



University of
Massachusetts
Amherst

On the Design of Methods to Estimate Network Characteristics

Item Type	Dissertation (Open Access)
Authors	Ribeiro, Bruno F.
DOI	10.7275/1562209
Download date	2026-04-22 23:46:11
Link to Item	https://hdl.handle.net/20.500.14394/38659

**ON THE DESIGN OF METHODS TO ESTIMATE
NETWORK CHARACTERISTICS**

A Dissertation Presented

by

BRUNO F. RIBEIRO

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

DOCTOR OF PHILOSOPHY

May 2010

Department Computer Science

© Copyright by Bruno F. Ribeiro 2010

All Rights Reserved

ON THE DESIGN OF METHODS TO ESTIMATE NETWORK CHARACTERISTICS

A Dissertation Presented

by

BRUNO F. RIBEIRO

Approved as to style and content by:

Donald F. Towsley, Chair

Jean Bolot, Member

David Jensen, Member

Patrick Kelly, Member

James F. Kurose, Member

Andrew G. Barto, Department Chair
Department Computer Science

ACKNOWLEDGMENTS

I would like to dedicate this thesis to my mother, Dr. Neyde Felisberto Martins Ribeiro (Zuca), who passed away on October, 6th, 2007. My mother bestowed the love for science upon me. Her values are a moral compass that I always strive to follow. Our Zuca will always be with my sisters and I in spirit.

This thesis would not have been possible without the support of many people. I want to express my love and gratitude to my sisters, Aline and Flávia, to my father José Roberto, and to the rest of my family in Brazil (from Recife to Rio de Janeiro). I cannot forget my “family”, namely Lucia, Fred, Renata, and Luciana.

I would like to express my deepest gratitude to my advisor Prof. Don Towsley, who was abundantly helpful and offered invaluable assistance and guidance. I also owe my deepest gratitude to Prof. Towsley for supporting me during the time I was in Brazil looking after my mother. I also would like thank the members of my thesis committee, Prof. James Kurose, Dr. Jean Bolot, Prof. Patrick Kelly, and Prof. David Jensen. I also convey thanks to my Master’s advisor Prof. Edmundo de Souza e Silva who has helped me in many ways.

I would like to thank the members of the Networks lab. Special thanks to my friend George Konidakis with whom I shared my culture shock experiences and to the IHCS (Marc and Dirk) for making “136” a home. Special thanks are also in order to Christine, Daniel², Guto, Antonio do Aloe, Paula, Emily, Andre, Jacque, Bruno, Marcelo, Joy, Ana Paula, and all my friends.

ABSTRACT

ON THE DESIGN OF METHODS TO ESTIMATE NETWORK CHARACTERISTICS

MAY 2010

BRUNO F. RIBEIRO

B.Sc., FEDERAL UNIVERSITY OF RIO DE JANEIRO

M.Sc., FEDERAL UNIVERSITY OF RIO DE JANEIRO

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Donald F. Towsley

Social and computer networks permeate our lives. Large networks, such as the Internet, the World Wide Web (WWW), AND wireless smartphones, have indisputable economic and social importance. These networks have non-trivial topological features, i.e., features that do not occur in simple networks such as lattices or random networks. Estimating characteristics of these networks from incomplete (sampled) data is a challenging task.

This thesis provides two frameworks within which common measurement tasks are analyzed and new, principled, measurement methods are designed. The first framework focuses on sampling directly observable network characteristics. This framework is applied to design a novel multidimensional random walk to efficiently sample loosely connected networks. The second framework focuses on the design of measurement

methods to estimate indirectly observable network characteristics. This framework is applied to design two new, principled, estimators of flow size distributions over Internet routers using (1) randomly sampled IP packets and (2) a data stream algorithm.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	iv
ABSTRACT	v
LIST OF TABLES	xi
LIST OF FIGURES	xii
 CHAPTER	
1. INTRODUCTION	1
1.1 Framework contribution	3
1.2 Thesis Outline	5
1.3 Extended Overview	7
1.3.1 Estimating Graph Characteristics with Multidimensional Random Walks	7
1.3.2 Designing TCP flow-level estimators from sampled packets	8
1.3.3 Designing a streaming algorithm	10
2. BACKGROUND	11
2.1 Notation	11
2.2 Basic Notions of Estimation Theory	12
2.3 Unconstrained Fisher information	12
2.3.1 Fisher information of n independent samples	13
2.3.2 Unconstrained Cramér-Rao inequality	13
2.4 Data processing inequality	16
2.5 Random walk sampling	16
2.5.1 Notation	16
2.5.2 Single random walker	17

2.5.3	Spectral decomposition of a random walk.	18
2.5.4	Stationary random walks.	19
3.	ESTIMATING GRAPH CHARACTERISTICS WITH MULTIDIMENSIONAL RANDOM WALKS	20
3.1	Introduction	20
3.1.1	Contributions	22
3.1.2	Outline	23
3.2	Vertex v.s. edge sampling	24
3.3	Random walk sampling	25
3.3.1	Strong Law of Large Numbers	26
3.3.2	Estimators	27
3.3.2.1	Label Density	27
3.3.2.2	Global Clustering Coefficient	28
3.3.3	Estimator Accuracy & Graph Structure	31
3.3.4	Multiple Independent Random Walkers	32
3.4	Frontier sampling	33
3.5	Results	37
3.5.1	In- and Out-degree Distribution Estimates	38
3.5.2	Frontier v.s. Random Sampling	42
3.5.3	Density of Special Interest Groups	45
3.5.4	Global Clustering Coefficient Estimates	45
3.5.5	Convergence to Stationarity	47
3.6	Distributed Frontier Sampling	48
3.7	Related work	51
3.8	Discussion and Future work	52
4.	DESIGNING TCP FLOW-LEVEL ESTIMATORS FROM SAMPLED PACKETS	54
4.1	Introduction	54
4.2	Estimating the TCP flow size distribution from randomly sampled packets	55
4.3	Contributions	57
4.4	Fisher information from sampled packets	57
4.5	Simplifications to constrained Cramér-Rao inequality	60
4.6	Designing summary functions	60

4.6.1	Real Internet Traces	61
4.6.2	No protocol information	62
4.6.3	TCP SYN flag information	63
4.6.4	TCP SEQ + SYN flag information	63
4.7	Simulation results	66
4.7.1	MLE with conjugate gradients	66
4.7.2	Results	68
4.7.3	MLE for SEQ+SYN summary: an efficient estimator.....	68
4.8	Conclusions	69
5.	DESIGNING A STREAMING ALGORITHM	71
5.1	Introduction	71
5.2	A naive algorithm	72
5.3	An algorithm without collision resolution	73
5.3.1	Counter independence	74
5.3.2	The likelihood function	76
5.3.3	The Fisher information	76
5.3.4	The Cramér-Rao inequality	77
5.3.5	A fast estimator.....	80
5.4	Further improvements: reducing the memory footprint.....	81
5.5	Overview of the measurement method	82
5.6	Measurement method description	84
5.6.1	Data structures	84
5.6.1.1	Sketch	85
5.6.1.2	Sketch histogram.....	86
5.6.1.3	Pseudo-random auxiliary counters	86
5.6.2	Histogram estimator	87
5.6.2.1	Estimates of large flows sizes ($\geq k$)	88
5.7	Evaluation.....	89
5.8	Related work	91
5.9	Conclusions & Contributions	93
6.	CONCLUSIONS & FUTURE WORK.....	95

APPENDIX: DESIGNING TCP FLOW-LEVEL ESTIMATORS FROM SAMPLED PACKETS	97
A.1 An approximation to $h(s_{min}, s_{max})$	97
A.2 Designing a streaming algorithm	101
A.2.1 Pseudo-random counting	101
A.2.2 Counter increment probability	102
A.2.3 Flow collision function	102
 BIBLIOGRAPHY	 103

LIST OF TABLES

Table	Page
3.1	Summary of the graph datasets used in the simulations. “Size of LCC” refers to the size of the largest connected component. 37
3.2	Frontier sampling: global clustering coefficient estimates. C is the true value of the global clustering coefficient and \hat{C} is its estimated value. 46
3.3	Relative worst-case difference between the steady state and the transient edge sampling probabilities after $B - K$ steps. Frontier edge sampling probabilities are closer to steady state in all graphs. Legend: (FS) = Frontier sampling ($K = 10$), (SRW) = Single ($K = 1$) Random Walker, and (MRW) = Multiple ($K = 10$) Random Walkers. 50
4.1	Trace Statistics 62
4.2	Minimum number of <u>sampled flows</u> that an unbiased estimator needs in order to achieve $\sqrt{E[(\hat{\theta}_1 - \theta_1)^2]} < 0.5$. Results for $w = 50$, $p = 1/200$, obtained with the flow size distribution of the BB-East-2 trace. 65

LIST OF FIGURES

Figure	Page
1.1 Schematics of a measurement method.	3
3.1 Illustration of the Markov chain associated to the Frontier sampler with dimension $m = 2$	34
3.2 (Flickr) Log-log plot of the in-degree CCDF.	39
3.3 (LCC of Flickr) The log-log plot of the NMSE of the in-degree distribution estimates with budget $B = V /100$	40
3.4 (Flickr) The log-log plot of the NMSE of the in-degree distribution estimates with budget $B = V /100$	41
3.5 (LCC of Flickr) Four sample paths of $\hat{\theta}_1$ ($\theta_1 = 0.53$) as a function of the number of steps n (horizontal axis in log scale).	42
3.6 (Livejournal) Log-log plot of the out-degree CCDF.	43
3.7 (Livejournal) The log-log plot of the NMSE of the out-degree distribution estimation with sampling budget $B = V /10$ (MSE over 10,000 runs).	44
3.8 (G_{AB} graph) Four paths of $\hat{\theta}_{10}$ as a function of the number of steps n ($\theta_{10} = 0.024$).	45
3.9 (G_{AB} graph) The log-log plot of the NMSE of the degree distribution estimation with sampling budget $B = V /10$ (MSE over 10,000 runs).	46
3.10 (Flickr) The log-log plot shows the NMSE of the in-degree distribution estimation with budget $B = V /100 = 18612$ (MSE over 10,000 runs).	47
3.11 (Livejournal) The log-log plot shows the NMSE of the in-degree distribution estimation with budget $B = V /100 = 52844$ (MSE over 10,000 runs).	48

3.12	(Flickr) The NMSE of the density estimates of the most popular groups in the Flickr graph.	49
4.1	Inverse Fisher information $(\mathcal{I}^+)_{ii}$ (i is the flow size) without protocol information.	63
4.2	Inverse Fisher information $(\mathcal{I}^+)_{ii}$ (i is the flow size) with TCP SYN protocol information.	64
4.3	Inverse Fisher information $(\mathcal{I}^+)_{ii}$ (i is the flow size) with TCP SYN and TCP SEQ protocol information.	65
4.4	Estimates from 120 runs with 5×10^9 sampled flows and $p = 1/200$. Summary function protocol information: TCP SYN flag against TCP SEQ+SYN flag. Note the strange behavior of the estimates from the SYN flag summary. This happens due to the low Fisher information in the sampled flows.	69
4.5	This graph compares the root mean squared error of the MLE estimate with the inverse of the Fisher information (which, according to the Cramér-Rao inequality, is a bound for the mean squared error of the MLE).	70
5.1	Schematics of a measurement method.	72
5.2	The figure plots the inverse Fisher information with varying maximum counter values ($k_{max} = 2^b - 1$).	78
5.3	Accuracy of the best packet sampling estimator. Inverse Fisher information $(\mathcal{I}^+)_{ii}$ (i is the flow size) with TCP SYN and TCP SEQ protocol information.	79
5.4	Multiplexing a sketch. One extra bit is used to store ownership in the physical sketch. Note that counters with index $\leq M$ in the virtual sketch always win contentions against counters with index $> M$	83
5.5	Histogram estimates with 8MB of memory. BB-East-2 trace histogram (line) v.s. histogram estimates (with a virtual sketch and with a regular sketch). Experiment: 9.6 million flows (average), 6 bit counters (7 bits <i>per</i> flow), 37 runs.	90

5.6	Histogram estimates with 16MB of memory. Exponential histogram (line) with $\lambda_i = 2 \times 10^6 e^{-i/10^4} / 9999.5$ v.s. histogram estimates (95% confidence interval is too small). Experiment: 20 million flows (average), 6 bit counters (7 bits <i>per</i> flow), 43 runs.	92
A.1	Number of sampled flows with label $(S, r), r \geq 1$ obtained from both h drawn synthetically, and \tilde{h} obtained using the real sampled trace. Results from the BB-East-2 trace. Packet sampling rate $p = 0.01$. This graph shows $nd_{(N,r)}^{\hat{}}$, the number of sample tuples (N, r) (from flows without a SYN sampled packet). Notice that the average is slightly overestimated.	98
A.2	Number of sampled flows with label $(S, r), r \geq 1$ obtained from both h drawn synthetically, and \tilde{h} obtained using the real sampled trace. Results from the BB-East-2 trace. Packet sampling rate $p = 0.01$. This graph shows $nd_{(S,r)}^{\hat{}}$, the number of sample tuples (S, r) (from flows with a SYN sampled packet). Notice that the average is slightly underestimated.	99

CHAPTER 1

INTRODUCTION

Social and computer networks permeate our modern lives. Large networks, such as the Internet, the World Wide Web (WWW) [59], Facebook [21], wireless smartphones [2], have indisputable economic and social importance. These networks have non-trivial topological features, i.e., features that do not occur in simple networks such as lattices or random networks, and are also known as *complex networks*. Inspired largely by empirical studies of the characteristics of real-world complex networks, the flourishing field of network science seeks to advance our knowledge of the structure and behavior of such systems. Unfortunately, obtaining characteristics of a large complex network from incomplete (sampled) data is a challenging task. As a result, principled and accurate studies of these systems depend heavily on principled and accurate estimates of the characteristics of these networks.

