n).$$

By Lemma 26, $P(u, H) = \omega(\ln n)$. We will prove $P(u, H_1) \geq P(u, H \setminus H_1)/2$, and it follows that $P(u, H_1) = \omega(\ln n)$. Then we can apply the same argument as in case $(\alpha+\beta)(\beta+\gamma) > 1$ and prove that for any cut $(S, H_1 \setminus S)$ of H_1, $P(S, H_1 \setminus S) > c \ln n$: assume vertex $\vec{1}$ is in S; for any vertex $u \in H_1$, either $P(u, S)$ or $P(u, H_1 \setminus S)$ is at least $c \ln n$; if $\exists u$ such that $P(u, H_1 \setminus S) > c \ln n$, then $P(S, H_1 \setminus S) \geq P(\vec{1}, H_1 \setminus S) \geq P(u, H_1 \setminus S) > c \ln n$; otherwise $\forall u \in H_1, P(u, S) > c \ln n$, since at least one vertex is in $H_1 \setminus S$, we have $P(S, H_1 \setminus S) > c \ln n$.

It remains to prove $P(u, H_1) \geq P(u, H \setminus H_1)/2$. We will map each vertex $v \in H \setminus H_1$ to a vertex $f(v) = v' \in H_1$ such that $v \leq v'$ (and hence $P[u, v] \leq P[u, v']$), and each vertex in H_1 is mapped to at most twice. Once we have such a mapping, then $P(u, H \setminus H1) = \sum_{v \in H \setminus H_1} P[u, v] \leq \sum_{v \in H \setminus H_1} P[u, f(v)] \leq \sum_{v' \in H_1} 2P[u, v'] = 2P(u, H_1)$. The mapping is as follows: for each i in $[k/2, k/2 + k^{1/6})$, construct a bipartite graph G_i where the left nodes L_i are vertices with weight i, and make two copies of all vertices with weight $i + k^{1/6}$ to form the right nodes R_i, and add an edge if a right node dominates a left node. It is easy to see that the union of L_is forms exactly $H \setminus H_1$, while all right nodes are in H_1 and each node appears at most twice. The bipartite graph G_i has a maximum matching of size $|L_i|$, because all left (right) nodes have the same degree by symmetry and $|L_i| < |R_i|$ (proved below). We take

any such maximum matching to define the mapping and it satisfies that $v \leq f(v)$, $f(v) \in H_1$ and each $v' \in H_1$ is mapped to at most twice. Finally we prove $|L_i| < |R_i|$ for $k/2 \leq i < k/2 + k^{1/6}$:

$$\begin{aligned}\frac{|R_i|}{|L_i|} &= \frac{2\binom{k}{i+k^{1/6}}}{\binom{k}{i}} \geq \frac{2\binom{k}{k/2+2k^{1/6}}}{\binom{k}{k/2}} = \frac{2\frac{k}{2}\ldots(\frac{k}{2}-2k^{1/6}+1)}{(\frac{k}{2}+2k^{1/6})\ldots(\frac{k}{2}+1)} \\ &\geq 2(\frac{\frac{k}{2}-2k^{1/6}}{\frac{k}{2}})^{2k^{1/6}} \geq 2e^{-ck^{-5/6}*k^{1/6}} = 2e^{-o(1)} = 2(1-o(1)) > 1\end{aligned}$$

□

Claim 2. $|H_1| = \Theta(n)$.

Proof of Claim 2. We count the number of vertices with weight $k/2 + i$:

$$\binom{k}{k/2+i} \leq \binom{k}{k/2} = \Theta(\frac{\sqrt{2\pi k}(k/e)^k}{(\sqrt{2\pi k/2}(k/2e)^{k/2})^2}) = \Theta(\frac{2^k}{\sqrt{k}})$$

Therefore the size of $H \setminus H_1$ is at most $\sum_{i=0}^{k^{1/6}} \binom{k}{k/2+i} \leq k^{1/6} * 2^k/\sqrt{k} = o(n)$. It is easy to see $|H| > n/2$, thus $|H_1| > n/2 - o(n)$. □

Combining the two claims, we get that with high probability H_1 forms a giant connected component of size $\Theta(n)$. □

5.3 Diameter

We analyze the diameter of a Kronecker graph under the condition that the graph is connected with high probability. When $\alpha = \beta = 1, \gamma = 0$, every vertex links to $\vec{1}$ so the graph has diameter 2; below we analyze the case where $\beta + \gamma > 1$. We will use the following result about the diameter of $G(n, p)$, which has been extensively studied in for example [KL81, Bol90, CL01].

Theorem 28. *[KL81, Bol90] If $(pn)^{d-1}/n \to 0$ and $(pn)^d/n \to \infty$ for a fixed integer d, then $G(n, p)$ has diameter d with probability approaching 1 as n goes to infinity.*

Theorem 29. *If $\beta+\gamma > 1$, the diameters of Kronecker graphs are constant with high probability.*

Proof. Let S be the subset of vertices with weight at least $\frac{\beta}{\beta+\gamma}k$. We will prove that the subgraph induced by S has a constant diameter, and any other vertex directly connects to S with high probability.

Claim 3. *With high probability, any vertex u has a neighbor in S.*

Proof of Claim 3. We compute the expected degree of u to S:

$$P(u,S) \geq \sum_{j \geq \frac{\beta}{\beta+\gamma}k} \binom{k}{j} \beta^j \gamma^{k-j} = (\beta+\gamma)^k \sum_{j \geq \frac{\beta}{\beta+\gamma}k} \binom{k}{j} (\frac{\beta}{\beta+\gamma})^j (\frac{\gamma}{\beta+\gamma})^{k-j}$$

The summation is exactly the probability of getting at least $\frac{\beta}{\beta+\gamma}k$ HEADs in k coin flips where the probability of getting HEAD in one trial is $\frac{\beta}{\beta+\gamma}$, so this probability is at least a constant. Therefore $P(u,S) \geq (\beta+\gamma)^k/2 > c \ln n$ for any u; by Chernoff bound any u has a neighbor in S with high probability. □

Claim 4. $|S| \cdot \min_{u,v \in S} P[u,v] \geq (\beta+\gamma)^k$.

Proof of Claim 4. We have

$$\min_{u,v \in S} P[u,v] \geq \beta^{\frac{\beta}{\beta+\gamma}k} \gamma^{\frac{\gamma}{\beta+\gamma}k}$$

and

$$|S| \geq \binom{k}{\frac{\beta}{\beta+\gamma}k} \approx \frac{(\frac{k}{e})^k}{(\frac{\beta k}{(\beta+\gamma)e})^{\frac{\beta}{\beta+\gamma}k} (\frac{\gamma k}{(\beta+\gamma)e})^{\frac{\gamma}{\beta+\gamma}k}} = \frac{(\beta+\gamma)^k}{\beta^{\frac{\beta}{\beta+\gamma}k} \gamma^{\frac{\gamma}{\beta+\gamma}k}}$$

Therefore $|S| \cdot \min_{u,v \in S} P[u,v] \geq (\beta+\gamma)^k$. □

Given Claim 4, it follows easily that the diameter of the subgraph induced by S is constant: let $\beta+\gamma = 1+\epsilon$ where ϵ is a constant, the diameter of $G(|S|, (\beta+\gamma)^k/|S|)$ is at most $d = 1/\epsilon$ by Theorem 28; since by increasing the edge probabilities of $G(n,P)$ the diameter cannot increase, the diameter of the subgraph of the Kronecker graph induced by S is no larger than that of $G(|S|, (\beta+\gamma)^k/|S|)$. Therefore, by Claim 3,

for every two vertices u and v in the Kronecker graph, there is a path of length at most $2 + 1/\epsilon$ between them. □

5.4 Searchability

In Section 5.3 we showed that the diameter of a Kronecker graph is constant with high probability, given that the graph is connected. However it is yet a question whether a short path can be found by a decentralized algorithm where each individual only has access to local information. In this section, we examine searchability of Kronecker graphs: instead of restricting with deterministic memoryless routing algorithms as in Chapter 4, we allow a more powerful family of routing algorithms - randomized algorithms with memory, and prove that Kronecker graphs do not admit short routing path even using such powerful algorithms.

We define searchability on $G(n, P)$ graphs using randomized algorithms with memory as follows:

Definition 4. *In a* decentralized routing algorithm for $G(n, P)$*, the message is passed sequentially from a current message holder to one of its neighbors until reach the destination t, using only local information. In particular, the message holder u at a given step has knowledge of:*

(i) the probability matrix P;

(ii) the label of destination t;

(iii) edges incident to all visited vertices.

A $G(n, P)$ graph is d-searchable *if there exists a decentralized routing algorithm such that for any destination t, source s, with high probability the algorithm can find an s-t path no longer than d.*

We first give a monotonicity result of searchability on general random graphs $G(n, P)$ (an analogue of Theorem 21 for randomized search algorithms with memory), then use it to prove Kronecker graphs with $\alpha < 1$ is not poly-logarithmic searchable. It is possible to directly prove our result on Kronecker graphs, but we believe the monotonicity theorem might be of independent interests.

Theorem 30. *If $G(n, P)$ is d-searchable, and $P \leq P'$ ($\forall i, j, P[i, j] \leq P'[i, j]$), then $G(n, P')$ is d-searchable.*

Proof. Given $G(n, P')$ we simulate $G(n, P)$ by ignoring some edges. Given a realization G of $G(n, P')$, we keep an edge (i, j) in G with probability $P[i, j]/P'[i, j]$, and delete the edge otherwise; do so for each edge independently. We claim that random graphs generated by the above process is equivalent to $G(n, P)$: the probability that edge (i, j) presents is $P'[i, j] * (P[i, j]/P'[i, j]) = P[i, j]$, independent on other edges. Now we have a $G(n, P)$ graph, we use its decentralized routing algorithm, which will find a path with length at most d with high probability for any s and t.

Note that we cannot process all edges in the beginning, because there is no global data structure to remember which edges are deleted. Instead we will decide whether to delete an edge the first time we visit one of its endpoints and this information will be available to all vertices visited later. □

Theorem 31. *Kronecker graphs are not $n^{(1-\alpha)\log_2 e}$-searchable.*

Proof. Let P be the probability matrix of the Kronecker graph, and P' be the matrix where each element is $p = n^{-(1-\alpha)\log_2 e}$. We have $P \leq P'$ because $max_{i,j} P[i, j] \leq \alpha^k \leq e^{-(1-\alpha)k} = n^{-(1-\alpha)\log_2 e} = p$. If the Kronecker graph is $n^{(1-\alpha)\log_2 e}$-searchable, then by Theorem 30 $G(n, p)$ where $p = n^{-(1-\alpha)\log_2 e}$ is also $n^{(1-\alpha)\log_2 e}$-searchable. However, $G(n, p)$ is not $\frac{1}{p}$-searchable. This is because given any decentralized algorithm, whenever we first visit a vertex u, independent on the routing history, the probability that u has a direct link to t is no more than p, hence the routing path is longer than the geometry distribution with parameter p, i.e. with constant probability the algorithm cannot reach t in $1/p$ steps. □

5.5 Other Settings of Parameters

In the previous sections, we investigate the various properties of stochastic Kronecker graphs with 2×2 initiator matrices under the constraint $\alpha \geq \beta \geq \gamma$. In this section we present some results on other settings of parameters, still with 2×2 initiator

matrices. Those settings also have interesting practical implications; for example, in the case where $min(\alpha, \gamma) \geq \beta$ a vertex tends to link to vertices more alike to itself.

5.5.1 Connectivity

We prove that if $\beta + min(\alpha, \gamma) > 1$, then Kronecker graphs are connected with high probability, for arbitrary $\alpha, \beta, \gamma \in [0, 1]$.

Theorem 32. *If $\beta + min(\alpha, \gamma) > 1$, then Kronecker graphs are connected with high probability.*

Proof. We first prove for the case where $\alpha \geq \gamma \geq \beta$. We may assume $\alpha = \gamma$. Let $d(u, v)$ be the Hamming-distance between the labels of u and v. Then $P[u, v] = \gamma^k (\frac{\beta}{\gamma})^{d(u,v)}$.

Claim 5. *In λk-th power of hypercube (i.e. a vertex u is connected to all vertices v with $d(u, v) \leq \lambda k$), the min-cut size is at least $\binom{k}{\lambda k - 1}/2$.*

Proof of Claim 5. Given a cut $(S, \bar{S})$, take any vertex $u \in S$. For any vertex u' with $d(u, u') = 1$, the number of common neighbors of u and u' is at least $\binom{k}{\lambda k - 1}$, because all vertices within distance $\lambda k - 1$ to u are within distance λk to u' and are thus their common neighbors. If the cut size is less than $\binom{k}{\lambda k - 1}/2$, then at least half of those common neighbors must be in S, and u' must also be in S. Apply the same argument iteratively and all vertices will end up being in S. □

Now given a cut $(S, \bar{S})$ in the Kronecker graph, for any edge of this cut in λk-th power of hypercube, the Hamming-distance of the two endpoints is at most λk, and the corresponding edge presents in the Kronecker graph with probability at least $\gamma^k (\frac{\beta}{\gamma})^{\lambda k}$, therefore

$$P(S, \bar{S}) \geq \frac{1}{2} \binom{k}{\lambda k - 1} \gamma^k (\frac{\beta}{\gamma})^{\lambda k} \geq \frac{1}{2k} ((\frac{1}{\lambda})^{\lambda} (\frac{1}{1-\lambda})^{1-\lambda} \gamma (\frac{\beta}{\gamma})^{\lambda})^k$$

Let $\lambda = \frac{\beta}{\beta+\gamma}$, $P(S, \bar{S}) \geq \frac{1}{2k}(\beta + \gamma)^k = \omega(\ln n)$. According to Theorem 22, the graph is connected with high probability.

For the case where $\beta \geq max(\alpha, \gamma)$, the proof is similar as above. Without loss of generality assume $\alpha \geq \gamma$. Instead of λk-th power of hypercube, we can prove that in the graph where a vertex u is connected to all vertices v with $d(u, v) \geq \lambda k$, the min-cut size is at least $\binom{k}{\lambda k-1}/2$; again, for any edge in such hypercube, the corresponding edge in the Kronecker graph has probability at least $\gamma^k(\frac{\beta}{\gamma})^{\lambda k}$, and we can use computation exactly like the above to bound the cut size in Kronecker graphs. We have proven the result when $\alpha \geq \beta \geq \gamma$ (Theorem 25). □

When $\beta + min(\alpha, \gamma) < 1$ or $\beta + min(\alpha, \gamma) = 1$ but all of them are strictly less than 1, the graph is not connected with at least constant probability because vertex $\vec{0}$ is isolated with probability at least e^{-2} as the proof in Theorem 25.

5.5.2 Giant Components

Unlike the previous results, we are not able to give a necessary and sufficient condition for giant components for other setting of parameters.

Theorem 33. *Consider the case where $\alpha \geq \gamma \geq \beta$. If $(\alpha + \beta)(\beta + \gamma) < 1$, then with high probability there is no giant component with size $\Theta(n)$; if $\beta + \sqrt{\alpha\gamma} > 1$, then with high probability there exists a giant component with size $\Theta(n)$.*

Proof. If $(\alpha + \beta)(\beta + \gamma) < 1$, then we can prove that the expected number of non-isolated nodes are $o(n)$, with exactly the same argument as in Theorem 27.

Now if $\beta + \sqrt{\alpha\gamma} > 1$, again let H be the subset of vertices with weight at least $k/2$ and we will show that the subgraph induced by H is connected with high probability, which forms a giant connected component of size $\Theta(n)$.

Claim 6. *In the subgraph induced by H in λk-th power of hypercube, the min-cut size is at least $\binom{k}{\lambda k-1}/8$.*

Proof of Claim 6. The proof is similar to that of Claim 5, except that now we are only concerned with nodes in H. We will prove that for any node $u \in H$, the number of vertices in H whose distance to u are $\lambda k - 1$ is at least $\binom{k}{\lambda k-1}/4$. Since the total number of such vertices is $\binom{k}{\lambda k-1}$, we only need to prove that at least a quarter of

them are in H; equivalently, we may consider the following process: we have k coins with $w = weight(u)$ HEADs and $k - w$ TAILs; randomly choose $l = \lambda k - 1$ coins and flip them; we need to prove that the probability that finally there are more than half HEADs is at least $1/4$ if we start with $w \geq k/2$ HEADs and flip $l < k/2$ coins. To prove this, let us imagine partitioning the coins into two sets, a set S with $k - w$ HEADs and $k - w$ TAILs, and a set T with the remaining HEADs. After flipping, the probability that S has at least as many HEADs as TAILs is $1/2$ by symmetry; if we flip less than half of the coins ($l < k/2$), it is easy to see that T ends up more HEADs than TAILs with probability at least $1/2$. When both events happen, we guarantee that finally there more than half HEADs, therefore this happens with probability at least $1/4$. □

Note that for any two vertices that are both in H, there must be more bits with $1 - 1$ matching than $0 - 0$ matching. Therefore if $u, v \in H$ have Hamming distance d, then the probability of edge (u, v) in the Kronecker graph is at least $\beta^d(\alpha\gamma)^{(k-d)/2}$.

Now given a cut $(S, \bar{S})$ in the subgraph induced by H in Kronecker graph, for any edge of this cut in λk-th power of hypercube, the Hamming-distance of the two endpoints is at most λk, so the corresponding edge presents in the Kronecker graph with probability at least $\beta^{\lambda k}(\sqrt{\alpha\gamma})^{(1-\lambda)k}$, therefore

$$P(S, \bar{S}) \geq \frac{1}{8}\binom{k}{\lambda k - 1}\beta^{\lambda k}(\sqrt{\alpha\gamma})^{(1-\lambda)k} \geq \frac{1}{8k}((\frac{1}{\lambda})^{\lambda}(\frac{1}{1-\lambda})^{1-\lambda}\sqrt{\alpha\gamma}(\frac{\beta}{\sqrt{\alpha\gamma}})^{\lambda})^k$$

Let $\lambda = \frac{\beta}{\beta+\sqrt{\alpha\gamma}}$, $P(S, \bar{S}) \geq \frac{1}{2k}(\beta + \sqrt{\alpha\gamma})^k = \omega(\ln n)$. Applying Theorem 22, the subgraph is connected with high probability. □

There is a gap between our necessary condition and sufficient condition. We conjecture that $(\alpha + \beta)(\beta + \gamma) > 1$ is also the sufficient condition for linear size giant component. The case $\gamma \geq \alpha \geq \beta$ is exactly symmetric to $\alpha \geq \gamma \geq \beta$. However, the proof technique does not apply directly to the $\beta \geq max(\alpha, \gamma)$ case where vertices with large Hamming distance are more likely to link to each other.

5.6 An Alternative Model: Vertices with Random Labels

In this section we consider an alternative model where the labels of the vertices are chosen uniformly at random from all possible k bit vectors; the probability of an edge is decided by the labels of the two endpoints as before. For simplicity of discussion, let us assume for now that the number of vertices is $n = 2^k$.

Ideally we wish we could prove a general connectivity theorem in this alternative model similar to Theorem 22: define $RG(n, P)$ to be a random graph with n vertices where the label l of any vertex is chosen at random from $1..n$ (with replacement), and an edge (u, v) presents independently with probability $P[l(u), l(v)]$; if the min-cut size of the weighted graph P is $c \ln n$, then $GR(n, P)$ is connected with high probability. If we could prove it, then all the results and proofs could automatically carry over to the alternative model. Unfortunately such a statement is not true. Consider a deterministic graph consisting of three parts S, T and $\{v\}$: each of S and T is a complete graph containing almost half vertices, and there is no edge between S and T; vertex v links to all vertices. Let P be the adjacent matrix of this graph. The min-cut size of P is much larger than $c \ln n$, but $GR(n, P)$ is not connected if v is not chosen, which happens with a constant probability $1/e$.

Fortunately most of the results regarding Kronecker graphs are still true in the random label model, and the proofs are similar except that now in addition we need to consider the random choices of labels. We present as an example the proof of the sufficient condition for connectivity in the random label model (analogous to Theorem 25); extending other results is very similar.

Theorem 34. *When $\beta + \gamma > 1$, Kronecker graphs with random labels are connected with high probability.*

Proof. Let us first decide the random choices of vertex labels; denote the choice by θ. The family of random graphs conditioned on θ is a $G(n, P)$ graph where P is decided by θ. We want to show that if $\beta + \gamma > 1$, with high probability (with respect to random choices of θ), $P(\theta)$ has min-cut size at least $c \ln n$; then we can apply

Theorem 22 and conclude that with high probability the graph is connected.

Now let us fix θ, and consider the min-cut in the matrix $P(\theta)$. Imagine that we add a dummy vertex with label $\vec{0}$, then its expected degree over random choices of θ is $(\beta/2+\gamma/2)^k * (2^k) = (\beta+\gamma)^k > 2c\ln n$. What is more, its expected degree is the sum of n i.i.d. random variables whose value is in the range of $[0,1]$, so its expected degree conditioned on θ sharply concentrates around the expectation according to Chernoff bound, i.e. with high probability its expected degree conditioned on θ is at least $2c\ln n$. Now consider any cut $(S, V \setminus S)$ where V is the vertex set excluding the dummy node; note that S and V are deterministic sets given θ. At least one side of the cut gets at least half of the expected degree of the dummy; without loss of generality assume it is S i.e. $P(\vec{0}, S) > c\ln n$. Take any node u in $V \setminus S$, by Lemma 23, $P(u,S) \geq P(\vec{0},S) > c\ln n$. Therefore the cut size $P(S, V\setminus S) \geq P(u,S) > c\ln n$. □

We may relax the requirement that the number of vertices $n = 2^k$. When $n = \Theta(2^k)$, the same results hold; for other n, our analysis techniques are still applicable, but the condition will depend on the values of n and k.

5.7 Summary and Open Problems

In this chapter we analyzed several important properties of stochastic Kronecker graphs with an initiator matrix of size 2; in particular, we studied connectivity, giant component, diameter and searchability for the most interesting case where $\gamma \leq \beta \leq \alpha$. We also extended some of the results to other settings of parameters and variants of the model.

There are several interesting problems about Kronecker graphs that we would like to explore further in the future:

1. Conditions for giant components with size $\Theta(n)$ in Kronecker graphs with 2×2 initiator matrices. We presented one necessary and one sufficient conditions for giant components (Section 5.5.2), but there is a gap between the two bounds. We propose the following conjecture:

Conjecture 35. *If* $(\alpha+\beta)(\beta+\gamma) < 1$*, then with high probability there is no giant component with size* $\Theta(n)$*; if* $(\alpha+\beta)(\beta+\gamma) > 1$*, then with high probability there exists a giant component with size* $\Theta(n)$*.*

2. Diameters of Kronecker graphs with 2×2 initiator matrices. We conjecture that the graph has a constant diameter with high probability under the condition that the graph is connected with high probability. We proved this result for the $\alpha \geq \beta \geq \gamma$ case (Theorem 29). For the $\alpha \geq \gamma \geq \beta$ case, we checked two extreme cases where $\alpha = \beta = \gamma$, or $\alpha = \gamma = 1$ and β is small; in both cases the diameter is constant.