Estimating characteristics, such as the degree distribution from a unknown network graph using incomplete data can be a challenging task. Only under certain conditions such task can be considered easy. Consider estimating the degree distribution, $\theta_1, \dots, \theta_w$, of a network graph, where θ_i is the fraction of vertices with degree i and w is the maximum degree of a vertex. Let $V' = \{v_1, \dots, v_n\}$ be a set of n sampled vertices. Assume that the vertices in V' are sampled independently (uniformly and with replacement). Furthermore, assume that we are able to directly query the degree of vertex v , denoted $\deg(v)$. Using the two above assumptions we can build a simple unbiased estimator of θ_i :

$$\hat{\theta}_i = \sum_{i=1}^n \deg(v_i)/n.$$

In general, however, the above assumptions do not hold and estimating graph characteristics can be a challenging task. For instance, we may need to relax the independent sampling assumption due to the cost associated to sampling vertices independently. This is a common scenario in online social networks, such as Facebook and MySpace, in which user profiles are associated to unique numerical IDs and the ID space is sparsely populated. User profiles are sampled uniformly by querying randomly generated numerical IDs but when the ID space is sparsely populated most ID queries return non-existing user accounts. In such scenario it is more practical to sample vertices using methods that do not guarantee independence, e.g., a random walk (RW). A RW samples a graph by moving a particle (walker) from a vertex to a neighboring vertex (through an edge). By this process edges and vertices are sampled. Vertices sampled by a random walker are not independent. This dependence may significantly increase the estimator error (variance), specially if the graph is loosely connected.

Another assumption that may not hold in general when we seek to characterize distributions over the graph is the assumption that querying a vertex returns a sample of the distribution of interest. In the example above we estimate the degree distribution by directly querying vertex degrees. In some networks directly querying the characteristic of interest may not be feasible, e.g., the query may not return the vertex degree but rather a random subset of its edges. In this case, the vertex degree is a latent variable and the degree distribution must be estimated from a mathematical model that correlates the observed edge subsets with the original degree distribution. A latent network characteristic is a network characteristic that needs to be estimated from a mathematical model that uses a summary of the (sampled) direct observations as input.

This thesis presents two main contributions:

- A framework within which common measurement tasks are analyzed and new, principled, measurement methods are designed. In this thesis the term *measurement method* refers to the way in which the (incomplete) data is collected from the original network and how the network characteristics are estimated from this (incomplete) data. More importantly, this framework is a powerful tool to analyze measurement methods that seek to estimate latent network characteristics, avoiding common caveats and mistakes observed in previous works.
- A new multi-dimensional random walk-type sampling process that, different from existing random walk sampling processes, reduce the estimation errors (variance) caused by loosely connected graphs.

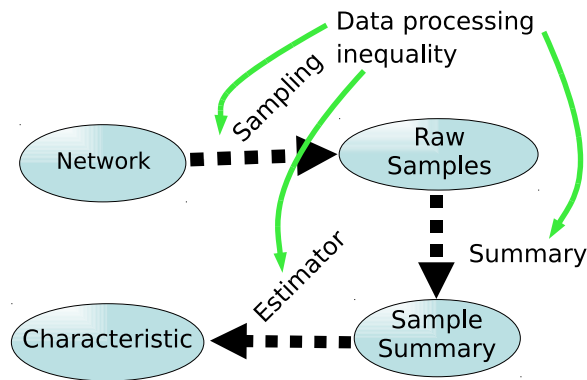


Figure 1.1. Schematics of a measurement method.

1.1 Framework contribution

In this thesis the term *measurement method* refers to the way in which the (incomplete) data is collected from the original network and how the network characteristics are estimated from this (incomplete) data. Figure 1.1 shows a schematic of a measurement method, which can be divided into three steps:

- *Collecting the raw samples:* raw samples are collected from the network using a sampling process. For instance sampling individuals independently and uni-

formly at random from a population. These raw samples can be, for instance, the logs of HTTP queries over an on-line social network website such as Facebook or the IP and TCP headers of Internet packets. The raw samples may or may not be stored.

- *Summarizing the samples:* the raw samples are then processed by a function that outputs a *sample summary*. The *sample summary* are the observed network characteristics and can be, for instance, how many “friends” a user has on his/hers on-line social network account (e.g., Facebook) or the number of sampled IP packets that belong to the same IP flow (independent of what the definition of an IP flow is, e.g., all packets with the same destination IP). The sample summary act as an input to the next step (estimation).
- *Estimation:* In the last step, an *estimator* is used to obtain the estimated graph characteristic. An estimator is a function that takes a summary of the observations (sampled data) as input and outputs an estimate of a unknown population parameter (graph characteristic). The estimator requires a model that describes the (statistical) relationship between the network characteristics and the summarized data.

In each of the above steps, our ability to accurately estimate the original network characteristics is governed by the *data processing inequality*. The data processing inequality, whose formal *estimation* theoretic definition is given in Section 2.4 (which is different than the more common information theoretic definition found in information theory books such as Cover and Thomas [16]), states that *processing* the data can never increase the amount of statistical information (with respect to the characteristic being estimated) already contained in the data.

This simple inequality has a profound impact in the design of measurement methods. It states that the statistical information lost in the early steps of the measure-

ment method cannot be recovered by later steps. For instance, if the sample summary (when combined with any available side information) does not contain enough statistical information to obtain accurate estimates, then *no estimator* applied to it can correct that: *we need either to find a better summary function or to find a better sampling process*. Using this insight, this thesis focuses on the design of the early steps of a measurement method, rather than focusing on estimation phase.

With help of Figure 1.1 it is easy to understand recurrent mistakes in the literature. Namely, the degree distribution of the Internet router-level graph used in Faloutsos et al. [22] is, in fact, the degree distribution of the “Sample Summary”, not the degree distribution of the original graph. Similarly, the degree distribution of the Web graph obtained by Broder et al. [9] and the network characteristics presented in Mislove et al. [50], Leskovec and Faloutsos [42], Yoon et al. [67], among others [17, 41] are all characteristics of the *sample summary*, not *estimated* characteristics of the original graph.

Another good use of the proposed framework is to resolve known conflicting results in the literature. For instance, Hohn and Veitch [34] and Duffield et al. [19] seek to estimate the flow size distribution of IP flows (a latent network characteristic). Using the data processing inequality and the Fisher information (introduced in Chapter 2), Chapter 4 shows that the *sample summary* obtained by both of these works does not have enough statistical information to obtain accurate estimates of the original flow size distribution. This implies that the good results presented in Duffield et al. [19] are likely to be the product of an evaluation error.

1.2 Thesis Outline

This thesis provides a framework in which common measurement tasks are analyzed and new, principled, measurement methods are designed. In what follows I

present a quick summary of this thesis. An extended version of this summary is found in Section 1.3.

- The necessary **background** is found in Chapter 2.
- **Estimating simple topological characteristics of a network** (Chapter 3):
This chapter focuses on the collection of “Raw Samples” from the network. This chapter compares independent random vertex sampling, independent random edge sampling, and random walks. Random walks are, arguably, one of the most important and widely used sampling methods in complex networks. However, problems arise when sampling a loosely connected graph with a random walk. Therefore, this chapter emphasizes the design and evaluation of a novel random walk sampling process, called *Frontier sampling* (FS), that mitigates these problems. We also see, analytically, why FS is preferable to random (uniform) vertex sampling for estimating the degree distribution tail of power law graphs.
- **Estimating latent traffic characteristics of a network:**
 - Chapter 4 presents a **measurement framework** used to **design better and more principled measurement methods**. The framework developed in this chapter can be broadly applied to a variety of measurement problems, and has already been successfully applied to the problem of estimating Internet flow size distribution from sampled packets [57, 63] and also to the Traffic Matrix estimation problem [65]. Chapter 4 presents an application to the TCP flow size estimation, and provides a definitive answer to the debate between Hohn and Veitch [34] and Duffield et al. [19] to whether it is possible to find accurate flow size distribution estimates from sampled packets without protocol information.
 - Chapter 5 **designs a fast streaming algorithm** to estimate the flow size histogram inside an Internet router. Here we can see that a principled

design can drastically reduce the resource requirements of the measurement method while retaining most of the statistical information about the flow sizes intact.

- **Conclusions and future work** are presented in Chapter 6.

1.3 Extended Overview

In what follows I expand on the description of the three main chapters of this thesis.

1.3.1 Estimating Graph Characteristics with Multidimensional Random Walks

Chapter 3 compares independent random vertex sampling, independent random edge sampling, and random walks. The advantages and disadvantages of random vertex sampling are contrasted with random edge sampling and random walks. Random walks are one of the most used and studied measurement methods for complex networks. Successful applications of random walk-based estimators can be found over a variety of interesting problems: sampling individuals in a population (connected through a social network) in order to obtain vertex-oriented graph characteristics [32, 60, 64] (e.g., estimation of HIV seroprevalence among drug users [48]), estimation of content prevalence in peer-to-peer networks [30, 47, 54, 61], estimation of degree distributions of the Facebook on-line social graph [29], uniformly sampling Web pages from the Internet [33, 59], and uniformly sampling Web pages from a search engine’s index [4], just to name a few examples.

Despite increasing interest in random walks and despite its numerous applications, measurement methods that use random walk-based sampling tend to have large estimation errors when sampling disconnected or loosely connected graphs. Chapter 3 presents a novel random walk-based sampling method: *Frontier sampling* (FS). FS

uses m *dependent* random walkers to mitigate the estimation errors induced by disconnected or loosely connected subgraphs. Let G denote the original graph. FS can be seen as single (m -dimensional) random walk over G^m , the m -th Cartesian power of G (G^m is formally defined in Chapter 3). FS is designed such that any estimator designed to work with samples obtained by a regular random walker can also work with FS without requiring any modifications.

1.3.2 Designing TCP flow-level estimators from sampled packets

Chapter 4 presents a **measurement framework** used to **design better and more principled measurement methods**. The framework developed in Chapter 4 can be broadly applied to a variety of measurement problems, and has already been successfully applied to the problem of estimating Internet flow size distribution from sampled packets [57, 63] and to the Traffic Matrix estimation problem [65]. This framework is illustrated with the problem of estimating the TCP flow size distributions from randomly sampled packets. The TCP flow size distribution is defined as the fraction of TCP flows that contains $i = 1, 2, \dots$ packets. It is one of the Internet traffic characteristics that garners the most interest from network operators as it can be used for traffic engineering, denial of service attack monitoring, and worm/virus outbreak detection. This is a difficult problem as random packet sampling affects the flow size distribution: large flows are more likely to be sampled than shorter flows and most small flows will not be sampled at all. We seek answers to the following open questions:

- *Is it possible to accurately estimate the original TCP flow size distribution from randomly sampled packets?*
- *Are there other observable characteristics (such as TCP protocol information) that can be extracted from the sampled packets and could improve the accuracy of the estimates?*

The literature concerning TCP flow size distribution estimation contains inconclusive answers to the questions above. Using the same sampling framework the following two works, both published in 2003, arrive at distinct conclusions:

- Hohn and Veitch [34] prove that the *inversion estimator* is not capable of accurately estimating the fraction of flows if the flow size is small. It is important to note that the inversion estimator is not necessarily optimal. While this is an interesting negative result, it does not imply that another estimator (e.g., Maximum Likelihood Estimator) would not be able to be accurate.
- Duffield, Lund, and Thorup [19] argue that the maximum likelihood estimator, in practice, can do a good job at estimating the fraction of flows of any size. Furthermore, Duffield et al. [19] argue that the *inversion estimator* used by Hohn and Veitch [34] is unfit for the task as its variance is too high.

In 1947 R.A. Fisher believed that, because of the statistical tools he helped to develop, this type of debate was all but settled [24]. In Chapter 4 we revisit these tools and argue that this part of Fisher’s lifetime work has been all but ignored in the Computer Science literature, and that it has profound implications in the way we perform network measurements. The debate between Hohn and Veitch [34] and Duffield et al. [19] makes TCP flow size estimation a good example for Chapter 4. Using the Fisher information and other information theoretic results (described in detail in Chapter 2), Chapter 4 argues that there is no (practical) unbiased estimator that can accurately estimate the flow size distribution of small flows using the summary functions in Hohn and Veitch [34] and Duffield et al. [19]. Moreover, I show that a simple change to their measurement method (adding the TCP Sequence Number information in the samples) is sufficient to yield enough information to accurately estimate the distribution.

1.3.3 Designing a streaming algorithm

Chapter 5 revisits the flow size estimation problem. Here, however, the focus is on combining the sampling and the summary function in order to reduce the memory footprint and processing power required by the measurement method. The ideas developed in Chapter 4 are used to design an improved measurement method by: (1) eliminating statistically irrelevant information being stored (thus reducing its memory requirements), and (2) drastically reducing the time to estimate the flow size distribution. The measurement method described in Chapter 5 is efficient enough to perform on-line estimation inside the router.

The next chapter covers some of the background needed in this thesis.

CHAPTER 2

BACKGROUND

Here we revisit the important estimation theory concepts that are used in the remaining chapters of this thesis.

2.1 Notation

The following notation is used in the remainder of this thesis.

- Random variables are denoted by capital letters, e.g., D .
- Matrices are denoted by bold capital letters, e.g., \mathbf{B} .
- ∇_d is the vector differential operator with respect to the variables $d = (d_1, \dots, d_k)$,

$$\nabla_d f(d) = \left(\frac{\partial f(d)}{\partial d_1}, \dots, \frac{\partial f(d)}{\partial d_k} \right),$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is differentiable everywhere in the domain of d .

- \top is the matrix transpose operator, i.e., \mathbf{B}^\top is the transpose of \mathbf{B} .
- $G(V, E)$ denotes a undirected labeled graph, where V is the set of vertices of G , E is the set of edges of G which induces a symmetric binary relation (\cdot, \cdot) among the vertices in V , and $\mathcal{L}(v)$ and $\mathcal{L}(e)$ are the set of labels associated to a vertex $v \in V$ and an edge $e \in E$, respectively.

2.2 Basic Notions of Estimation Theory

Estimating characteristics of a complex network requires designing a measurement method consisting of a: sampling process, summary function, and estimator. Often the characteristic of interest cannot be directly observed. For instance, when we sample TCP packets in an Internet router, the size of the flow from which the packet belongs to is a characteristic that cannot be directly observed from the (incomplete) set of sampled packets. The flow size has to be estimated. A directly *observable* characteristic is a characteristic that can be directly observed from the network. For instance, the number of friends in a Facebook [21] profile is an observable characteristic of a Facebook profile, as the true value can be directly queried from Facebook servers. A *latent* characteristic is a characteristic that cannot be directly observed but need to be estimated from other observable characteristics. An example of a latent characteristic is the number of packets in a TCP flow when we can only observe a subset of the packets in the TCP flow.

In the case of latent characteristics we need a model that correlates them with the observed data. More formally, if $D = (D_1, \dots, D_n)$ is a sequence of random variables that describes the observable characteristics of the sampled (incomplete) data and θ is a vector of the latent characteristics of interest, the model is defined as $P[D = d | \theta]$, the probability that, given the latent characteristics θ , we sample the observable characteristic d . $P[D = d | \theta]$ is also known as the *likelihood function*. The necessity of such a model is an intrinsic requirement, as one needs $P[D = d | \theta]$ to be able to assess how the latent characteristics affect the observable ones.

2.3 Unconstrained Fisher information

The Fisher information (named after R.A. Fisher) can be thought of as the amount of information that a set of k samples, $D = (D_1, \dots, D_k)$, carries about a set of parameters $\theta = (\theta_1, \dots, \theta_n)$ upon which the probability distribution of the samples

depends. Here we assume that the parameters have no constraints, i.e., $\theta \in \mathbb{R}^n$. Later we see how the Fisher information increases when we add constraints to θ . The Fisher information is defined over a set of samples.

In general, the unconstrained Fisher information is a matrix $J(D) = [J_{ij}(D)]$ where

$$J_{ij}(D) \triangleq E \left[\frac{\partial \ln P[D | \theta]}{\partial \theta_i} \cdot \frac{\partial \ln P[D | \theta]}{\partial \theta_j} \right]. \quad (2.1)$$

Alternatively, we can write J in matrix notation

$$J(D) \triangleq E \left[(\nabla_{\theta} \ln P[D | \theta]) (\nabla_{\theta} \ln P[D | \theta])^{\top} \right], \quad (2.2)$$

where A^{\top} denotes the transpose of matrix A and ∇ is the vector differential operator. In what follows we look at some interesting characteristics of the Fisher information.

2.3.1 Fisher information of n independent samples

Lemma 2.3.1. *Let J be the Fisher information of one sample. The Fisher information of a set of n independent samples is nJ .*

Proof. This is a well known result. It comes from the fact that the joint likelihood of two independent samples D_1 and D_2 is equal to the likelihood of D_1 times the likelihood of D_2 . □

2.3.2 Unconstrained Cramér-Rao inequality

The most notable property of the Fisher information is a bound on the accuracy of estimators. The Cramér-Rao theorem states that the mean squared error of any unbiased estimator is lower bounded by the inverse of the Fisher information, provided some weak regularity conditions are met (Hájék regularity is needed [35]). Let T be

an estimator of θ (T is a function such that $T(d)$ outputs an estimate of θ). T is a unbiased estimator if $E[T(D)] = \theta$. Then, if T is **unbiased**:

$$E[(T(D)_i - \theta_i)^2] \geq (J(D)^{-1})_{ii}, \quad (2.3)$$

where $J(D)^{-1}$ is the inverse of the Fisher information matrix. The Cramér-Rao inequality (eq. (2.3)) states that the mean square error of any unbiased estimator that uses the samples D must be greater than or equal to the inverse of the Fisher information.

The constrained Cramér-Rao inequality

The parameters θ may have constraints that reduce the uncertainty about the value of θ in the estimator. Here we consider the following constraints:

$$0 < \theta_i < 1, \forall i \quad (2.4)$$

and

$$\sum_{\forall i} \theta_i = 1. \quad (2.5)$$

The Cramér-Rao inequality seen in Section 2.3.2 (and found in most textbooks, e.g., [16]) does not take into account the increase in Fisher information due to the constraints. This increase is due to the reduction in uncertainty about the parameter values. Fortunately, we can move the inequality constraints shown in eq. (2.4) into the likelihood function using the following change in variables:

$$\theta_i = \beta(\gamma_i) = \frac{1}{1 + \exp(-\gamma_i)},$$

with $\gamma_i \in \mathbb{R}$. Function β maps γ_i with domain \mathbb{R} to $(0, 1)$, thus automatically satisfying our inequality constraints.

In what follows we take the equality constraint in eq. (2.5) into account in the Cramér-Rao inequality (Gorman and Hero [31]). Let n be the number of samples. Let $J(D)$ denote the unconstrained Fisher information in respect to γ , given by

$$J(D) \triangleq \sum_{\forall j} (\nabla_{\gamma} \ln P[D = \hat{d} | \theta]) (\nabla_{\gamma} \ln P[D = \hat{d} | \theta])^{\top} d_j.$$

Let $\mathbf{K}(D)$ denote the reduction in the Cramér-Rao bound due to the equality constraint, which is given by

$$\mathbf{K}(D) = J^{-1}(D) \mathbf{G}^{\top} (\mathbf{G} J^{-1}(D) \mathbf{G}^{\top})^{-1} \mathbf{G} J^{-1}(D),$$

where

$$\mathbf{G} = \nabla_{\gamma} g(\boldsymbol{\gamma}),$$

with $g(\boldsymbol{\gamma}) = \sum_{\forall i} \beta(\gamma_i) - 1$. Note that the reduction in the Cramér-Rao bound (represented by $K(D)$) can be seen as an increase in the Fisher information caused by the (equality) constraint imposed over the parameters. Note that $g(\boldsymbol{\gamma}) = 0$ replaces the equality constraint in equation (2.5). The constrained Cramér-Rao bound is then

$$E[(\boldsymbol{\gamma} - \hat{\boldsymbol{\gamma}})(\boldsymbol{\gamma} - \hat{\boldsymbol{\gamma}})^{\top}] \geq -(\mathcal{I}^+(D))_{ii}/n, \quad (2.6)$$

where

$$\mathcal{I}^+(D)_{ii} = J^{-1}(D) - \mathbf{K}(D). \quad (2.7)$$

The pseudo-inverse of Fisher information can be used in the Cramér-Rao inequality to obtain a lower bound on the mean squared error of any unbiased estimator (in respect to θ).

The Cramér-Rao bound obtained in eq. 2.6 is a constraint on $\boldsymbol{\gamma}$ not on θ . Fortunately, a similar bound for θ can be obtained [62]. Let $\mathbf{H} = [h_{i,j}]$ with $h_{i,j} = \partial\beta(\gamma_j)/\partial\gamma_i$. Then,

$$E[(\theta - \hat{\theta})(\theta - \hat{\theta})^\top] \geq \mathbf{H}(-\mathcal{I}(\boldsymbol{\gamma})^+/n) \mathbf{H}^\top. \quad (2.8)$$

2.4 Data processing inequality

The data processing inequality states that for any function f (with domain \mathbb{R}^k), the following inequality holds (see Zamir [69])

$$\mathcal{I}(D) \geq \mathcal{I}(f(D)). \quad (2.9)$$

The above equation has a powerful interpretation. It states that **processing** the observable data can never increase the amount of information already contained in the data. This property is very useful to us. In analyzing even the most convoluted measurement method we can simply look at the amount of information it captures, i.e., no algorithm can extract more information than what is given by $\mathcal{I}(D)$. The inequality

$$\mathcal{I}(\{D, Y\}) \geq \mathcal{I}(D).$$

also holds true. This means that adding more information, in the form of Y , never decreases the amount of Fisher information.

2.5 Random walk sampling

2.5.1 Notation

Here I present a formal definition of the sampling problem. Let $G_d = (V, E_d)$ be a labeled directed graph representing the (original) network graph. We assume that each vertex in G_d has at least one incoming or outgoing edge. An edge in G_d is an

ordered pair of nodes (u, v) representing a connection from u to v . The in-degree of a vertex u in G_d is the number of distinct edges $(v_1, u), \dots, (v_k, u)$ into u , and its out-degree is the number of distinct edges $(u, v_1), \dots, (u, v_k)$ out of u . If a random walker has the ability to retrieve incoming and outgoing edges from a queried vertex (and vertices are distinguishable), then we can represent G_d as a undirected graph. Let $G = (V, E)$ be the undirected counterpart of G_d , i.e., $E = \{(u, v) | (u, v) \in E_d \vee (v, u) \in E_d\}$. Note that G is not necessarily connected. Let $\deg(v)$ denote the degree of vertex v in G . Let \mathcal{L} be a finite set of vertex labels (it is trivial to extend our results to include edge labels). Each vertex in $v \in V$ is associated to a set of labels $\mathcal{L}(v) \subseteq \mathcal{L}$. For instance, a vertex label in G can be its in-degree in the original graph G_d .

2.5.2 Single random walker

A random walk visits a graph G by moving a particle (walker) from a vertex to a neighboring vertex through an outgoing edge chosen uniformly at random. Describing a random walk is simple. Let $\deg(v)$ denote the degree of vertex v in G (recall that G is undirected). A random walk that visits B vertices and a starting vertex $v_0 \in V$ can be described as:

- (1) Set $n \leftarrow 0$.
- (2) Set $v \leftarrow v_0$.
- (3) Choose an edge (v, u) from the edges of v with probability $1/\deg(v)$.
- (4) Set $v \leftarrow u$ and $n \leftarrow n + 1$.
- (5) While $n < B$ goto (3).

In what follows we look at the spectral decomposition of the Markov chain associated to a random walker over G .