   Conjecture 36. *Under the parameters where the graph is connected with high probability, the diameter is constant with high probability.*

3. Conditions for connectivity in Kronecker graphs with initiator matrices of arbitrary sizes. One necessary condition is that the sum of any row is at least 1. Otherwise suppose the matrix is $d \times d$ and row 0 is $(a_1, \ldots, a_d)$ where $a_1 + \ldots + a_d < 1$, then the expected degree of vertex $\vec{0}$ is $(\frac{1}{d}(a_1 + \ldots + a_d))^k * d^k = o(1)$, therefore $\vec{0}$ is isolated with high probability. The other necessary condition is that if we view the initiator matrix as the adjacent matrix of a graph, the graph must be connected. Otherwise suppose 0 and 1 are disconnected in the initiator matrix, then vertices $\vec{0}$ and $\vec{1}$ in the Kronecker graph generated from this matrix are disconnected with probability 1.

Chapter 6

Link Privacy in Social Networks

6.1 Motivation

Participation in online communities is becoming ubiquitous. Not only do people keep personal content such as their journals, photos, bookmarks and contacts online, they also increasingly interact online, both socially and professionally. When participating in an online blogging community, such as Blogger or LiveJournal, it is fairly common to specify a set of friends, i.e., a set of users whose blogs one reads, and who are permitted to read one's blog posts written within a friend-only privacy setting. When participating in photo-sharing communities, such as Flickr and PicasaWeb, it is also fairly common to specify a set of friends, i.e. a set of users who are permitted to see one's private pictures, and whose pictures one is interested in seeing or being notified about.

In online communities whose primary goal is social networking, such as MySpace, Facebook, and LinkedIn, each user's set of trusted users is of paramount importance to their activity on the website. For example, in the case of LinkedIn, an online network of professionals, each connection signifies a professional relationship between two individuals, such as having worked together in the past. One's connections, and connections' connections, and so on, form a network that an individual has access to and can tap to foster professional connections or to find potential collaborators, clients, employers and subject experts.

It is no surprise then that a major part of the value of a social network or of participating in a web-service with an online community, is in the community itself and in its structure. Furthermore, entities other than the users themselves could benefit from the knowledge of the social network structure even when such an outcome is not intended. For example, a potential employer might want to be able to look at the immediate network of a potential employee, to evaluate its size and quality, or to be able to ask former colleagues of the individual for their opinions. A potential advertiser might want to look at the profiles and interests of people in the user's network, in order to more accurately infer the user's interests for the purpose of targeted advertisements. In other words, knowledge of the network opens the door for powerful data mining, some of which may not be desirable to the users of the social network.

Some social networks such as LiveJournal allow users to see all the links of any user in the network. However our motivation for this chapter is networks such as LinkedIn where relationships (in this case professional) between users may be sensitive, and the link information is a valuable asset to the network owner, so a user is permitted only limited access to the link structure. We will use LinkedIn as a running example throughout the paper. A LinkedIn user can see the profiles and the list of friends of each of his friends, the profiles of friends of friends, as well as the names of people that the friends of friends are connected to. Viewing the social network as a graph (with users as nodes and links between users as edges), a LinkedIn user can see his edge-node-edge-node neighborhood as well as the names of users at distance 3 from him.

Even though the intention is to give users access to only a small portion of the social network graph, one could imagine resourceful users stitching together local information about different parts of the network to gain global information about the network as a whole. In this chapter, we analyze methods one could employ to obtain information about the link structure of a social network, and the difficulties that lie therein depending on the type of neighborhood access permitted by the web application. We concentrate on the case in which an attacker, whose goal is to ascertain a significant fraction of the links in a network, obtains access to parts of

the network by gaining access to the accounts of some select users. This is done either maliciously by breaking into user accounts or by offering each user a payment or service in exchange for their permission to view their neighborhood of the social network. We describe both experimental and theoretical results on the success of such an attack in obtaining the link structure of a significant portion of the network, and make recommendations for the type of neighborhood access that a web application should permit to prevent such an attack and protect the privacy of its network and its users.

Our work is thus different from the line of work in [BDK07] and [MMJ+07] where "anonymized" releases of social network graphs are considered. In that setting the owner of the social network releases the underlying graph structure after removing all username annotations of the nodes, and the goal of an attacker is to uniquely identify the node that corresponds to a real world entity in this anonymized graph. In contrast, we consider a case where no underlying graph is released and, in fact, the owner of the network would like to keep the entire structure of the graph hidden from any individual. However, an attacker with access to some number of user accounts is able to learn who links to whom for a significant fraction of users on the network.

In Section 6.2, we discuss related work on privacy in social networks. Section 6.3 lays out the formal model of the kind of attack we consider and the goal of the attack. We present experimental results of the success of different attack strategies on both simulated and real world social network graphs in Section 6.4, and present a rigorous theoretical analysis in Section 6.5. We conclude in Section 6.6 with recommendations of actions for web service providers that would preserve user privacy.

6.2 Related Work

There has been much recent interest in the context of anonymized data releases. [BDK07] considers a framework where a social network owner announces the intention to release an anonymized version of the network graph, i.e. a copy where true user names are replaced with random ids but the network structure is unchanged. They show that, if given a chance to create as few as $\Theta(\log(n))$ new accounts in the network,

prior to the release of the anonymized network, an attacker can efficiently recover the structure of connections between any $\Theta(\log^2(n))$ nodes chosen apriori by identifying the new accounts that he inserted in to the network. In [MMJ+07], the authors experimentally evaluate how much background information about the structure of the neighborhood of an individual would be sufficient for an attacker to uniquely identify the individual in such an anonymized graph. In [ZG07] the emphasis is on protecting the types of links associated with individuals in an anonymized release. Simple edge-deletion and node-merging algorithms are proposed to reduce the risk of sensitive link disclosure.

While the privacy attack model of [BDK07] is very interesting and has received substantial research focus, in this paper we study the privacy in social networks from an entirely different angle. We consider a case where no underlying graph is released, and, in fact, the owner of the network would like to keep the entire structure of the graph hidden from any one individual. An attacker we consider does not have access to the entire anonymized structure of the graph, nor is his goal to de-anonymize particular individuals from that graph. In contrast, he aims to compromise the link privacy of as many individuals as possible by determining the link structure of the graph based on the local neighborhood views of the graph from the perspective of several non-anonymous users.

In the attack strategies that we consider, the effectiveness of the attack is likely to depend to a large extent on the degree distribution of the nodes in the social network which is commonly known to be close to power law [WF94, CS04, MK]. In our theoretical analysis of the effectiveness of an attack, we would therefore like to use a model for social networks that guarantees a power law distribution. We use the Power Law Random Graph model of [BC78] and [ACL00]. Unlike the evolutionary models such as preferential attachment, this model does not consider the process that forms the power law degree sequence; rather, it takes the power law degree distribution as given and generates a random graph whose degree distribution follows such a power law. The model is described in more detail in Section 6.4.

As a side note, our theoretical and experimental results also have implications on the power of lookahead in speeding up web crawls studied in [MST07]. [MST07]

analyzes a particular crawling strategy where the crawler performs a random walk on the graph; we can potentially use some of the strategies in this paper for crawling, and they provide larger coverage than a random walk crawler. A similar problem has been studied in [ALPH01], but as pointed out by [MST07], the main result of [ALPH01] does not hold.

6.3 The Model

In this section we formalize the privacy threat drafted in the Introduction. We first define the primary goal of the privacy attack considered in this paper (Section 6.3.1); then discuss the knowledge of social networks available to users, and thus adversaries (Section 6.3.2); finally, we list possible attack strategies (Section 6.3.3).

6.3.1 Goal of the Attack

We view a social network as an undirected graph $G = (V, E)$, where the nodes V are the users and the edges E represent friendships or interactions between users. While some online social networks (such as LiveJournal) allow friendship to be one-directional, many others (such as LinkedIn and Facebook) do require mutual friendship, and in those graphs interactions between users are naturally modeled as undirected edges. From now on we consider only undirected graphs for simplicity of discussion and analysis.

As informally discussed in Section 6.1, the primary goal of the privacy attack is to discover information about who links to whom in the network. Thus we measure the effectiveness of an attack by the amount of the network graph structure exposed to the attacker:

> *Node Coverage*: the fraction of nodes whose entire immediate neighborhood is known. We say that a node is *covered* and its entire immediate neighborhood is known, if the attacker knows precisely which nodes it connects to and which nodes it is not connected to.

We also refer to node coverage as *user coverage* or simply *coverage*. We may also measure the effectiveness of an attack by "edge coverage". There are two possible definitions of edge coverage. One definition is: among all existing edges, the fraction of edges known to the attacker. However, this notion of edge coverage does not measure the attacker's knowledge of non-existence of an edge, and is therefore not a comprehensive view of an attacker's knowledge. The other possible definition is: among all pairs of users, the fraction of pairs between which the attacker knows whether or not an edge exists. As we will see in the following sections, node coverage makes sense for the attacker strategies we consider and directly implies this kind of edge coverage. We will use node coverage as the primary measure throughout the chapter.

6.3.2 The Network through a User's Lens

As mentioned in Section 6.1, LinkedIn allows a user to see all edges incident to himself as well as all edges incident to his friends. In general, a social network could choose how much visibility to provide its users depending on how sensitive links are. We distinguish such differing neighborhood visibilities by the term *lookahead*. We say that the social network has lookahead of 0 if a user can see exactly who he links to; that the social network has lookahead 1 if a user can see exactly the friends that he links to as well as the friends that his friends link to. In general, we say that the social network has lookahead l if a user can see all of the edges incident to the nodes within distance l from him. Using this definition, LinkedIn has lookahead 1. In terms of node coverage, a lookahead of l therefore means that each node covers all nodes within distance l of it; the nodes that are at distance $l + 1$ are seen, but not covered.

There are several other variations on the type of access that a user can have to the graph structure. Some networks allow a user to see the shortest path between himself and any other user, some display the path only if it is relatively short, some only display the length of the shortest path, and others let the user see the joint friends he has with any other user. Some networks offer the users a combination of these options. We largely ignore these additional options in our discussion, while noting

that all of them potentially make the task of discovering the entire link structure easier.

In addition to the connection information, a typical online social network also provides a search interface, where we can search for users by username, first name, last name, other identifying information such as email, or different combinations of the above. The search interface will return usernames of all users who satisfy the query, often with the numbers of friends of those users, i.e. the degrees of the nodes corresponding to those users in the social network graph, G. For example LinkedIn allows such queries and provides degree information.

To summarize, the possible interface that a social network exposes to users which may be leveraged by attackers to target specific user accounts includes:

- *neighbors(username, password, l)*: Given a username with proper authentication information, return all users within distance l and all edges incident to those users in the graph G;

- *exists(username)*: Given a username, return whether the user exists in the network;

- *degree(username)*: Given a username, return the degree of the user with that username. Note that *degree(username)* implies *exists(username)*;

- *userlist()*: Return a complete list of all usernames in the network.

Among above, only *neighbors()* requires authentication information; all others are publicly available. A social network might expose some or all of those functions to its users. For example, LinkedIn provides *neighbors(username, password, l)* for $l = 0$ or 1, but not for $l > 1$; it also provides *exists(username)* and *degree(username)*. Most social networks do not expose *userlist()* directly; however, adversaries may be able to generate the complete (or nearly complete) list of usernames through other functionalities provided by the network. For example, Facebook allows fuzzy search of, say, the first two letters of last names, so one can easily obtain its username list by searching all possible two-letter combinations.