2.5.3 Spectral decomposition of a random walk.

We start with known facts about random walks [45]. Assume G is not bipartite. Let $A = [a_{ij}; \forall i, j]$, where $a_{ij} = 1$, if $(v_i, v_j) \in E$ and $a_{ij} = 0$ otherwise, be the adjacency matrix of G and let

$$D = \begin{bmatrix} \deg(v_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \deg(v_{|V|}) \end{bmatrix}$$

be a diagonal matrix whose diagonal elements are the degrees of the vertices in G . Let $\mathbf{P} = D^{-1}A$ be the one-step transition probability matrix of the random walk. The probability of visiting a given vertex or edge can be obtained as a function of the starting state using the spectral decomposition of \mathbf{P} .

Let $\langle a, b \rangle = \sum_{\forall i} a_i b_i$ denote the inner product of vectors a and b . A transition probability matrix \mathbf{P} of a random walk can be decomposed into its left and right eigenvectors and eigenvalues [36]

$$\varphi_k \mathbf{P} = \lambda_k \varphi_k \quad \text{and} \quad \mathbf{P} \psi_k = \lambda_k \psi_k, \quad k = 1, \dots, |V|,$$

where $\forall k$, $\langle \varphi_k, \varphi_k \rangle = \langle \psi_k, \psi_k \rangle = 1$, and the indexes k are ordered such that $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{|V|}$. As G is connected, symmetric, and non-bipartite (and because \mathbf{P} is a stochastic matrix), it follows from the Frobenius-Perron Theorem that $1 = \lambda_1 > \lambda_2 \geq \dots \geq \lambda_{|V|} > -1$ [45]. The probability that a random walk reaches vertex v in n steps, given that it starts from vertex u , is [36, 45]

$$p_{uv}^{(n)} = \frac{\deg(v)}{2|E|} + \sqrt{\frac{\deg(v)}{\deg(u)}} \sum_{k=2}^{|V|} \lambda_k^n \psi_k(u) \varphi_k(v). \quad (2.10)$$

As $|\lambda_k| < 1$ for all $k \neq 1$, it is straightforward to see that

$$\lim_{n \rightarrow \infty} p_{uv}^{(n)} = \deg(v)/2|E|,$$

which is the same result as solving $\pi \mathbf{P} = \pi$. It is also worth noting that [45]

$$\sum_{\forall u \in V} \pi_u p_{uv}^{(n)} = \deg(v)/(2|E|), \quad n = 0, 1, \dots$$

and if $\pi_v^{(0)}$ denotes the probability that the random walk starts at vertex v , the probability of visiting edge (t, s) is

$$\sum_{\forall u \in V} \pi_u^{(0)} \left(\frac{p_{ut}^{(n)}}{\deg(t)} + \frac{p_{us}^{(n)}}{\deg(s)} \right).$$

2.5.4 Stationary random walks.

Section 2.5.3 shows that the random walker has a unique stationary (aka stable) distribution $\pi \mathbf{P} = \pi$ such that $\sum_{\forall v \in V} \pi_v = 1$. At any given step, a stationary random walker visits vertex v with probability $\deg(v)/|E|$. The probability that an **edge is visited is** $1/|E|$ (i.e., edges are visited uniformly at random). This means that if the initial vertex v_0 is chosen from V with probability $\deg(v_0)/|E|$ then the sequence of visited vertices and edges (V' and E' , respectively) form a stationary sequence [45]. A sequence X_1, X_2, \dots of random variables is said to be stationary if for any positive integers i and k , the joint distribution of $(X_i, X_{i+1}, \dots, X_{i+k})$ is independent of i . Note that a random walk over G starting at some (arbitrary) initial vertex v_0 is asymptotically stationary [45]. It can also be shown that starting at some (arbitrary) initial vertex v_0 , and walking sufficiently many (say n) steps over the graph, then the $n+1$ -st step is almost stationary (according to an appropriate metric), provided that n is large enough [45].

CHAPTER 3

ESTIMATING GRAPH CHARACTERISTICS WITH MULTIDIMENSIONAL RANDOM WALKS

3.1 Introduction

Social and computer networks permeate our modern lives. Inspired largely by the empirical study of data from real-world computer and social networks, the flourishing field of network science seeks to advance the knowledge about the structure and behavior of such systems. A number of recent studies [8, 20, 29, 42, 43, 50, 56, 52, 54, 64] (to cite a few) are dedicated to the characterization of complex networks. A complex network is a network with non-trivial topological features (features that do not occur in simple networks such as lattices or random networks). Examples of such networks include the Internet, the World Wide Web, social, business, and biological networks [8, 52]. This chapter represents a complex network as a directed graph with labeled vertices and edges. A label can be, for instance, the degree of a vertex or, in a social network setting, someone's hometown. Examples of network characteristics include the degree distribution, the fraction of HIV positive individuals in a population [48], or the average number of copies of a file in a peer-to-peer (P2P) network [30].

Characterizing the labels of a graph requires querying vertices and/or edges; each query has an associated cost in resources (time, bandwidth, money). Characterizing a large graph by querying the whole graph is often too costly. As a result, researchers have turned their attention to the estimation of graph characteristics based on incomplete (sampled) data. This chapter presents a new tool to characterize complex

networks. In what follows *random vertex (edge) sampling* refers to sampling vertices (edges) independently and uniformly at random (with replacement).

Distinct sampling strategies have different resource requirements depending on the network being sampled. For instance, in a network where each vertex is assigned a unique user-id (e.g., travelers and their passport numbers) it is a widespread practice to perform random vertex sampling by querying randomly generated user-ids. This approach can be resource-intensive if the user-id space is sparsely populated (e.g., less than 10% of all MySpace user-ids between the highest and lowest valid user-ids are currently occupied [56]). Another way to sample a network is by querying edges instead of vertices. Randomly sampling edges can be harder than randomly sampling vertices if edges are not be associated to unique IDs (or the IDs cannot be randomly queried). We summarize some drawbacks of random vertex and edge sampling:

- Random edge sampling can be impractical when edges cannot be directly queried (e.g., Facebook [29] and MySpace [56]).
- Random vertex sampling may be undesirable when user-ids are sparsely populated and queries are subject to resource constraints (e.g., queries are rate-limited in Flickr, Livejournal [50], and Bittorrent [38]). In a P2P network like Bittorrent, a client can randomly sample peers (vertices) by querying a tracker (server); however, trackers may rate-limit client queries [38].
- Even when random vertex sampling is not severely resource-constrained, some characteristics may be better estimated with random edge sampling (e.g., the degree distribution tail of a (finite) power-law graph).

An alternative, and often cheaper, way to sample a network is with a random walk (RW). A RW samples a graph by moving a particle (walker) from a vertex to a neighboring vertex (through an edge). By this process edges and vertices are sampled. The probability by which the random walker selects the next neighboring vertex

determines the probability by which vertices and edges are sampled. This chapter is interested in random walks that sample *edges* uniformly. These samples can be used to obtain unbiased estimates of a variety of graph characteristics (I present two examples in Section 3.3).

In order to estimate network characteristics, this chapter assumes that a random walker has the ability to query incoming and outgoing edges of a vertex (Section 3.3 provides the reason behind this assumption). This is possible over graphs such as Twitter, LiveJournal [50], YouTube [50], Facebook [29], MySpace [56], P2P networks [54], and the arXiv citations network. This chapter revisits the theory behind random walks in Section 3.3.

Sampling graphs with random walks is not without drawbacks. The accuracy of the estimates depends not only on the graph structure but also on the characteristic being estimated. The graph structure can create distortions in the estimates by “trapping” the random walker inside a subgraph. An extreme case happens when the graph consists of two or more disconnected components. For instance, wireless mobile social networks exhibit connection graphs with multiple disconnected subgraphs [20]. But even connected graphs can suffer from the same problem. A random walker can get “temporarily trapped” and spend most of its sampling budget exploring the local neighborhood near where it got “trapped”. In such scenario, estimates may be inaccurate if the characteristics of the local neighborhood differ from the overall characteristic of the graph. This problem is well documented (see [44]) and our goal is to mitigate it.

3.1.1 Contributions

This chapter proposes a new multidimensional random walk sampling method (*Frontier sampling*) that preserves all of the important statistical properties of a regular random walk, while mitigating the large estimation errors caused by discon-

nected or loosely connected subgraphs that can “trap” a random walker and distort the estimated graph characteristic. In Section 3.5 we see that estimates from Frontier sampling have smaller Mean Squared Errors (MSEs) than estimates obtained from regular random walkers (single and multiple independent walkers (reviewed in Section 3.3.4)) in a variety of scenarios.

This chapter makes two additional contributions: (1) we compare random walk-based estimates to random vertex and random edge sampling. I show analytically that the tail of power law graphs is better estimated using random walks (or random edge sampling) than using random vertex sampling. These results help explain recent empirical results [54]; (2) another contribution of this chapter comes in the form of estimators of graph characteristics. While the literature focuses on vertex-centric estimators for random walks (estimators that use sampled vertices), e.g., Respondent-Driven Sampling (RDS) [64], casting these estimators as edge-centric simplifies the design of edge-centric characteristic estimators such as the global clustering coefficient (described in Section 3.3.2.2).

3.1.2 Outline

The notation used in this chapter is found in Chapter 2.5. The outline of this chapter is as follows. Section 3.2 contrasts random vertex with random edge sampling. Section 3.3 revisits single and multiple independent random walk sampling and estimation. Section 3.4 introduces *Frontier Sampling* (FS), a sampling process that uses m dependent random walkers in order to mitigate the high estimation errors caused by disconnected or loosely connected subgraphs. Section 3.4 also shows that FS can be seen as an m -dimensional random walk over the m -th Cartesian power of the graph (formally defined in Section 3.4). In Section 3.5 we see that FS outperforms both single and multiple independent random walkers in a variety of scenarios. This chapter also compares independent sampling of vertices and edges with FS sam-

pling. Section 3.7 reviews the relevant literature. Finally, Section 3.8 presents the conclusions and future work.

3.2 Vertex v.s. edge sampling

Here we consider a simple estimation problem. I use this to illustrate the tradeoff between random edge and random vertex sampling. Consider the problem of estimating the out-degree distribution of G_d . Let θ_i be the fraction of vertices with out-degree $i > 0$ and $E[D]$ be the average out-degree. The error metric used in most examples is the normalized root mean square error of $\hat{\theta}_l$, which is a normalized measure of the dispersion of the estimates, defined as

$$\text{NMSE}(l) = \frac{\sqrt{E[(\hat{\theta}_l - \theta_l)^2]}}{\theta_l}. \quad (3.1)$$

We assume that $E[D]$ is known and that a sampled edge (u, v) only provides the out-degree of u . It is easy to see that the probability that random edge sampling samples a vertex with out-degree i is $\pi_i = i\theta_i/E[D]$. Random vertex sampling samples a vertex with out-degree i with probability θ_i . A simple calculation shows that the NMSE (equation (3.1)) of B randomly sampled *edges* with out-degree i is

$$\text{NMSE}(i) = \sqrt{(1/\pi_i - 1)/B}, \quad i > 0. \quad (3.2)$$

Similarly, the NMSE of randomly sampled *vertices* with out-degree i is

$$\text{NMSE}(i) = \sqrt{(1/\theta_i - 1)/B}. \quad (3.3)$$

Now note that $\pi_i/\theta_i = i/E[D]$, which means that $\pi_i > \theta_i$ if $i > E[D]$ and $\pi_i < \theta_i$ if $i < E[D]$. From equations (3.2) and (3.3) we see that random edge sampling more

accurately estimates large out-degrees ($i > E[D]$) while random vertex sampling more accurately estimates small out-degrees ($i < E[D]$) for the same fixed number of samples. This means that random edge sampling exhibits smaller NMSE when estimating the tail of the out-degree distribution. This characteristic of random edge sampling is also known as importance sampling estimation [58].

The example above is just one of many instances where random edge sampling is preferred over random vertex sampling. Another example: one can argue that random edge sampling simplifies the estimation of edge-centric graph characteristics such as the global clustering coefficient. Unfortunately, as discussed in Section 3.1, random edge sampling is rarely practical. In what follows we see that, if G is connected, random walks exhibit similar statistical properties to random edge sampling, without the (costly) need of independence.

3.3 Random walk sampling

In this section we review random walk (RW) sampling and estimation over G . In what follows we assume that G is connected and non-bipartite. Sampling G with a RW is a simple task. A random walker with budget B starts at vertex $v_0 \in V$. For the sake of simplicity, in the remainder of this thesis we assume that all queries of edges and vertices have unitary cost and that we have a fixed sampling budget B (generalizing the unitary cost assumption is quite straightforward). Let $V' = \{v_i\}_{i=1}^B$ be a sequence of sampled vertices and $E' = \{(u_i, v_i)\}_{i=1}^B$ be the corresponding sequence of sampled edges in a RW. We define V' and E' as sequences because the same vertices (edges) may be sampled multiple times. We refer to $v_i \in V'$ and $(u_i, v_i) \in E'$ as the i -th sampled vertex and edge, respectively. At the n -th step, the random walker at vertex v chooses an outgoing edge (v, u) uniformly at random (as seen in Section 2.5.4). The walker adds v to V' and (v, u) to E' . At step $n + 1$ the random walker starts at vertex u and the sampling continues until step B . The RW described here is the

most common type of RW found in the literature [45]. Other types of random walks differ in the way outgoing edges are sampled (e.g., random walks that mimic random vertex sampling); please refer to [58] for more details.

An important property of a RW is its ability to reach a unique stationary regime. A necessary condition for stationarity is that G must be connected and non-bipartite. In a stationary RW, E' is a stationary sequence. A sequence X_1, X_2, \dots of random variables is said to be stationary if for any positive integers n and k , the joint distribution of (X_n, \dots, X_{n+k}) is independent of n . Stationarity is a natural generalization of random sampling where the assumption of independence is dropped. Once it reaches steady state, the above RW shares two important properties with random edge sampling. Both sample edges uniformly at random (as shown in Section 2.5.4) and both obey the strong law of large numbers, as we see next.

3.3.1 Strong Law of Large Numbers

The strong law of large numbers is a powerful tool that states that the sample average of any function over the samples converges almost surely to its expected value. This property is very useful in building accurate estimators. In this section we see that the average of any function f over the sampled edges (vertices) of a stationary RW converges almost surely to its expected value, under certain constraints [49]. Here I provide details of this known property, which is included here for the sake of completeness. Let X_n be the n -th edge sampled by a RW over G (a similar result can be obtained for the n -th sampled vertex) and B be the size of E' (the number of RW steps).

Theorem 3.3.1 (SLLN). *A RW over G satisfies the strong law of large numbers, namely that for any function f , where $\sum_{(u,v) \in E} |f(u,v)| < \infty$,*

$$\lim_{B \rightarrow \infty} \frac{1}{B} \sum_{n=1}^B f(X_n) \xrightarrow{a.s.} \frac{1}{|E|} \sum_{\forall (u,v) \in E} f(u,v),$$

where “a.s.” denotes “almost sure” converge, i.e., the event happens with probability one.

Proof. The Markov chain associated to a random walker over G is ergodic, as G is undirected and non-bipartite [45]. Thus, we can directly apply the Strong Law of Large Numbers for ergodic Markov chains [58, Theorem 6.63]. \square

Theorem 3.3.1 allows us to construct estimators of graph characteristics that converge to their true values as the size of E' goes to infinity ($B \rightarrow \infty$). In what follows we apply Theorem 3.3.1 to estimate graph characteristics; we also present two examples.

3.3.2 Estimators

An *estimator* is a function that takes the observations (sampled data) as input and outputs an estimate of a unknown population parameter (graph characteristic). In this section we see how to estimate graph characteristics using E' (the sampled edges of a RW). Estimators that take V' as input are commonly used to estimate vertex-oriented metrics (such as the degree distribution) and can be found in the literature [58, 64].

Here I present estimators of two graph characteristics: the vertex (edge) label density (the fraction of vertices (edges) with a given label in the graph) and the global clustering coefficient. The design of the estimator is simple: (1) First we find a function f that computes the characteristic of G assuming $V' = V$ and $E' = E$; (2) later we replace the assumption that $V' = V$ and $E' = E$ with the assumption that V' and E' are sequences drawn from a stationary RW.

3.3.2.1 Label Density

This section illustrates how to build an estimator using a simple example. Recall that we can record the in- and out-degrees of G_d as vertex labels in G . Each vertex

in $v \in V$ is associated with a label $\mathcal{L}(v) \subseteq \mathcal{L}$, where \mathcal{L} is the set of labels defined in Section 2.5.1. A label can be, for instance, the in-degree of v in the original graph G_d . We seek to calculate, θ_l , the fraction of vertices with label l in G . The following estimator is a simple edge-based version of the vertex-based RDS estimator [64]. Because (for now) we assume that V and E are known, we have

$$\theta_l \equiv \sum_{\forall(u,v) \in E} f(u,v) \equiv \sum_{\forall(u,v) \in E} (h_l(v) + h_l(u)), \quad (3.4)$$

where

$$h_l(v) = \begin{cases} \frac{1}{\deg(v)|V|} & \text{if } l \in \mathcal{L}(v) \\ 0 & \text{otherwise.} \end{cases}$$

It is trivial to verify that θ_l is the fraction of vertices with label l . Now we replace the assumption that $V' = V$ and $E' = E$ with the assumption that $E' = \{(u_i, v_i)\}_{i=1}^B$ is a sequence of B edges sampled by a stationary RW. To eliminate the dependence of h_l on any unknown values (e.g. $|V|$ and $|E|$) we need to redefine h_l :

$$h'_l(v) = \begin{cases} 1/\deg(v) & \text{if } l \in \mathcal{L}(v) \\ 0 & \text{otherwise.} \end{cases}$$

Following equation (3.10) in [58, pg. 95] (substituting “ $f(x_j)$ ” for “ $1/|V|$ ” and “ $g(x_j)$ ” for “ $\deg(x_j)/|E|$ ”), we have that

$$\hat{\theta}_l \equiv \frac{1}{SB} \sum_{i=1}^B h'_l(v_i) + h'_l(u_i), \quad (3.5)$$

where $S = \sum_{i=1}^B 1/\deg(v_i) + 1/\deg(u_i)$, is asymptotically unbiased.

3.3.2.2 Global Clustering Coefficient

In the literature the term *clustering coefficient* often refers to the local clustering coefficient [66]. In the following example we estimate a different metric: the *global*

clustering coefficient. In a social network the global clustering coefficient, C , is the probability that the friend of John's friend is also John's friend [52]. More formally, the global clustering coefficient can be defined as [52]

$$C = \frac{6 \times \text{number of triangles in the graph}}{\text{number of directed paths of length two}},$$

where a triangle is a clique with 3 vertices and a directed path of length two refers to any directed path that connects two vertices in the graph.

$$\Delta(E) \equiv \sum_{i=1}^B f_{\Delta}(u_i, v_i)/3, \quad (3.6)$$

where $f_{\Delta}(u, v)$ is a function that returns the number of common neighbors between u and v . We can also calculate the number of directed paths of length two

$$l(E) \equiv \sum_{i=1}^B f_l(u_i, v_i) \equiv \sum_{i=1}^B ((\deg(u_i) - 1) + (\deg(v_i) - 1)), \quad (3.7)$$

as an edge (u, v) belongs to $2(\deg(u) + \deg(v) - 2)$ directed paths of length two and each path is counted twice in the summation. Note that C is well defined only if $l(E) > 0$.

As with the previous estimator example, we replace the assumption that $E' = E$ with the assumption that E' is sampled by a stationary RW. Applying Theorem 3.3.1 we have that $\lim_{B \rightarrow \infty} l(E')/B \xrightarrow{\text{a.S.}} l(E)|E|$ and that $\lim_{B \rightarrow \infty} \Delta(E')/B \xrightarrow{\text{a.S.}} \Delta(E)|E|$.

From the above we obtain the following lemma.

Lemma 3.3.2. *Let $l(E) > 0$ and*

$$\hat{C} = \frac{6 \Delta(E')}{l(E')}.$$

Then \hat{C} is an asymptotically unbiased estimator of C , i.e., $E[\lim_{B \rightarrow \infty} \hat{C}] = C$.

Proof. The proof is quite easy. As $l(E) > 0$, we have that C is well defined. Let

$$l^* = \lim_{B \rightarrow \infty} l(E')(|E|/B)$$

and

$$\Delta^* = \lim_{B \rightarrow \infty} \Delta(E')(|E|/B).$$

Thus,

$$\lim_{B \rightarrow \infty} \hat{C} = \frac{\Delta^*}{l^*}.$$

From Theorem 3.3.1 we know that

$$l^* \xrightarrow{\text{a.s.}} l(E)$$

and

$$\Delta^* \xrightarrow{\text{a.s.}} \Delta(E).$$

Let $\Gamma_l = l^* - l(E)$ and $\Gamma_\Delta = \Delta^* - \Delta(E)$. Almost sure convergence of l^* and Δ^* means that

$$P[\Gamma_l = 0] = 1 \quad \text{and} \quad P[\Gamma_\Delta = 0] = 1,$$

respectively, which also implies that

$$P[\Gamma_\Delta = 0 \cap \Gamma_l = 0] = 1$$

as

$$\begin{aligned} P[\Gamma_\Delta = 0] &= \sum_{\forall \gamma} P[\Gamma_\Delta = 0 \mid \Gamma_l = \gamma] P[\Gamma_l = \gamma] \\ &= P[\Gamma_\Delta = 0 \mid \Gamma_l = 0] P[\Gamma_l = 0]. \end{aligned}$$

Then,

$$\begin{aligned} E \left[\lim_{B \rightarrow \infty} \hat{C} \right] &= \sum_{\forall \gamma_l} \sum_{\forall \gamma_\Delta} \frac{\Delta(E) + \gamma_\Delta}{l(E) + \gamma_l} P[\Gamma_l = \gamma_l \cap \Gamma_\Delta = \gamma_\Delta] \\ &= \frac{\Delta(E)}{l(E)} = C, \end{aligned}$$

which concludes the proof. \square

3.3.3 Estimator Accuracy & Graph Structure

Sampling a graph using a RW is not without drawbacks. A random walker can get (temporarily) “trapped” inside a subgraph whose characteristics differ from those of the whole graph. If the random walker starts in steady state (i.e., is stationary), this scenario may increase the mean squared error of the estimates. If the random walker does not start in steady state, this scenario may cause an increase in the estimation bias as well as the mean squared error. Ideally, the random walker needs to mitigate the effect of these traps over the estimates.

The above two types of estimation error are well documented in the literature and various solutions are available [28, 58]. For instance, if the random walker does not start in a stationary regime (transient), it is common practice to discard the first w samples [28]. The value of w is called the *burn-in period*. There are two problems with this solution: (1) it only reduces the error related to the non-stationarity of the samples; (2) it is difficult to determine a good value for w when the size and structure of G are unknown.

A simple naive solution to the RW “trapping” problem (adopted in [29] to sample Facebook), is to sample the graph using multiple independent random walkers [28]. In what follows we see that such a naive approach can lead to increased estimation errors. In Section 3.4 we see how to mitigate the random walk “trapping” problem with K *dependent* random walkers.

3.3.4 Multiple Independent Random Walkers

Here we sample G using K (parallel) independent random walkers (*MultipleRW*). In order to distinguish *MultipleRW* and sampling using a single random walker, we denote the former *SingleRW*. To simplify our exposition we assume that if B is the sampling budget, each walker takes B/K steps. Let (V_1, \dots, V_K) be the state of K independent random walkers in steady state. It is easy to verify that, as in the *SingleRW* case, edges are sampled with probability $1/|E|$.

A drawback of *MultipleRW* can be explained using a simple example. Consider two random walkers ($K = 2$) walking over a graph that has two disconnected large components (subgraphs) G_A and G_B . Let $\text{vol}(G_A)$ and $\text{vol}(G_B)$ denote the total number of edges in G_A and G_B , respectively. One random walker starts in G_A and the other one in G_B . The random walker stuck in G_A (G_B) samples edges of G_A (G_B) with probability that converges to $1/\text{vol}(G_A)$ ($1/\text{vol}(G_B)$). If $1/\text{vol}(G_A) > 1/|E|$ then edges of G_A are oversampled and, consequently, the edges of G_B are undersampled (as $1/\text{vol}(G_B) < 1/|E|$). On the other hand, when $1/\text{vol}(G_A) < 1/|E|$ the edges of G_A are undersampled and the edges of G_B are oversampled. Thus, for disconnected graphs the starting vertices of the random walkers are a key factor to determine the accuracy of *MultipleRW*. Increasing the sampling budget minimizes the problem only if G is connected.

Moreover, the reduction by a factor of K in the budget of each random walker can exacerbate their non-stationarity. This issue is well documented in the literature and there seems to be no consensus whether *MultipleRW* estimates are more accurate than *SingleRW* ones (refer to [28] for a discussion). *MultipleRW* can also be used to detect the convergence of the estimates to their true value by, say, comparing the estimates obtained by each RW with the estimates combining all K RW together (e.g. Gelman-Rubin convergence diagnostics [27]). Here too there is no consensus. Some

authors argue that convergence is better diagnosed by dividing a longer SingleRW into K non-overlapping segments [28].