Note that a particular network may expose only a subset of the above functions; even if all functions are available, their costs may vary greatly. Therefore when we discuss attack strategies in the next section we list the functions required by each strategy, and when we evaluate and compare strategies there is a trade-off between effectiveness of an attack and the interface it requires.

6.3.3 Possible Attack Strategies

As mentioned in Section 6.1, we consider an attack that attempts to discover the link structure of a social network by strategically gaining access to the accounts of some select users, and combining the views from their different perspectives in order to obtain a coherent network picture. The access to user accounts is gained either by maliciously breaking into the accounts or by offering each user a payment, service or reward that increases their status within the social network (for example, adding them as a friend, or writing them a positive recommendation or review) in exchange for their permission to view their neighborhood of the social network. Recall that each time the attacker gains access to a user account, he immediately covers all nodes that are at distance less than the lookahead distance enabled by the social network, i.e., he gets to see exactly the edges incident to these nodes. So if the lookahead is l, then by bribing node u, he immediately covers all nodes that are within distance l of u. Additionally, he gets to see all nodes that are within distance $l + 1$. We will call the users, to whose user accounts the attacker has managed to obtain access, *bribed* users.

We now discuss the possible strategies that an attacker may want to utilize when targeting users to bribe. Different strategies may require different knowledge of the social network; we order the strategies in decreasing order of the information needed for the attacker to implement them. The attacker would like to obtain as large coverage as possible by bribing a fixed number of users; in Sections 6.4 and 6.5 we will study the performance of these strategies both experimentally and theoretically.

- BENCHMARK-GREEDY: In this strategy at each time, the attacker picks the next user to bribe as the one whose perspective on the network will give

the largest possible amount of new information. More formally, in each step the algorithm picks the node covering the maximum number of nodes not yet covered. For lookahead less than or equal to 1, this can be implemented if the attacker can access the degrees of all users on the network. However for lookahead greater than 1, it requires that for each node the attacker have access to all usernames covered by that node, which is not a primitive that we consider available to the attacker. So this strategy serves as a benchmark rather than a feasible attack – it is the optimal bribing algorithm that is computationally feasible when given access to the entire graph G[1].
Requires: G;

- Heuristically Greedy: A heuristically greedy bribing strategy picks the next user to bribe as the one who can offer the largest possible amount of new information, according to some heuristic measure. The heuristic measure is chosen so that the attacker does not need to know G to evaluate it. In particular, we consider the following strategy:

  – DEGREE-GREEDY: Select the next user to bribe as the one with the maximum "unseen" degree, i.e., its degree according to the *degree(username)* function minus the number of edges incident to it already seen by the adversary.
  Requires: *neighbors(username, password, l), degree(username), userlist();*

- HIGHEST-DEGREE: In this strategy, users are bribed in descending order of their degrees.
  Requires: *neighbors(username, password, l), degree(username), userlist();*

- Random: In this strategy, the users to bribe are picked randomly. Variations could include picking the users uniformly at random, proportional to their degrees, etc. In particular, we study one strategy in this category:

[1]The attacker's goal is to cover G or most of G by bribing as few nodes as possible. However, to find the optimal bribing set for a given G is NP hard, and the best polynomial-time (thus computationally feasible) approximation algorithm is the greedy algorithm described above. This can be proved by a reduction to the set cover problem.

  - UNIFORM-RANDOM: here the attacker selects users uniformly at random.
    Requires: *neighbors(username, pwd, l), userlist();*

- Crawler: This strategy is similar to the Heuristically Greedy strategy, but the attacker chooses the next node to bribe only from the nodes already seen (within distance $l + 1$ of some bribed node). We consider one such strategy:

  - DEGREE-GREEDY-CRAWLER: The next node to bribe is selected as the one with the maximum unseen degree among nodes already seen.
    Requires: *neighbors(username, password, l), degree(username)*;

Note that the DEGREE-GREEDY-CRAWLER and UNIFORM-RANDOM strategies are very easily implementable in practice on most social networks, since they do not require any knowledge of nodes that are not within the attacker's visible neighborhood. Further the DEGREE-GREEDY-CRAWLER strategy could also be used by web crawlers to crawl web pages more rapidly when each web page stores information about its lookahead.

6.4 Experimental Results

In this section we present experimental results of applying the strategies of Section 6.3.3 to both synthetic and real world social network data. At a high level, the experiments explore the fraction, f, of nodes that need to be bribed for an attacker using the different bribing strategies to achieve $1 - \varepsilon$ node-coverage for a social network with lookahead l. Our experimental results show that the number of users an attacker needs to bribe in order to acquire a fixed coverage decreases exponentially with increase in lookahead. In addition, this number is also fairly small from the perspective of actual implementation, indicating that several of the attack strategies from Section 6.3.3 are feasible to implement in practice and achieve good results.

We implemented and evaluated the following five strategies, ordered in decreasing order of the complexity of the interface that needs to be exposed for an attacker

to implement them: BENCHMARK-GREEDY (abbreviated as BENCHMARK); DEGREE-GREEDY (abbreviated as GREEDY); HIGHEST-DEGREE (abbreviated as HIGHEST); UNIFORM-RANDOM (abbreviated as RANDOM); DEGREE-GREEDY-CRAWLER (abbreviated as CRAWLER).

6.4.1 Results on Synthetic data

Generating Synthetic Graphs

In order to measure the effectiveness of the different attack strategies, we generate random graphs with power-law degree distributions and apply our strategies to them. We use the Power Law Random Graph model in [ACL00] to generate the graphs (see Section 6.2 for an explanation of why we choose this model). The model essentially generates a graph that satisfies a given degree distribution, picking uniformly at random from all such graphs.

More specifically, let n be the total number of nodes in G, α ($2 < \alpha \leq 3$) be the power law parameter; let d_0 and d_{max} be the minimum and maximum degree of any node in the graph, respectively. First, we generate the degrees of all the nodes $d(v_i), i = 1, \ldots, n$ independently according to the distribution $Pr[d(v_i) = x] = C/x^{\alpha}, d_0 \leq x \leq d_{max}$, where C is the normalizing constant. Second, we consider $D = \sum d(v_i)$ minivertices which correspond to the original vertices in a natural way and generate a random matching over D. Finally, for each edge in the matching, we construct an edge between corresponding vertices in the original graph. As a result, we obtain a random graph with a given power-law degree distribution. The graph is connected almost surely [GMS03]. The graph has a few multi-edges and self-loops that we remove in our experiments, without affecting the power law degree distribution.

Furthermore, following the practice of [MST07], we cap d_{max}, the maximum number of connections that a user may have at $\sqrt{n}$, reflecting the fact that in a large enough social network, a single person, even a very social one, cannot know a constant fraction of users.

We denote the fraction of nodes bribed by f, the number of nodes bribed by

$k = fn$, and the coverage achieved by $1 - \varepsilon = \frac{number\ of\ nodes\ covered}{n}$.

Comparison of Strategies

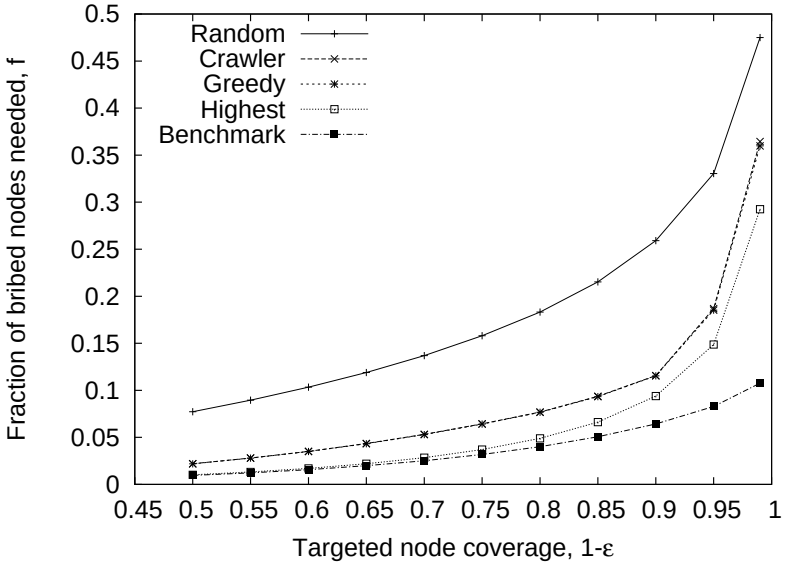
We analyze the relative performance of five of the strategies proposed in Section 6.3.3 on random power-law graphs with $100,000$ nodes, $\alpha = 3$ and $d_{min} = 5$. We run each strategy on 10 power-law graphs generated as described in 6.4.1, with the aim of achieving coverage of 0.5 through 0.99. For each strategy, we average across the runs the fraction of nodes that need to be bribed with that strategy in order to achieve the desired coverage. This gives us f as a function of $1 - \varepsilon$ for each strategy. We present the results for lookahead 1 and 2 in Figure 6.1.

From the experimental results we can see that BENCHMARK has the best performance, i.e., to achieve a fixed coverage of $1 - \varepsilon$, BENCHMARK needs to bribe fewer nodes than any other strategy. However, as mentioned previously, BENCHMARK is not possible to implement in practice because it requires knowledge of the entire graph structure, and so it can only serve as a benchmark upper bound on how good any given strategy can be.

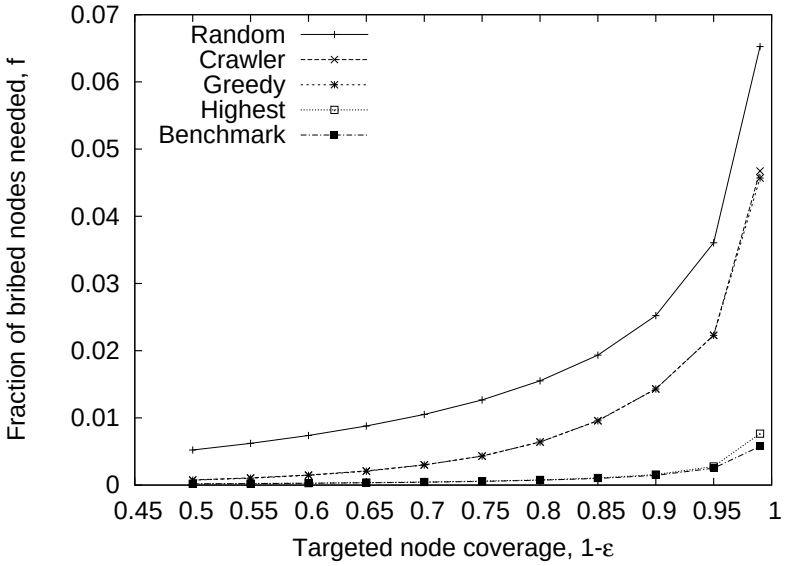
Some of the other observations we make are that HIGHEST and BENCHMARK perform almost equally well when the desired coverage is less than 90%. However the performance of HIGHEST deteriorates as the lookahead increases and desired coverage increases.

Somewhat surprisingly, we find that GREEDY performs worse than HIGHEST while GREEDY and CRAWLER perform equally well. Not surprisingly, RANDOM performs the worst out of all the strategies.

We choose the following strategies to analyze in more detail and show that these can pose serious threats to the link privacy: HIGHEST and CRAWLER as a measure of performance of a somewhat sophisticated yet still implementable strategy; RANDOM as the most easily implementable attack strategy that can serve as a lower bound on how well other strategies can work.