We say that a graph is *homogeneously explored* by a set of random walkers when the edge sampling probabilities of each sampled edge are similar. In Section 3.5 we see practical examples of non-homogeneous exploration by MultipleRW; we also see that this implies large estimation errors. Thus, Sections 3.3.3 and 3.3.4 leave us with the following question:

Q: *Is it possible to “homogeneously explore” a graph using multiple random walkers?*

3.4 Frontier sampling

In this section we present a new and promising approach to address the above question. *Frontier Sampling* (FS) performs m *dependent* random walks in the graph. We refer to m as the dimension of the FS random walk. Let c be the cost of randomly sampling m vertices. The FS algorithm is simple:

- (1) $n \leftarrow 0$ // n is the number of steps //
- (2) Initialize with a collection of m randomly chosen vertices $L = (v_1, \dots, v_m)$
- (3) Select $u \in L$ with probability $\deg(u) / \sum_{v \in L} \deg(v)$
- (4) Select an outgoing edge of u , (u, v) , uniformly at random
- (5) Replace u by v in L , add u to V' , and add (u, v) to E'
- (6) $n \leftarrow n + 1$
- (7) While $n < B - c$ goto line (3)

Frontier Sampling (FS) is a centrally coordinated sampling algorithm that maintains a list of m vertices representing m random walkers. This way FS is less likely to get stuck in loosely connected subgraphs than a single random walker. However, unlike m independent random walkers, all m Frontier samplers (random walkers) share the same sampling process and budget. In all of our simulations, presented in Section 3.5, FS estimates are more accurate than both single and m independent random walkers. Section 3.8 describes how the FS algorithm can be made fully distributed.

Frontier Sampling: An m -dimensional Random Walk

Now we see that FS shares many of the same statistical properties of a single random walker. The key insight behind Theorem 3.4.2 below is that the FS stochastic process is equivalent to the stochastic process of a single random walker over the m -th Cartesian power of G , $G^m = (V^m, E_m)$, where

$$V^m = \{(v_1, \dots, v_m) \mid v_1 \in V \wedge \dots \wedge v_m \in V\}$$

is the m -th Cartesian power of V and $\forall \mathbf{v}, \mathbf{u} \in V^m$, $(\mathbf{v}, \mathbf{u}) \in E_m$ if there exists an index i such that $(v_i, u_i) \in E$ and $u_j = v_j$ for $j \neq i$.

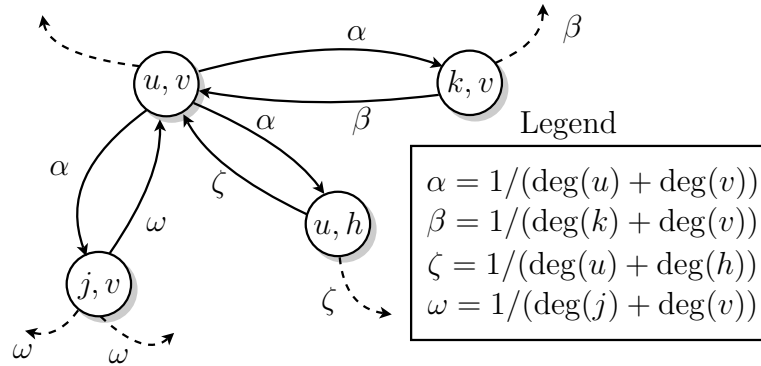


Figure 3.1. Illustration of the Markov chain associated to the Frontier sampler with dimension $m = 2$.

Lemma 3.4.1. *The Frontier sampling process is equivalent to the sampling process of a single random walker over G^m .*

Proof. Consider the $(n - 1)$ -st step of FS. The reader may find Figure 3.1 helpful in following the proof. Let $L_n = (v_1, \dots, v_m)$ be the state of FS before the n -th step. Clearly $L_n \in V^m$. Let $e(L_n)$ denote the collection of all edges associated to the vertices in L_n . We refer to $e(L_n)$ as the edge frontier at the n -th step. We describe the transition from state L_n to state L_{n+1} as follows (lines (3) and (4) of the FS algorithm): Select a vertex $v \in L_n$ with probability proportional to $\deg(v)$ and then

replace vertex v in L_n with one of its neighbors (selected uniformly at random). This is equivalent to randomly sampling an edge from $e(L_n)$ with probability

$$p = \frac{1}{|e(L_n)|} = \frac{1}{\sum_{\forall v \in L_n} \deg(v)}.$$

Therefore, L_n is able to transition to state L_{n+1} iff $(L_n, L_{n+1}) \in E_m$ and the transition probability from L_n to L_{n+1} is $1/|e(L_n)|$. Thus, the Markov chain that describes FS is equivalent to the Markov chain of a single random walker over G^m . \square

Theorem 3.4.2. *If G is connected and non-bipartite, then FS is asymptotically stationary and has a unique stable distribution where: (1) edges are sampled with probability $1/|E|$, (2) sampled edges form a stationary sequence, and (3) the sequence satisfies the Strong Law of Large Numbers (Theorem 3.3.1).*

Proof. Consider the $(n - 1)$ -st step of Frontier sampling. The reader may find Figure 3.1 helpful in following the proof. Let $L_n = (v_1, \dots, v_m)$ be the state of Frontier sampling before the n -th step. Clearly $L_n \in V^m$. In what follows let $e(L_n)$ denote the collection of all edges associated to the vertices in L_n . We refer to $e(L_n)$ as the edge frontier at the n -th step. We describe the transition from state L_n to state L_{n+1} as follows (lines 3 and 4 of the frontier sampling algorithm): Select a vertex $v \in L_n$ with probability proportional to $\deg(v)$ and then replace element v in L_n with one of its neighbors (selected uniformly at random). This is equivalent to randomly sampling an edge from $e(L_n)$ with probability

$$p = \frac{1}{|e(L_n)|} = \frac{1}{\sum_{\forall v \in L_n} \deg(v)}.$$

Thus, L_n is able to transition to state L_{n+1} iff $(L_n, L_{n+1}) \in E_m$ and the transition probability from L_n to L_{n+1} is $1/|e(L_n)|$. Thus, we conclude that Frontier sampling is a single random walker over the m -th Cartesian power of G , $G^m = (V^m, E_m)$, where

$$V^m = \{(v_1, \dots, v_m) \mid v_1 \in V \wedge \dots \wedge v_m \in V\}$$

is the m -ary Cartesian product of V and $\forall \mathbf{v}, \mathbf{u} \in V^m, (\mathbf{v}, \mathbf{u}) \in E_m$ if exists an index i such that $(v_i, u_i) \in E$ and $u_j = v_j$ for $j \neq i$. Note that $|E_m| = m|V|^{m-1}|E|$.

Now we need to prove that the distribution of L_∞ is stable and unique. For this we only need to show that the random walk over G^m is ergodic. A random walk (Markov chain) is ergodic when it is aperiodic and recurrent non-null. Recall that the random walk over G is ergodic. The probability that Frontier sampling transitions from $L_n \in V^m$ to $L_{n+1} \in V^m$ such that L_n and L_{n+1} only differ in their i -th element is always greater than zero, otherwise there is an infinite increasing degree sequence in the vertices of G . But this is not possible as the random walk over G is recurrent non-null (an infinite increasing degree sequence would be a sink in the random walk over G). Thus, any finite sequence of transitions $\{L_{n+w}\}_{w=1}^\Delta$ that only updates its i -th element has probability greater than zero. Thus, as the sequence $\{L_{n+w}\}_{w=1}^\Delta$ is also a single random walk over G , it is aperiodic for any chosen $i = 1, \dots, m$. Thus, a random walker over G^m must also be aperiodic. We can use the same argument to show that the random walk over G^m is recurrent non-null. As random walk over G^m is ergodic, we have that L^\star is distributed according to the steady state distribution of a random walk over G^m

$$P[L_\infty = (v_1, \dots, v_m)] = \frac{\sum_{i=1}^m \deg(v_i)}{m|V|^{m-1}|E|},$$

where $L_\infty \equiv \lim_{n \rightarrow \infty} L_n$, which is unique and stable (similar to a single random walker as seen in Section 3.3).

The rest of the proof is straightforward. Each edge in G^m is actually an edge in G . As each edge in G is copied m times into G^m , we have that edges in G are also sampled uniformly at random in a random walk over G^m . As Frontier sampling is a random walk over G^m , its samples form a stationary sequence and follow the Strong

Graph	Flickr	LiveJournal	YouTube	Hep-th Citations	Internet RLT
Description	Social Net.	Social Net.	Social Net.	ArXiv pubs.	Internet tracert.
Type of graph	Directed	Directed	Directed	Directed	Directed
# of Vertices	1,715,255	5,204,176	1,138,499	27,770	192,244
Size of LCC	1,624,992	5,189,809	1,134,890	27,400	609,066
# of Edges	22,613,981	77,402,652	9,890,764	352,807	609,066
Average Degree	12.2	14.6	8.7	12.7	3.2
% of Original Graph	26.9%	95.4%	$\leq 79.1\%$	NA	NA

Table 3.1. Summary of the graph datasets used in the simulations. “Size of LCC” refers to the size of the largest connected component.

Law of Large Numbers seen in Theorem 3.3.1. The same is true for the sequence of sampled vertices. \square

3.5 Results

In this section we compare FS with SingleRW and MultipleRW. We also contrast FS with random vertex and edge sampling. The experiments consist of executing these sampling methods on a variety of real world graphs. The datasets used in the simulations are summarized in Table 3.1: “Flickr”, “Livejournal”, and “YouTube” are popular photosharing, blog (weblog), and video sharing websites, respectively. Users represent as vertices of a graph. In these websites a user can subscribe to other user updates; an edge (u, v) exists between users u and v if user u subscribes to user v . At “Livejournal” and “YouTube” it is possible to query the incoming and outgoing edges of a given user. Further details of these three datasets can be found in [50]. “Hep-th Citations” is a graph of citation references in the ArXiv high energy physics publications archive [68]. “Internet RLT” is a router-level Internet graph collected from traceroute measurements of 23 monitors distributed over the world [25]. Note that some of these graphs contain disconnected components (subgraphs).

In the following simulations the starting vertex of each random walker is chosen uniformly at random from the set of all vertices. The results show that FS estimates are consistently more accurate than their SingleRW and MultipleRW counterparts. Moreover, when restricted to the largest connected component, FS reaches steady state faster than SingleRW and MultipleRW in the simulations presented in Section 3.5.5.

3.5.1 In- and Out-degree Distribution Estimates

Here we treat the graphs in Table 3.1 as undirected graphs. In-degrees and out-degrees are represented as vertex labels. Consider the in-degree distribution. Let $\theta = \{\theta_i\}_{\forall i}$ denote the in-degree distribution, where θ_i is the fraction of vertices with in-degree i . In the simulations θ is estimated using equation (3.5). Each simulation consists of 10,000 runs (sample paths) used to compute the empirical NMSE (equation (3.1)), which is then used to compare the accuracy of the estimates obtained from FS (dimension $m \in \{10, 100, 1000\}$), SingleRW, and MultipleRW ($K \in \{10, 100, 1000\}$ walkers). For the sake of conciseness, the following presentation is restricted to a handful of representative results.

Consider first two representative results from the Flickr graph, whose in-degree CCDF (complementary cumulative distribution function) log-log plot is shown in Figure 3.2. The sampling budget is $B = 18,123 = |V|/100$, which amounts to sampling 1% of the vertices. In the first simulation, the sampling is restricted to the *Largest Connected Component* (LCC) (which contains 94% of the vertices). The objective is to test if FS can outperform SingleRW and MultipleRW even when there are no disconnected subgraphs. Figure 3.3 shows a log-log plot of the NMSE of FS ($m = 1000$), SingleRW, and MultipleRW ($K = 1000$). First, note that the shape of the NMSE for high in-degrees is a consequence of the fact that vertices with high degrees in G tend to have unique high in-degree labels and that, similar

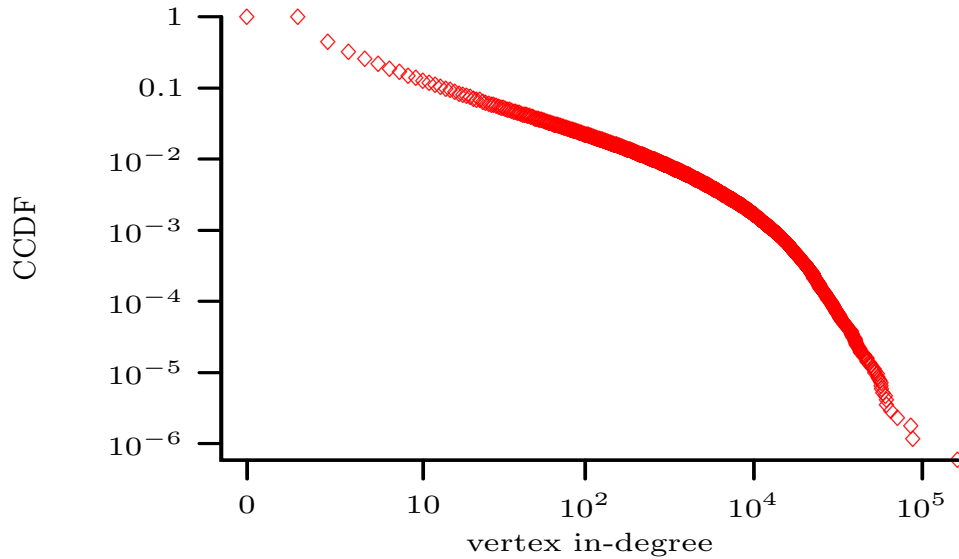


Figure 3.2. (Flickr) Log-log plot of the in-degree CCDF.

to random edge sampling, the NMSE decreases with the degree. Figure 3.3 shows that FS outperforms both SingleRW and MultipleRW (particularly at estimating small in-degrees). It is interesting to note that, for most degrees, estimates obtained by SingleRW are more accurate than the estimates obtained by MultipleRW. Now consider the complete Flickr graph. Figure 3.4 shows a log-log plot of the NMSE of the in-degree distribution. Contrasting the plots in Figures 3.3 and 3.4 note that the gap between FS and both SingleRW and MultipleRW has significantly increased, favoring FS.

To better understand the differences between these sampling methods, Figure 3.5 focuses on four runs (sample paths) of the simulation over the complete Flickr graph. Figure 3.5 plots the evolution of $\hat{\theta}_1$ (the estimate of θ_1) as a function of n (the number of steps in the random walk). At each run of the simulator both FS and MultipleRW start at the same vertices (initially chosen using random vertex sampling). Figure 3.5 shows that all four FS sample paths (runs) quickly converge to the value of θ_1 . For SingleRW, three out of the four runs start inside the LCC. These runs do not converge to the value of θ_1 as some vertices with in-degree one lie outside the LCC. In

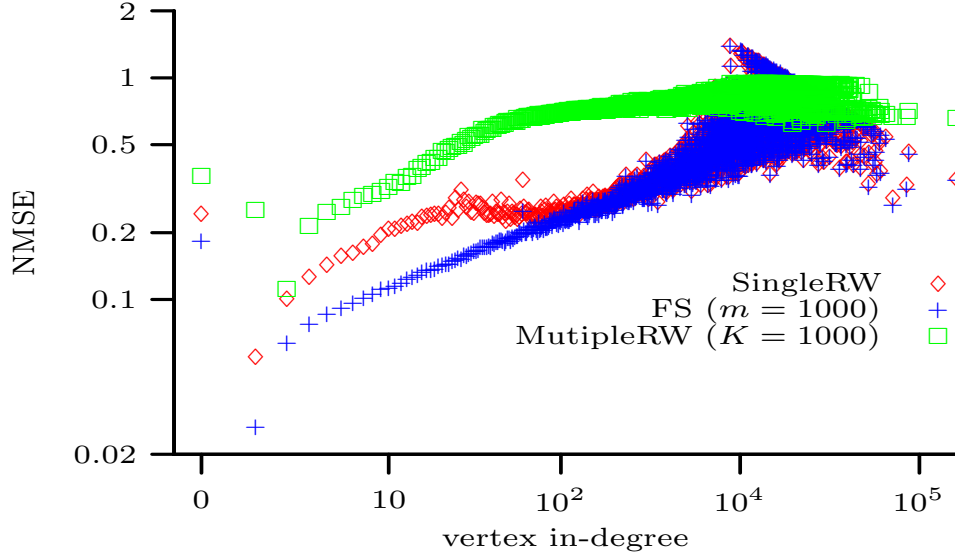


Figure 3.3. (LCC of Flickr) The log-log plot of the NMSE of the in-degree distribution estimates with budget $B = |V|/100$.

one of the runs, SingleRW starts in a small disconnected subgraph and, thus, grossly overestimates the value of θ_1 . For a similar reason, i.e., walkers starting at small disconnected subgraphs, MultipleRW grossly overestimates the value of θ_1 . The MultipleRW jump around $n = 10^3$ steps needs further investigation. It may be due to the transient of the random walk (discussed in Section 3.3.4). Even when $n \gg 1$ (not shown in Figure 3.5) the MultipleRW estimate is unable to converge to θ_1 . Modifying both SingleRW and MultipleRW methods to cope with disconnected components is an interesting open problem.

For the sake of conciseness, the simulation results for the remaining graphs (Table 3.1) are omitted as they are similar to the results observed over the Flickr graph. However, consider the *out-degree* distribution estimates of Livejournal. Figure 3.6 shows a log-log plot of the CCDF of the original out-degrees. The log-log plot of the NMSE is shown in Figure 3.7 for FS ($m = 100$), SingleRW, and MultipleRW ($K = 100$) with sampling budget $B = |V|/10$. From Figure 3.7 we see that esti-

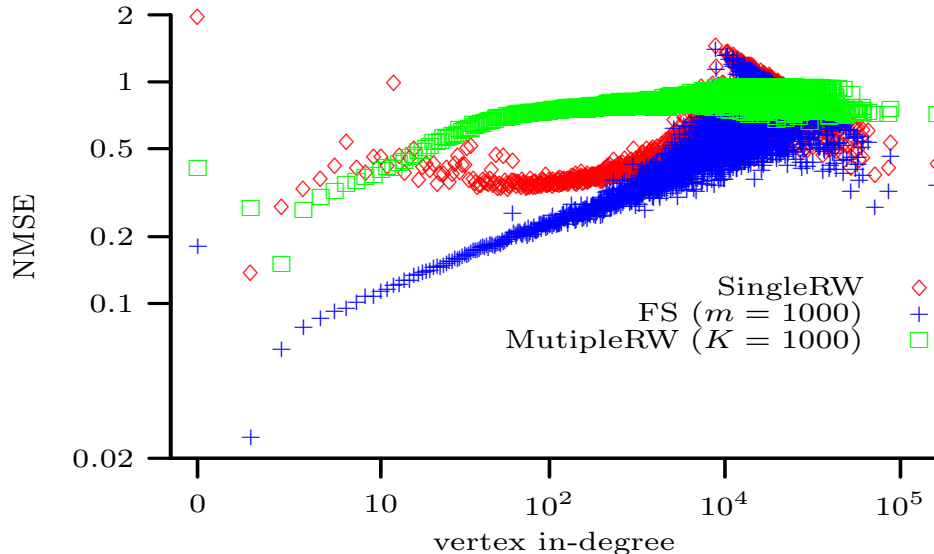


Figure 3.4. (Flickr) The log-log plot of the NMSE of the in-degree distribution estimates with budget $B = |V|/100$.

mates of vertices with small out-degrees in FS are up to one order of magnitude more accurate than those obtained from both SingleRW and MultipleRW.

The next experiment focuses on studying the impact of loosely connected subgraphs over the degree estimates. Consider a graph that consists of two instances of a random undirected Barabási-Albert [5] graph, G_A and G_B , with 5×10^5 vertices each and average degrees 2 and 10, respectively, joined by a single edge connecting the two smallest degree vertices in G_A and G_B (ties are resolved arbitrarily). Henceforth, this graph is referred to as G_{AB} .

The experiment consists of estimating the degree distribution of G_{AB} using FS ($m = 100$), SingleRW, and MultipleRW ($K = 100$). Again, both FS and MultipleRW start at the same vertices in each execution of the simulation, which are initially chosen uniformly at random. In this experiment the hypothesis is that, for small sampling budgets, each random walker will see the degree distribution of either G_A or G_B but not the degree distribution of G_{AB} . Moreover, as the starting vertex of each random walker is chosen uniformly at random, G_A , which has the same number

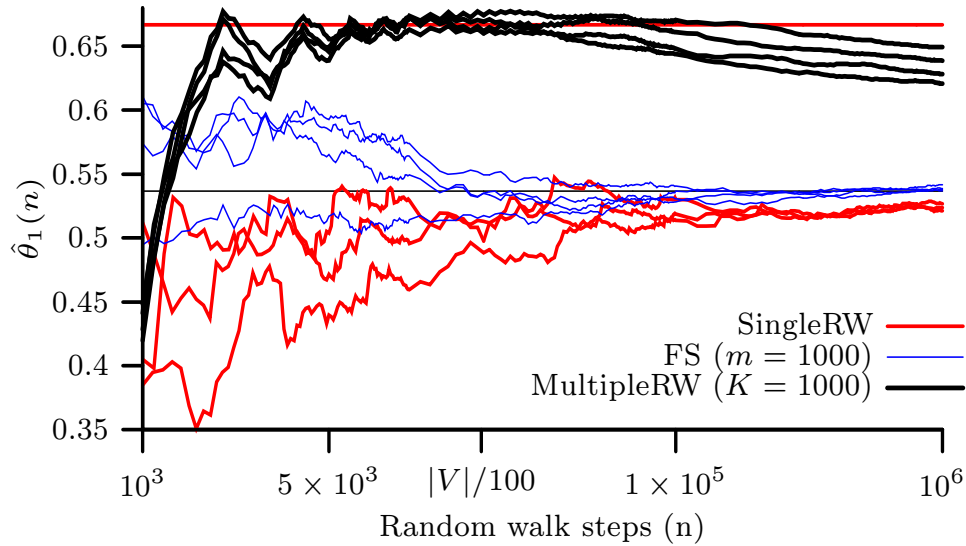


Figure 3.5. (LCC of Flickr) Four sample paths of $\hat{\theta}_1$ ($\theta_1 = 0.53$) as a function of the number of steps n (horizontal axis in log scale).

of vertices as G_B but $1/5$ of the edges, receives more random walkers than its *per edge* “share”. Consequently, MultipleRW oversamples G_A .

Figure 3.8 shows the results of four simulation runs and plots the evolution of the estimates of θ_{10} ($\hat{\theta}_{10}$) as a function of the number of steps. In this simulation note that: (1) FS quickly converges to a value that is close to the correct value; (2) two out of the four SingleRW runs overestimate θ_{10} and the remaining two underestimate it; (3) three out of the four MultipleRW runs converge to the same, incorrect, fraction (underestimating the true value of θ_{10}). FS is designed to be robust to disconnected or loosely connected subgraphs. All of the FS runs quickly converge to good estimates of θ_{10} . Figure 3.9 also shows that the NMSE for FS, SingleRW, and MultipleRW, that of FS is consistently lower.

3.5.2 Frontier v.s. Random Sampling

In Section 3.2 we show that if the degrees of two neighboring vertices are independent, random edge sampling is more accurate than random vertex sampling when it comes to estimating the tail of the degree distribution. In this section we observe

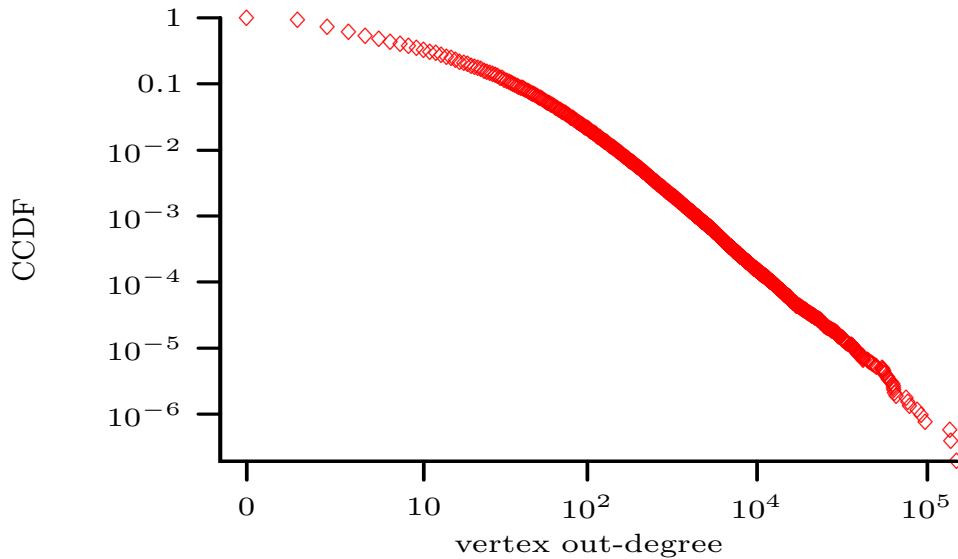


Figure 3.6. (Livejournal) Log-log plot of the out-degree CCDF.

this to be true over large real world graphs; we also observe that the accuracy of FS closely matches the accuracy of random edge sampling. In the following simulations we estimate the in-degree distribution. Random edge sampling uses the estimator $\hat{\theta}_i$, equation (3.5) (the estimator used for sampled vertices is trivial).