(a) Lookahead 1



(b) Lookahead 2

Figure 6.1: Comparison of attack strategies on synthetic data. We plot the fraction of bribed nodes against node coverage on synthetic graphs (with 100,000 nodes), using the five bribing strategies with lookahead 1 and 2. Lines for CRAWLER and GREEDY are almost overlapping.

Dependence on the Number of Users

We analyze how the the performance of a bribing strategy changes with an increase in the number of nodes in the graph. We observe in Figure 6.2 that the number of nodes k that need to be bribed using the HIGHEST strategy in order to achieve a fixed coverage of $1 - \varepsilon$ is linear in the size of the network, for various values of ε and lookahead of 2. The same was observed for other values of lookahead. Since HIGHEST has the best performance among all the suggested realistically implementable strategies, this implies that k is linear in n for other strategies as well. However, it is worth observing that the slope of the linear function is very small, for all ε not very close to 1. As discussed in the next section, this makes all of the strategies a realistic threat at lookahead greater than 1.

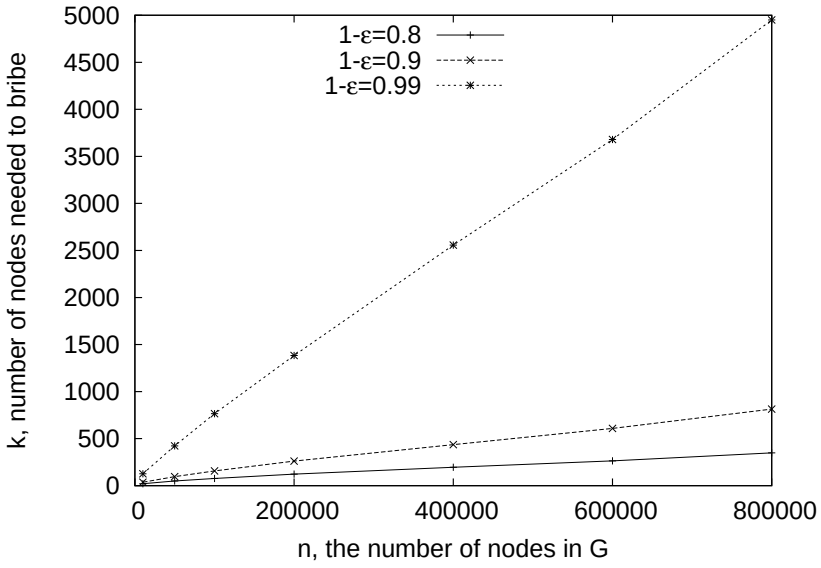


Figure 6.2: Number of nodes that need to be bribed for graph sizes n using HIGHEST with lookahead 2 for coverage $0.8, 0.9, 0.99$.

Dependence on Lookahead

The performance of all strategies substantially improves with increase in lookahead. Consider, for example, the performance of the HIGHEST strategy, plotted in Figure 6.3, and also detailed in Table 6.1. With each increase in lookahead, the number of

nodes k that need to be bribed in order to achieve the same $1-\varepsilon$ coverage decreases by two orders of magnitude. If in an $800,000$-user social network, one needs to bribe $36,614$ users to achieve a 0.8 coverage with lookahead 1 using HIGHEST, then in the network of the same size one only needs to bribe 348 users to achieve the same coverage with lookahead 2, and only 7 users to achieve the same coverage at lookahead 3. In other words, the number of nodes that need to be bribed to achieve fixed coverage decreases exponentially in the lookahead, rapidly making the HIGHEST strategy attack a feasible threat at lookahead 2 in social networks with under 1 million users, and a feasible threat at lookahead 3 in social networks with as many as 100 million users.

1-ε	f_1/f_2	f_2/f_3
0.7	112.3	39.3
0.8	105.0	49.1
0.9	88.6	65.1
0.95	73.1	79.0
0.99	46.6	101.7

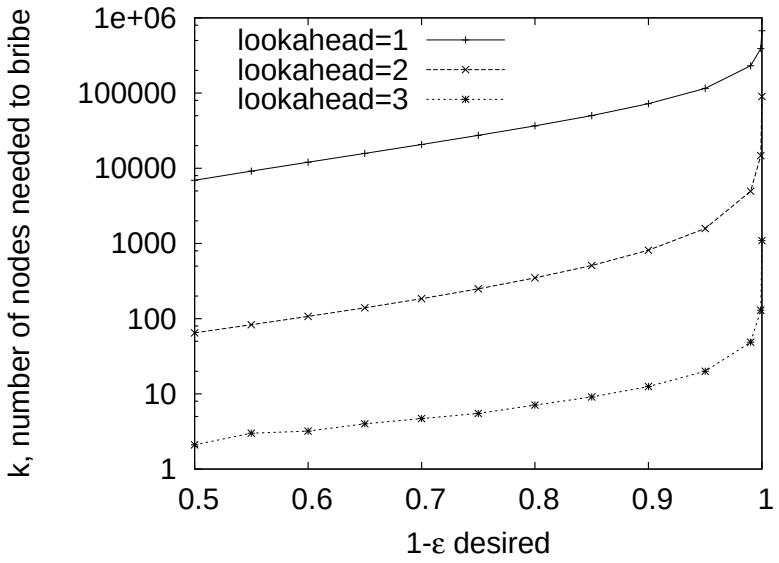
Table 6.1: Factors of improvement in performance of HIGHEST with increases in lookaheads.

We observe a similar exponential decrease with increase in lookahead in the number of nodes that need to be bribed for CRAWLER (Figure 6.3); similar result is observed for RANDOM as well.

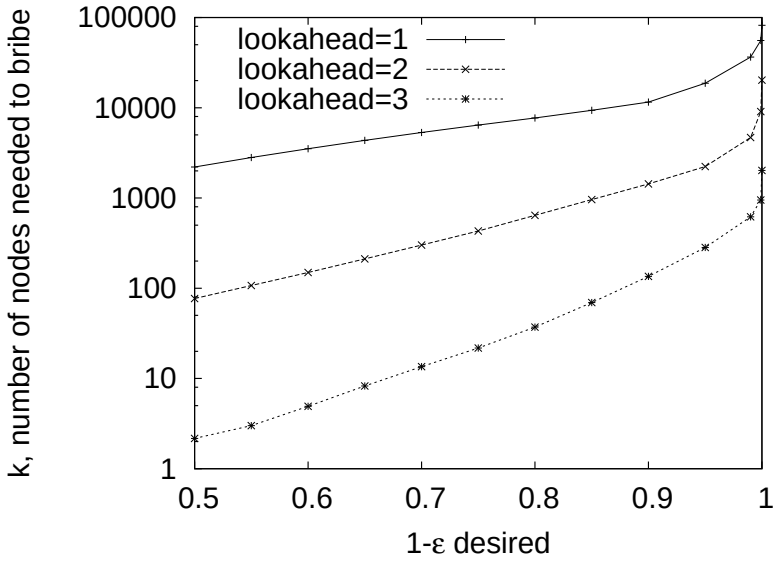
6.4.2 Results on Real data

We also ran our experiments on real network data. As we felt that actually bribing LinkedIn users with a goal of recovering the network's structure would be inappropriate as a research exercise, we used LiveJournal instead as a proxy, since its link structure is readily available. We crawled LiveJournal using the friends and friends-of listings to establish connections between users. We extracted a connected component of $572,949$ users.

The obtained LiveJournal grah has an average degree of 11.8, $d_{min} = 1, d_{max} =$



(a) HIGHEST, $n = 800k$



(b) CRAWLER, $n = 100k$

Figure 6.3: Effect of lookahead on synthetic data. The figures show the number of nodes to bribe to achieve $1 - \varepsilon$ coverage with different lookahead, using HIGHEST and CRAWLER respectively. Note that y axis is log scale.

$1974, \alpha = 2.6$.

Comparison of Strategies

Analogous to our discussion in Section 6.4.1 we compare the performance of the different bribing strategies on the LiveJournal graph at lookaheads of 1 and 2 in Figures 6.4. The relative performance of the different strategies is the same as on the synthetic data.

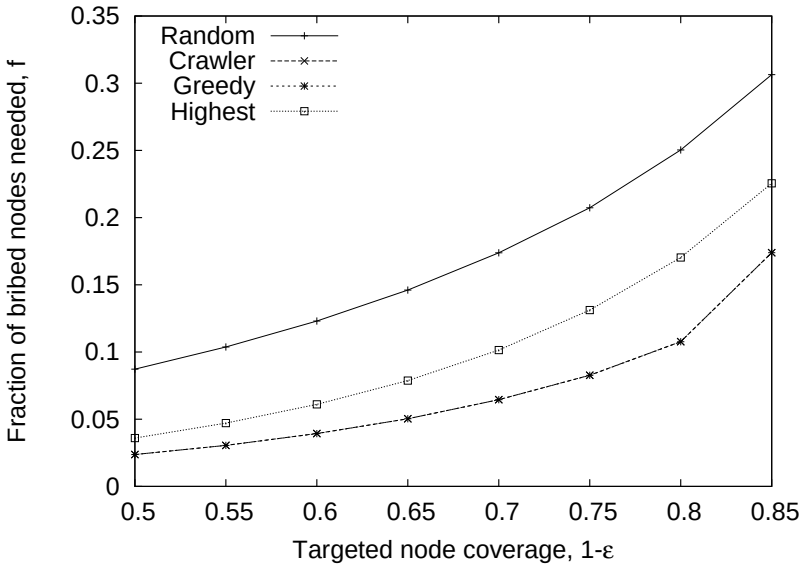
Dependence on Lookahead

Furthermore, as on the synthetic data, the number of nodes that need to be bribed in order to achieve fixed coverage of LiveJournal decreases exponentially with an increase in lookahead (see Figure 6.5).

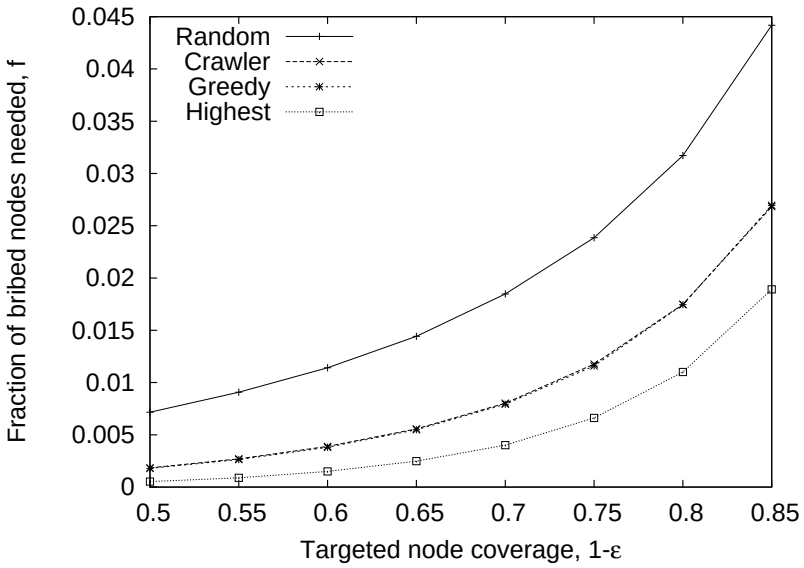
These experiments also confirm our hypothesis that while none of the strategies are a truly feasible threat at lookahead 1, some of them become feasible at lookahead 2, and all of them become feasible at lookahead 3. For example, in order to obtain 80% coverage of a $572,949$-user LiveJournal graph using lookahead 2 HIGHEST needs to bribe $6,308$ users, and to obtain the same coverage using lookahead 3 HIGHEST needs to bribe 36 users – a number that is sufficiently small given the size of the network; and thus possible to bribe in practice.

6.5 Theoretical Analysis for Random Power Law Graphs

In this section, we theoretically analyze the performance of a couple of the bribing strategies from Section 6.3: we analyze the fraction of nodes an attacker needs to bribe to reach a constant node coverage with high probability for a power law graph drawn from the Power Law Random Graph model (details about the model in Section 6.4). We were able to analyze the following two strategies in particular: UNIFORM-RANDOM and HIGHEST-DEGREE. We carry out the analysis under power law

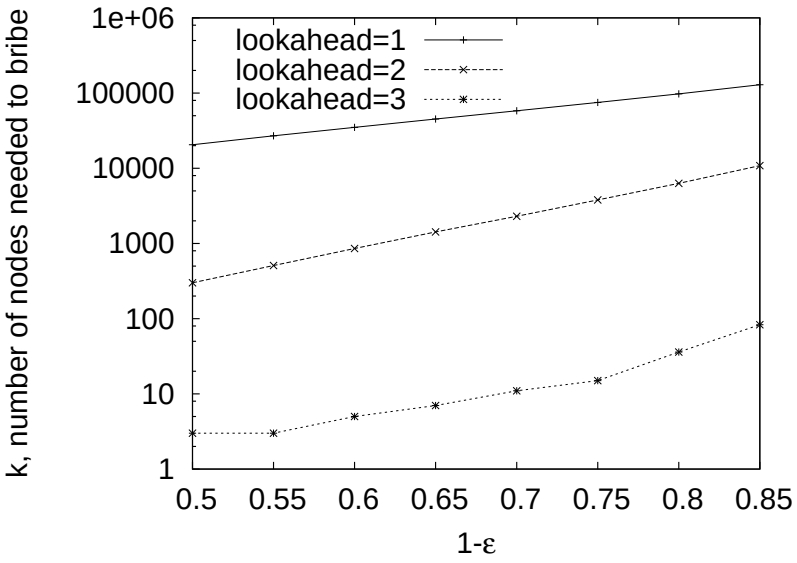


(a) Lookahead 1

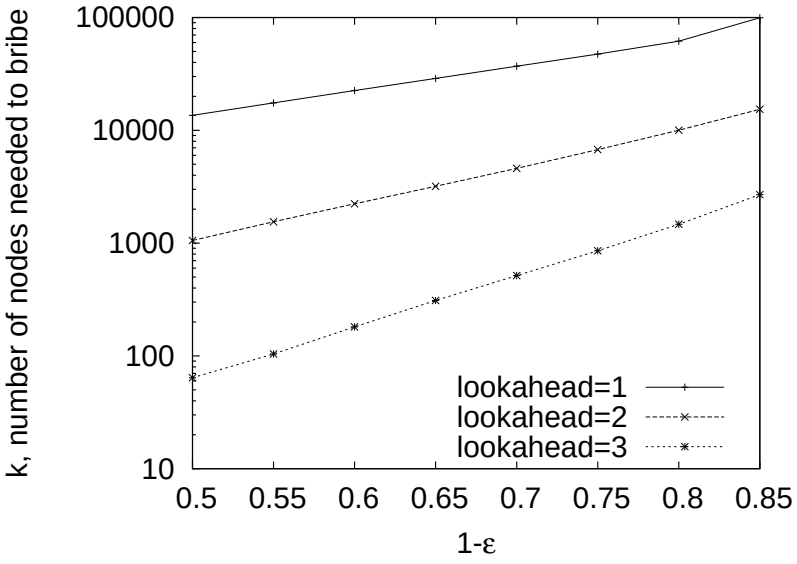


(b) Lookahead 2

Figure 6.4: Comparison of attack strategies on LiveJournal data. We plot the fraction of bribed nodes against node coverage on LiveJournal graph, using the four bribing strategies with lookahead 1 and 2. The two lines for CRAWLER and GREEDY are almost overlapping.



(a) HIGHEST, $n = 800k$



(b) CRAWLER, $n = 100k$

Figure 6.5: Effect of lookahead on LiveJournal data. The figures show the number of nodes to bribe to achieve $1 - \varepsilon$ coverage with different lookaheads, using HIGHEST and CRAWLER respectively. Note that y axis is log scale.

graphs; for configuration models with other degree distributions, our analysis technique still applies, but the result will depend on the degree distribution.

We use the same notations as defined in Section 6.4. To recap: n is the number of nodes in the network; m is the number of edges; d_0 is the minimum degree of a node; $\sqrt{n}$ is the maximum degree; α is the power law parameter; C is the normalizing constant in degree distribution so that $\sum_{d=d_0}^{\sqrt{n}} Cd^{-\alpha} = 1$; the target node coverage is $1-\varepsilon$; f is the fraction of bribed nodes and $k = fn$ is the number of bribed nodes.

6.5.1 Analysis of Lookahead 1

We first answer the following question: if in each trial we cover a node randomly with probability proportional to its degree (all trials being independent), after how many trials will we have covered $(1-\varepsilon)n$ distinct nodes? Once we answer this question, we will come back to estimating the number of nodes to bribe by studying the rate of "covering nodes" of different bribing strategies. This question is similar to the well-known *coupon collector* problem if all nodes have an equal probability of being covered.

Lemma 37. *[MR95] (Coupon Collector) Consider an unlimited supply of coupons of n distinct kinds. At each trial if we collect a coupon uniformly at random and indpendently of previous trials, then after t trials, the number of distinct coupons collected has the expectation $n(1-e^{-t/n})$ and is sharply concentrated.*

Now in our question each node has a different probability of being covered (collected), thus can be viewed as a weighted coupon collector problem. Schelling studied this problem in 1954 [vS54] when the probability of sampling each coupon is explicitly given. In our problem we not only need to consider the random choices of coupon collection, but also the random realization of the graph.

Lemma 38. *In each trial we cover a node randomly with probability proportional to its degree, independently of previous trials. After $\frac{-\ln \varepsilon_0}{d_0} 2m$ trials, the number of distinct nodes covered is at least $n(1-\varepsilon-o(1))$ with high probability, where $\varepsilon = \sum_{d=d_0}^{\sqrt{n}} \varepsilon_0^{d/d_0} Cd^{-\alpha}$.*

Proof. First consider all nodes with degree d_0 in the graph. Let c_0 be the fraction of such nodes; $c_0 = \Theta(1)$ with high probability (see for example [MST07]). In each trial, the probability of covering a node with degree d_0 is $c_0 n d_0/2m$ (since the total sum of degrees is $2m$); in $\frac{-\ln \varepsilon_0}{d_0} 2m$ trials, on expectation there are $-c_0 n \ln \varepsilon_0$ trials choosing nodes with degree d_0, and by Chernoff bound ([MR95]) there are at least $-(c_0 - o(1))n \ln \varepsilon_0$ such trials with high probability. All nodes with degree d_0 have an equal probability of being covered, so it is a classic coupon collector problem if constrained on such trials. By Lemma 37, the expected number of nodes with degree d_0 collected is at least

$$c_0 n(1 - e^{(c_0 - o(1))n \ln \varepsilon_0 / c_0 n}) = c_0 n(1 - \varepsilon_0 - o(1))$$

and by sharp concentration the number of such nodes collected is at least $c_0 n(1 - \varepsilon_0 - o(1))$ with high probability.

Now consider nodes with degree $d_i = \Theta(1)$. Let c_i be the fraction of such nodes and again $c_i = \Theta(1)$ with high probability. By a similar argument as above, there are at least $-(c_i - o(1))\frac{d_i}{d_0} n \ln \varepsilon_0$ trials choosing nodes with degree d_i, and the number of such nodes collected is at least $c_i n(1 - \varepsilon_0^{d_i/d_0} - o(1))$.

Finally for all the remaining nodes with degree $\omega(1)$, the total number of such nodes is $o(n)$, so we miss at most $o(n)$ such nodes.

In total, with high probability we miss at most $\sum_{d_i} c_i n \varepsilon_0^{d_i/d_0} + o(n)$ nodes after $\frac{-\ln \varepsilon_0}{d_0} 2m$ trials. In the power law random graph model, $c_i = C d_i^{-\alpha} + o(1)$ with high probability, therefore we miss at most $\sum_{d=d_0}^{\sqrt{n}} C d^{-\alpha} n \varepsilon_0^{d/d_0} + o(n)$, i.e. we collect at least $n(1 - \varepsilon - o(1))$ nodes. □

Both ε and ε_0 are between 0 and 1, and we can show that ε is always smaller than ε_0. Table 6.2 gives some values of ε and ε_0; for example when $\alpha = 3$ and $d_0 = 5$, $\varepsilon = 0.4$ gives $\varepsilon_0 = 0.534$.

Now we come back to the original question: how many nodes do we need to bribe to cover a $1-\varepsilon$ fraction of the graph, using different bribing strategies with lookahead 1? Remember that with lookahead 1 we cover a node only if it is a direct neighbor of a bribed node.

Pick a node to bribe using any strategy. Consider one edge of the bribed node, the other endpoint of the edge can be any node v and the probability of it being v is $d(v)/2m$ if we randomize over all graphs with the given degree sequence (the argument can be formalized using Principle of Deferred Decisions [MR95]). Therefore if we bribe a node with degree d and covers all its neighbors, it is equivalent to having made d trials to cover nodes in the graph. And if we bribe nodes $b_1, b_2, \ldots, b_k$ and cover all their neighbors, it is like having made $D = \sum_{i=1}^{k} d(b_i)$ such trials. However not each trial covers a node v with the same probability proportional to its degree: if v was already covered in a previous trial, the probability of covering it again decreases, whereas if it was not covered in a previous trial, the probability of covering it with each new trial increases. More formally, the events that a node is collected in different trials are negatively correlated; this only increases the number of distinct nodes we expect to cover and so the result in Lemma 38 on the number of distinct nodes collected can still serve as a lower bound. In summary, we have the following theorem.

Theorem 39. *Bribe nodes $b_1, b_2, \ldots, b_k$ (all b_is distinct) selected using an arbitrary strategy. Denote the sum of their degrees by $D = \sum_{i=1}^{k} d(b_i)$. If $D = \frac{-\ln \varepsilon_0}{d_0} 2m$, then the node coverage is at least $1-\varepsilon-o(1)$ with high probability under lookahead 2, where $\varepsilon = \sum_{d=d_0}^{\sqrt{n}} \varepsilon_0^{d/d_0} C d^{-\alpha}$.*

Theorem 39 establishes the connection between the total degree of bribed nodes (regardless of the strategy of choosing bribed nodes) and the node coverage. Now for any particular bribing strategy, all we need to analyze is the total degree of k bribed nodes.

We first analyze the strategy of bribing nodes uniformly at random (without replacement). In any graph a node chosen uniformly at random has the expected degree $\bar{d} = 2m/n$, and bribing k nodes yields expected total degree $D = 2mk/n$; plugging into Theorem 39 we get the following Corollary.

Corollary 40. *If an attacker bribes $\frac{-\ln \varepsilon_0}{d_0} n$ nodes according to the UNIFORM-RANDOM strategy, then he covers at least $n(1 - \varepsilon - o(1))$ nodes with high probability, where $\varepsilon = \sum_{d=d_0}^{\sqrt{n}} \varepsilon_0^{d/d_0} C d^{-\alpha}$.*

Next we analyze the HIGHEST-DEGREE strategy. To apply Theorem 39, we compute the expected total degree of the top $k = fn$ nodes, where f is a constant. Let d be such that

$$\sum_{x=d+1}^{\sqrt{n}} Cx^{-\alpha} < f \leq \sum_{x=d}^{\sqrt{n}} Cx^{-\alpha}$$

When n is large we can use integration to approximate the sum and get the equation

$$\int_{d}^{\sqrt{n}} Cx^{-\alpha} dx = f$$

Remember that C is the normalizing constant satisfying $\int_{d_0}^{\sqrt{n}} Cx^{-\alpha} dx = 1$. Solving the equation, we get $C \approx (\alpha - 1)d_0^{\alpha-1}$ and $d \approx d_0 k^{1/(1-\alpha)}$. When n is large and f is a constant, the smallest degree of the top fn nodes is sharply concentrated around d; we can roughly assume it is d. Now the top fn nodes have maximum degree $\sqrt{n}$ and minimum degree d, and the probability of having degree x is proportional to $x^{-\alpha}$. Therefore the expected sum of degrees of the top fn nodes is

$$fn * \frac{\sum_{x=d}^{\sqrt{n}} x * x^{-\alpha}}{\sum_{x=d}^{\sqrt{n}} x^{-\alpha}} \approx nC \int_{d}^{\sqrt{n}} x * x^{-\alpha} dx \approx \frac{\alpha - 1}{\alpha - 2} d_0 n k^{\frac{\alpha-2}{\alpha-1}}$$

On the other hand the overall total degree

$$2m = \sum_{x=d_0}^{\sqrt{n}} x * Cx^{-\alpha} n \approx \frac{\alpha - 1}{\alpha - 2} d_0 n$$

Therefore the expected sum of the degrees of the top fn nodes is $D = 2mk^{\frac{\alpha-2}{\alpha-1}}$. When n is large and f is a constant, the smallest degree of the top fn nodes sharply concentrates around d and the above analysis holds with high probability with a lower order error. (Note that sharp concentration may not hold when $f = O(1/n)$ thus k is a constant.)

Corollary 41. *If an attacker bribes* $(\frac{-\ln \varepsilon_0}{d_0})^{\frac{\alpha-2}{\alpha-1}} n$ *nodes according to the HIGHEST-DEGREE strategy, then he covers at least* $n(1 - \varepsilon - o(1))$ *nodes with high probability, where* $\varepsilon = \sum_{d=d_0}^{\sqrt{n}} \varepsilon_0^{d/d_0} C d^{-\alpha}$.

Even though Corollaries 40 and 41 give lower bounds on the node coverage, our simulation results indicate that the analysis is close to being tight (see Section 6.5.3). Compare the two strategies: to cover a certain fraction of the nodes, we need to bribe much fewer nodes in the case of HIGHEST-DEGREE than in UNIFORM-RANDOM. For example when $\alpha = 3$, if we bribe an f fraction of the nodes randomly, then we only need to bribe an f^2 fraction of the nodes using HIGHEST-DEGREE to get the same coverage. On the other hand, the bad news is that even if we have the power to choose highest degree nodes, a linear number of nodes need to be bribed to cover a constant fraction of the whole graph (the number of nodes to bribe is linear with n in both Corollaries).

6.5.2 Heuristic Analysis of Lookahead $l > 1$

Finally we consider a social network with lookahead $l > 1$, i.e., the attacker covers all nodes within distance l of a bribed node. As before we analyze the fraction of nodes f that need to be bribed for the attacker to get a constant $(1 - \varepsilon)$ coverage.

Our heuristic analysis shows that using the UNIFORM-RANDOM strategy, f needs to be approximately $\frac{-\ln \varepsilon_0}{d_0 b^l}$ for $1-\varepsilon$ coverage, where ε and ε_0 satisfy the equation in Lemma 38, b is of order lnn. For the strategy of HIGHEST-DEGREE, the attacker needs approximately $f = (\frac{-\ln \varepsilon_0}{d_0 b^l})^2$ (for $\alpha = 3$). Below is the analysis.

For simplicity we use $\alpha = 3$; the analysis can be generalized to any $\alpha > 2$.

Denote by B the set of bribed nodes; by $N_l(B)$ the set of nodes whose shortest distance to B is exactly l. Our goal is to estimate the number of nodes within distance l, denoted by $D_l(B) = |\bigcup_{0 \le i \le l} N_i(B)|$ – then we have $f = |B|/n$ where $D_l(B) = (1 - \varepsilon)n$.

Let us first assume $N_l(B)$ is small enough such that there is no loop, i.e. $\bigcup_{0 \le i \le l+1} N_i(B)$ is a forest rooted at B. In reality there may exist a few loops, but it does not introduce too much error in estimating $D_l(B)$ when $N_l(B)$ is very small. Under this assumption, $|N_l(B)|$ is much larger than all $|N_i(B)|$s ($i < l$), so we can use $|N_l(B)|$ as an approximation to $D_l(B)$. To compute $|N_l(B)|$, we first study the expansion rate from N_l to N_{l+1}, denoted by $b(l) = |N_{l+1}(B)|/|N_l(B)|$. Under the no-loop assumption, $b(l)$

equals to the average degree of nodes in $N_l(B)$ minus 1 (we need minus 1 to exclude the edges coming from $N_{l-1}(B)$). Note that nodes in $N_l(B)$ are not chosen uniformly at random; rather, they are chosen with probability proportional to their degrees, because of the random realization of the graph. Therefore the probability that such a node has degree x is proportional to $x * Cx^{-3}$, and consequently the expected degree of such node is

$$\frac{\sum_{x=d_0}^{\sqrt{n}} x * xCx^{-3}}{\sum_{x=d_0}^{\sqrt{n}} xCx^{-3}} \approx d_0 \ln \frac{\sqrt{n}}{d_0}$$

Thus we have the expansion rate $b = d_0 \ln \frac{\sqrt{n}}{d_0} - 1$, independent of l. It follows that $d_l(B) \approx |N_l(B)| \approx b|N_{l-1}(B)| \approx b^{l-1}|N_1(B)|$.

When $b * |N_l(B)|$ is large, we can no longer use the above assumption to estimate $|N_{l+1}(B)|$: we still have $b*|N_l(B)|$ edges incident to $N_{l+1}(B)$ but now some of the edges may share the same endpoints. This is the same as the weighted coupon collector problem in Lemma 38, so we can apply the result: if $b * |N_l(B)| = \frac{-\ln \varepsilon_0}{d_0} 2m$, then $|N_{l+1}(B)| \approx n(1-\varepsilon)$.

Now we compute the fraction of bribed nodes for $1-\varepsilon$ node coverage, i.e. compute $f = |B|/n$ where B satisfies $D_l(B) = n(1-\varepsilon)$. We need $b|N_{l-1}(B)| = \frac{-\ln \varepsilon_0}{d_0} 2m$ by Lemma 38, or $|N_{l-1}(B)| = \frac{-\ln \varepsilon_0}{d_0} 2m/b$. For large n, $b = \Theta(\ln n)$ is also large, so $|N_{l-1}(B)|$ is already small and we use the approximation $|N_{l-1}(B)| = b^{l-2}|N_1(B)|$. Thus we have $|N_1(B)| = \frac{-\ln \varepsilon_0}{d_0} 2m/b^{l-1}$. For the strategy of UNIFORM-RANDOM, $|N_1(B)| = \bar{d} f n$, so approximately we need $f = \frac{-\ln \varepsilon_0}{d_0 b^l}$. For the strategy of HIGHEST-DEGREE, $|N_1(B)| = 2\sqrt{f} d_0 n$ (given $\alpha = 3$), so we need $f = (\frac{-\ln \varepsilon_0}{d_0 b^l})^2$.

We can see that the number of nodes to bribe decreases exponentially with the lookahead distance. With lookahead $\ln \ln n$, bribing a constant number of nodes is enough to cover almost the whole graph.

6.5.3 Validating the Analysis with Simulation

We validate our theoretical analysis by simulation.

For lookahead 1, our theoretical analysis shows that for fixed node coverage the number of nodes to bribe is linear with the number of total nodes, i.e. f is a constant

with varying n. This confirms our simulation results in Section 6.4.1.

		UNIFORM-RANDOM		HIGHEST-DEG	
ε	ε_0	f_p	f_s	f_p	f_s
0.4	0.534	0.125	0.103	0.016	0.015
0.2	0.309	0.235	0.183	0.055	0.045
0.1	0.173	0.350	0.259	0.123	0.090

Table 6.2: Predicted values vs simulation results. We compute f for varying ε, with two bribing strategies. We compute f (1) by solving the equation in Corollary 40 and 41, shown in the column "f_p"; (2) by simulation, shown in the column "f_s". We use $\alpha = 3$ and $d_0 = 5$ in the table.

Next we check whether the f values predicted by Corollary 40 and 41 match the simulation results. We can see from Table 6.2 that the actual f values in simulation are smaller than those predicted in Corollary 40 and 41. This is because Theorem 39 gives a lower bound on the number of covered nodes. There are two factors causing underestimation: (1) the different trials cover uncovered nodes with higher probability; (2) we did not count the bribed nodes as covered. The second underestimation is more severe when the number of bribed nodes is not negligible compared to the number of covered nodes, especially when we use UNIFORM-RANDOM strategy. We can take into consideration the bribed nodes and refine our analysis. Using the same parameters in Table 6.2, for $\varepsilon = 0.4, 0.2, 0.1$, the refined predicted f for random bribing strategy are $0.110, 0.204, 0.305$ respectively, which are closer to the simulation results.

For lookahead $l > 1$, both the theoretical analysis and simulation results indicate that f decreases exponentially with the ability of lookahead l. The predicted values are not too far from the actual results, although not as close as in case of lookahead 1. For example, for UNIFORM-RANDOM with lookahead 2, to get 0.8-coverage ($\varepsilon = 0.2$), we predict $f = 0.0092$, while the simulation result is $f = 0.0145$.

6.6 Summary

In this chapter we provided both theoretical and experimental analysis of the vulnerability of a social network such as LinkedIn to a certain kind of privacy attack. We proposed several strategies for carrying out such attacks, and analyzed their success as a function of the lookahead permitted by the social network. We have shown that the number of user accounts that an attacker needs to subvert to obtain a fixed portion of the link structure of the network decreases exponentially with increase in lookahead permitted. We conclude that social networks interested in protecting their users' link privacy ought not to permit lookahead higher than 2 and may also want to create a separate privacy setting enabling users to opt out from displaying the number of connections that they have to anyone.

Chapter 7

Conclusions

In this thesis we described our research effort towards a better understanding of the Web. We focus on two major topics: (1) measuring the size of the Web and indexable web; (2) modeling the Web and social networks using webgraph models.

In the first part, we discussed the problem of estimating the Web size. We developed the first methods for estimating absolute index sizes of search engines assuming only access to their public query interface – the Random Document method and Independent Query Pool method. We validated our methods with synthetic data sets, and then applied them to estimate index sizes for major search engines. After presenting the empirical results, we then studied the problem from a different angle by mapping it to a classic theoretical problem of sum estimation, and proposed near optimal algorithms for this problem.

Our algorithms of estimating search index size still reply on certain assumptions and suffer from biases (see Section 2.6 for detailed discussion), so one future work is to design more accurate estimators, preferably with theoretical guarantee on the quality of estimator. It will also be of public interest to implement the existing algorithms and constantly monitor search index sizes and the indexable web size. There is already such effort: the website http://www.worldwidewebsize.com/ is keeping track of indexable web size and updating every day. However, their estimation algorithm is not very convincing: it is based on the algorithm in [BB98], with simplifications which further introduce biases; in particular, they trust the number of hits reported by

search engines for queries like "the", which is known to be highly unreliable especially for such queries with large numbers of hits.

In the second part, we presented our work on analyzing and applying random webgraph models. We first study searchability in random graphs: we gave a characterization of random graphs that are searchable with deterministic memoryless searching algorithms, and based on this characterization we proved a monotonicity result. Next, we analyzed graph properties of Stochastic Kronecker Graph model. Finally we studied link privacy in social networks using webgraph models; we formalized a particular attack on link privacy, and quantified how social networks' access control policy affects the complexity of attack.

Since the introduction of the first webgraph model by Barabási and Albert in 1999, a large number of models have been proposed in the last decade. But none of them can capture all the important properties of the Webgraph, and theoreticians are still in search for a "ultimate" model that incorporates all the nice properties of existing models. Also, given so many web graph models proposed, there is a need of systematic methods to evaluate and validate webgraph models.

An equally important direction is to apply webgraph models to studying real world applications, and this has received far less attention than deserved. Compared to the amount of research effort on this topic in academic community, there is very little appreciation about the power of webgraph models in industry. I believe more effort should be invested in advocating this nice theoretical tools to practitioners, and I look forward to more applications of webgraph models.

Bibliography

[ABC05] A. Anagnostopoulos, A. Z. Broder, and D. Carmel. Sampling search-engine results. In *Proceedings of the 14th International Conference on World Wide Web*, 2005.

[ACK+07] E. Arcaute, N. Chen, R. Kumar, D. Liben-Nowell, M. Mahdian, H. Nazerzadeh, and Y. Xu. Searchability in random graphs. In *Proceedings of the 5th Workshop On Algorithms And Models For The Web-Graph*, 2007.

[ACL00] W. Aiello, F. Chung, and L. Lu. A random graph model for power law graphs. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science*, 2000.

[ADLT05] N. Alon, N.G. Duffield, C. Lund, and M. Thorup. Estimating arbitrary subset sums with few probes. In *Proceedings of the 24th ACM SIGMOD -SIGACT-SIGART Symposium on Principles of Database Systems*, 2005.

[ALPH01] L. Adamic, R. Lukose, A. Puniyani, and B. Huberman. Search in power-law networks. *Physics Review E*, 2001.

[AMS99] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, pages 137–147, 1999.

[BA99] A. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(15):509–512, 1999.

[BB98] K. Bharat and A. Z. Broder. A technique for measuring the relative size and overlap of public web search engines. *Computer Networks*, 30(1-7):379–388, 1998.

[BB99] K. Bharat and A. Broder. Mirror, mirror, on the web: A study of host pairs with replicated content. *Computer Networks*, 31(11-16):1579–1590, 1999.

[BBCS05] N. Berger, C. Borgs, J. Chayes, and A. Saberi. The epidemic threshold in scale-free graphs. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2005.

[BBDH00] K. Bharat, A. Z. Broder, J. Dean, and M. R. Henzinger. A comparison of techniques to find mirrored hosts on the WWW. *Journal of the American Society for Information Science*, 51(12):1114–1122, 2000.

[BC78] E. A. Bender and E. R. Canfield. The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory, Series A*, pages 296–307, 1978.

[BC88] B. Bollobás and F. R. K. Chung. The diameter of a cycle plus a random matching. *SIAM Journal on Discrete Mathematics*, 1(3):328–333, 1988.

[BDK07] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, 2007.

[BFJ+06] A. Broder, M. Fontura, V. Josifovski, R. Kumar, R. Motwani, S. Nabar, R. Panigrahy, A. Tomkins, and Y. Xu. Estimating corpus size via queries. In *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge*, 2006.

[BI06] J. Bar-Ilan. Size of the web, search engine coverage and overlap – methodological issues. Unpublished, 2006.

[Bol90] B. Bollobás. The diameter of random graphs. *IEEE Transactions on Information Theory*, pages 285–288, 1990.

[Bro00] A. Z. Broder. Web measurements via random queries. Presentation at the Workshop on Web Measurement, Metrics, and Mathematical Models (WWW10 Conference), 2000.

[BYBC+00] Z. Bar-Yossef, A. Berg, S. Chien, J. Fakcharoenphol, and D. Weitz. Approximating aggregate queries about web pages via random walks. In *Proceedings of the 26th International Conference on Very Large Data Bases*, 2000.

[BYG06] Z. Bar-Yossef and M. Gurevich. Random sampling from a search engine's index. In *Proceedings of the 15th International World Wide Web Conference*, 2006.

[BYG07] Z. Bar-Yossef and M. Gurevich. Efficient search engine measurements. In *Proceedings of the 16th International World Wide Web Conference*, 2007.

[BYKS01] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Sampling algorithms: lower bounds and applications. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing*, 2001.

[CC01] J. Callan and M. Connell. Query-based sampling of text databases. *ACM Transactions on Information Systems*, 19(2):97–130, 2001.

[CCMN00] M. Charikar, S. Chaudhuri, R. Motwani, and V. Narasayya. Towards estimation error guarantees for distinct values. In *Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2000.

[CEG95] R. Canetti, G. Even, and O. Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, pages 17–25, 1995.

[CF03] C. Cooper and A. Frieze. A general model of web graphs. *Random Structures and Algorithms*, pages 311–335, 2003.

[CL01] F. Chung and L. Lu. The diameter of random sparse graphs. *Advances in Applied Mathematics*, pages 257–279, 2001.

[CL03] F. Chung and L. Lu. The average distance in a random graph with given expected degrees. *Internet Mathematics*, 1(1):91–114, 2003.

[CS04] G. Csanyi and B. Szendroi. Structure of a large social network. *Physical Review E*, 2004.

[DEM01] E. Drinea, M. Enachescu, and M. Mitzenmacher. Variations on random graph models of the web. *Harvard Computer Science Technical Report TR-06-01*, 2001.

[DHLS06] P. Duchon, N. Hanusse, E. Lebhar, and N. Schabanel. Could any graph be turned into a small world? *Theoretical Computer Science*, 355(1):96–103, 2006.

[DLT05] N.G. Duffield, C. Lund, and M. Thorup. Learn more, sample less: control of volume and variance in network measurements. *IEEE Transactions on Information Theory*, pages 1756–1775, 2005.

[ER59] P. Erdős and A. Rényi. On random graphs I. *Publications Mathematics Debrecen*, 6:290–297, 1959.

[Fra05] P. Fraigniaud. Greedy routing in tree-decomposed graphs. In *Proceedings of the 13th Annual European Symposium on Algorithms*, pages 791–802, 2005.

[GB01] G.Bianconi and A-L. Barabasi. Competition and multiscaling in evolving networks. *Europhysics Letters*, pages 436–442, 2001.

[GMS03] C. Gkantsidis, M. Mihail, and A. Saberi. Conductance and congestion in power law graphs. In *Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, 2003.

[GS05] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Proceedings of the 14th International Conference on World Wide Web (Special interest tracks & posters)*, 2005.

[HHMN00] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. On near-uniform URL sampling. *Computer Networks*, 33(1-6):295–308, 2000.

[KL81] V. Klee and D. Larmann. Diameters of random graphs. *Canadian Journal of Mathematics*, pages 618–640, 1981.

[Kle00] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proceedings of the 32nd ACM Symposium on Theory of Computing*, 2000.

[Kle01] J. Kleinberg. Small-world phenomena and the dynamics of information. *Advances in Neural Information Processing Systems*, 14:431–438, 2001.

[KLNT06] R. Kumar, D. Liben-Nowell, and A. Tomkins. Navigating low-dimensional and hierarchical population networks. In *Proceedings of the 14th Annual European Symposium on Algorithms*, pages 480–491, 2006.

[KMN08] A. Korolova, R. Motwani, and S. Nabar. Link privacy in social networks. In *Proceedings of the 24th International Conference on Data Engineering*, 2008.

[KRR+00] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science*, 2000.

[KS96] D. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the ACM*, pages 601–640, 1996.

[LCKF05] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos. Realistic, mathematically tractable graph generation and evolution, using Kronecker multiplication. *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 133–145, 2005.

[LF07] J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *Proceedings of the 24th Annual International Conference on Machine Learning*, 2007.

[LG98] S. Lawrence and C. L. Giles. Searching the world wide web. *Science*, 280(5360):98–100, 1998.

[LG00] S. Lawrence and C.L. Giles. Accessibility of information on the web. *Intelligence*, 11(1):32–39, 2000.

[Liu96] J. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistical Computing*, pages 113–119, 1996.

[LKF05] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. In *Proceedings of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005.

[LNNK+05] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *Proceedings of the National Academy of Sciences*, 102(33):11623–11628, 2005.

[LYM02] K. Liu, C. Yu, and W. Meng. Discovering the representative of a search engine. In *Proceedings of the 11th International Conference on Information and Knowledge Management*, pages 652–654, 2002.

[Mil67] Stanley Milgram. The small world problem. *Psychology Today*, 1:61–67, 1967.

[MK] Y. Medynskiy and J. Kaye. Characterizing livejournal: Sccs, liverank, and six degrees. *Technical Report*.

[MMJ+07] M.Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava. Anonymizing social networks. *University of Massachusetts, Amherst Technical Report*, 2007.

[MPX07] R. Motwani, R. Panigrahy, and Y. Xu. Estimating sum via weighted sampling. In *Proceedings of the 34th International Colloquium on Automata, Languages and Programming*, 2007.

[MR95] R. Motwani and P. Raghavan. *Randomized algorithms*. Cambridge University Press, New York, NY, USA, 1995.

[MST07] M. Mihail, A. Saberi, and P. Tetali. Random walks with lookahead in power law random graphs. *Internet Mathematics*, pages 147–152, 2007.

[MX06] R. Motwani and Y. Xu. Evolution of page popularity under random web graph models. In *Proceedings of the 25th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, 2006.

[MX07] M. Mahdian and Y. Xu. Stochastic kronecker graphs. In *Proceedings of the 5th Workshop On Algorithms And Models For The Web-Graph*, 2007.

[PRU02] G. Pandurangan, P. Raghavan, and E. Upfal. Using pagerank to characterize web structure. In *Proceedings of the 8th Annual International Conference on Computing and Combinatorics*, 2002.

[RPLG01] P. Rusmevichientong, D. M. Pennock, S. Lawrence, and C. Giles. Methods for sampling pages uniformly from the world wide web. In *Proceedings of the Fall Symposium on Using Uncertainty Within Computation*, 2001.

[Sel99] E. Selberg. Towards comprehensive web search. *PhD thesis, University of Washington*, 1999.

[Sli05] A. Slivkins. Distance estimation and object location via rings of neighbors. In *Proceedings of the 16th ACM Symposium on Principles of Distributed Computing*, pages 41–50, 2005.

[Sze06] M. Szegedy. The dlt priority sampling is essentially optimal. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, 2006.

[vR79] C. J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 1979.

[vS54] H. von Schelling. Coupon collecting for unequal probabilities. *American Mathematical Monthly*, 61:306–311, 1954.

[WF94] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*. Cambridge University Press New York, NY, USA, 1994.

[WGC03] S. Wu, F. Gibb, and F. Crestani. Experiments with document archive size detection. In *Proceedings of the 25th European Conference on IR Research*, 2003.

[WS98] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[YS07] S. Young and E. Scheinerman. Random dot product graph models for social networks. In *Proceedings of the 5th Workshop On Algorithms And Models For The Web-Graph*, 2007.

[ZG07] E. Zheleva and L. Getoor. Preserving the privacy of sensitive relationships in graph data. *Proceedings of the International Workshop on Privacy, Security and Trust in KDD*, 2007.