In this first simulation we set the sampling cost of random vertex sampling to one and random edge sampling has cost two (as each edge samples two vertices). The sampling budget is $B = |V|/100$. We label this simulation “100% hit ratio” to indicate the unitary cost of randomly sampling vertices. Figure 3.10 shows a log-log plot of the NMSE of our simulation over the (complete) Flickr graph. The vertical line indicates the average in-degree. Note that random edge sampling is more (less) accurate than random vertex sampling at estimating in-degrees larger (smaller) than the average in-degree, as predicted by equations (3.2) and (3.3) of our model in Section 3.2. We also observe that the accuracy of FS ($m = 1000$) closely matches the accuracy of random edge sampling.

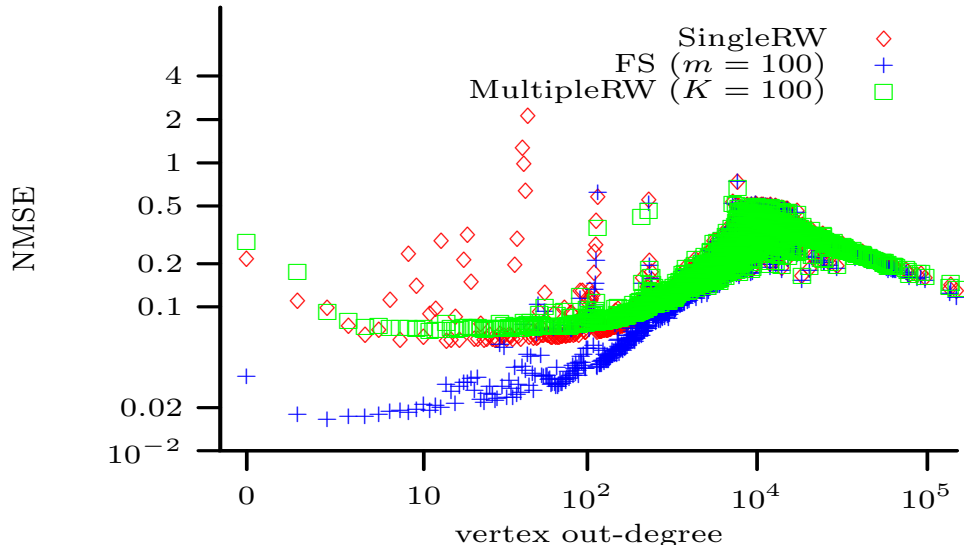


Figure 3.7. (Livejournal) The log-log plot of the NMSE of the out-degree distribution estimation with sampling budget $B = |V|/10$ (MSE over 10,000 runs).

Some complex networks exhibit a sparse user-id space. In this scenario a fraction of the sampling budget B can be spent querying invalid users-ids. Motivated by recent experiments over the MySpace network [56], the following experiment assumes that only 10% of the user-ids are valid, i.e., in average only one in every ten randomly sampled vertices are valid. We denote 10% to be the *hit ratio*. For random edge sampling we assume a *hit ratio* of 1% (the choice of 1% is arbitrary). Figure 3.11 shows a log-log plot of the NMSE of our simulation over the (complete) Livejournal graph with sampling budget $B = |V|/100 = 52844$. We observe that FS ($m = 1000$), which samples $m = 1000$ random vertices and (in average) crawls $B - 10m$ vertices, performs better than random edge sampling. Also note that FS estimates are more accurate than the estimates obtained from random vertex sampling for all but the three smallest in-degrees. This indicates that FS is more robust to low hit ratios than random vertex and edges sampling.

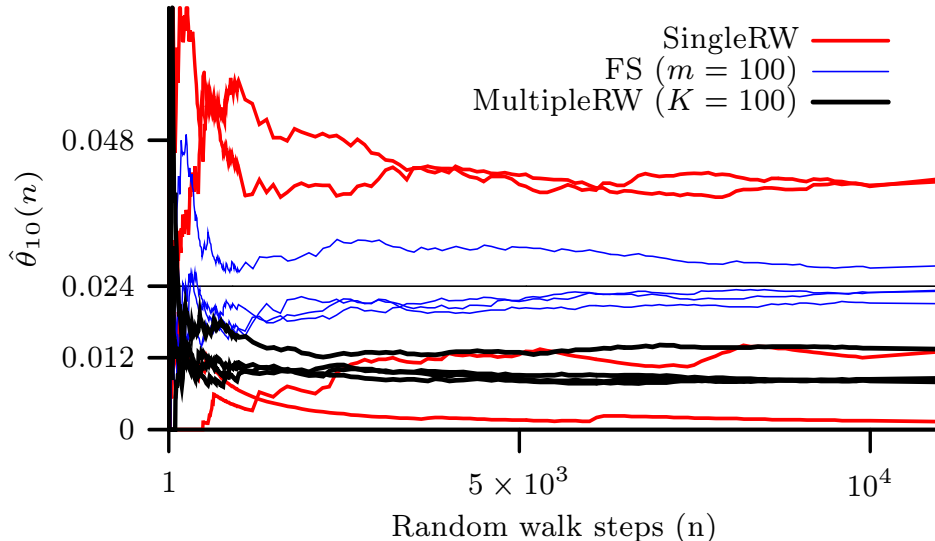


Figure 3.8. (G_{AB} graph) Four paths of $\hat{\theta}_{10}$ as a function of the number of steps n ($\theta_{10} = 0.024$).

3.5.3 Density of Special Interest Groups

In a variety of complex networks, e.g., on-line social networks, each vertex (user) is associated with multiple labels that represent group affiliations, e.g., user interests, user geolocation, among others. For example, in the Flickr graph 21% of the users belong to one or more special interest groups [50]. Let \mathcal{L} denote the set of groups in the Flickr graph and θ_l the fraction of vertices that belong to group $l \in \mathcal{L}$. In the simulations θ_l is estimated using FS ($m = 100$), SingleRW, and MultipleRW ($K = 100$) with budget $B = |V|/100$. Figure 3.12 shows the NMSE (from 10,000 runs) of the most popular 200 groups ordered in decreasing popularity. FS is clearly superior to both SingleRW and MultipleRW. Even when restricting the random walks to the largest connected component, FS still noticeably outperforms MultipleRW ($K = 100$) and SingleRW.

3.5.4 Global Clustering Coefficient Estimates

Here the accuracy of estimating the global clustering coefficient is evaluated using FS, SingleRW, and MultipleRW. The simulations show little difference between FS

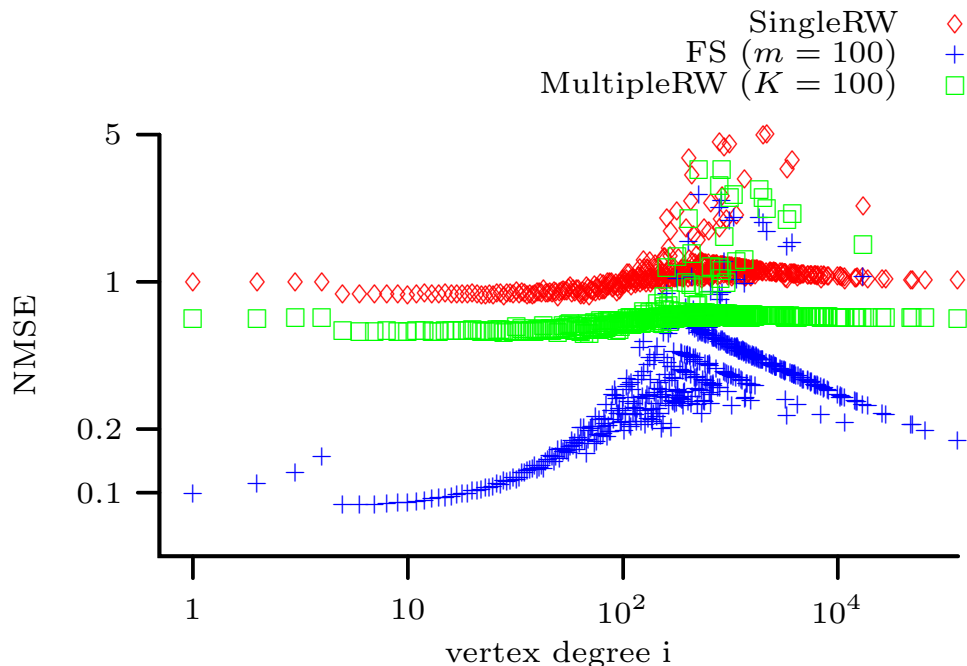


Figure 3.9. (G_{AB} graph) The log-log plot of the NMSE of the degree distribution estimation with sampling budget $B = |V|/10$ (MSE over 10,000 runs).

($m = 100$), SingleRW, MultipleRW ($K = 100$). Table 3.2 presents the average and the root mean squared error (\sqrt{MSE}) over 100,000 runs of the estimates obtained using FS over three graphs. From the results of Table 3.2 we observe that FS accurately estimates the global clustering coefficient.

Graph	Budget(B)	C	$\hat{C} \pm \sqrt{MSE}$
Joint Barabási-Albert	$ V /10$	10^{-4}	$10^{-4} \pm 10^{-5}$
Flickr	$ V /20$	0.05	0.05 ± 10^{-3}
LiveJournal	$ V /50$	0.06	0.06 ± 0.01

Table 3.2. Frontier sampling: global clustering coefficient estimates. C is the true value of the global clustering coefficient and \hat{C} is its estimated value.

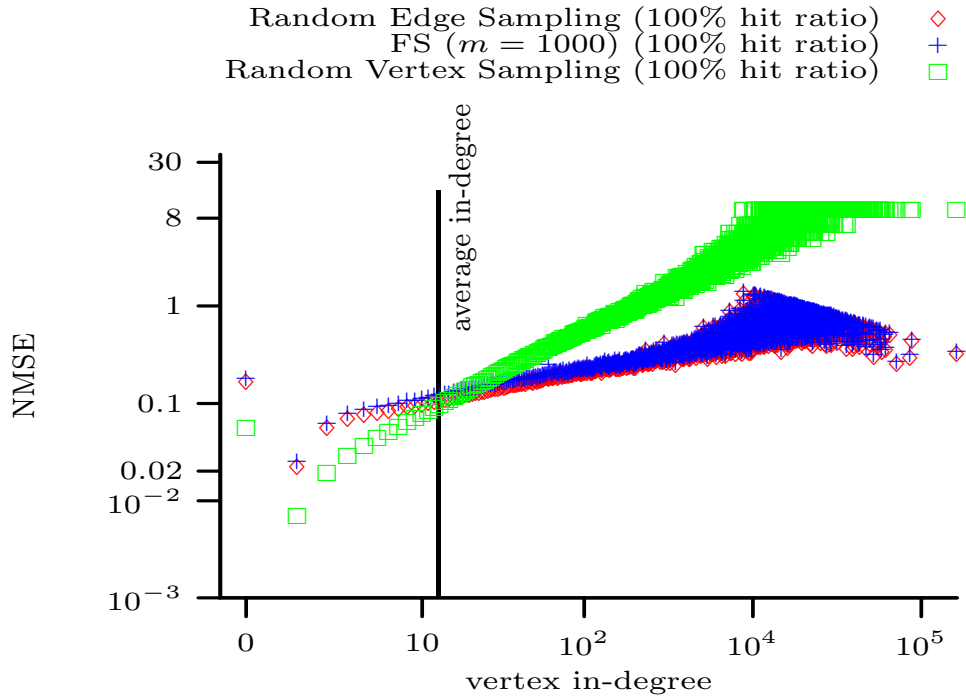


Figure 3.10. (Flickr) The log-log plot shows the NMSE of the in-degree distribution estimation with budget $B = |V|/100 = 18612$ (MSE over 10,000 runs).

3.5.5 Convergence to Stationarity

In this last set of experiments we see how fast FS, SingleRW, and MultipleRW converge to their stationary edge sampling probabilities. In this simulation $K \in \{1, 10\}$ (number of independent random walkers), $m = 10$ (Frontier sampling dimension) and restrict our analysis to the largest connected component of the three graphs in our datasets with the smallest number of vertices (in order to speed the computation): “Internet RLT”, “YouTube”, and “Hep-th”. Let $p_{u,v}^{(B)}$ denote the probability that a random walker, whose initial vertex is chosen uniformly at random, samples edge (u, v) at its the end of its sampling budget B . To measure the convergence to the stationary edge sampling probability, we use the largest relative difference between the stationary sampling probability $1/|E|$ and $p_{u,v}^{(B)}$:

$$\max_{(u,v) \in E} 1 - \frac{p_{u,v}^{(B)}}{1/|E|}.$$

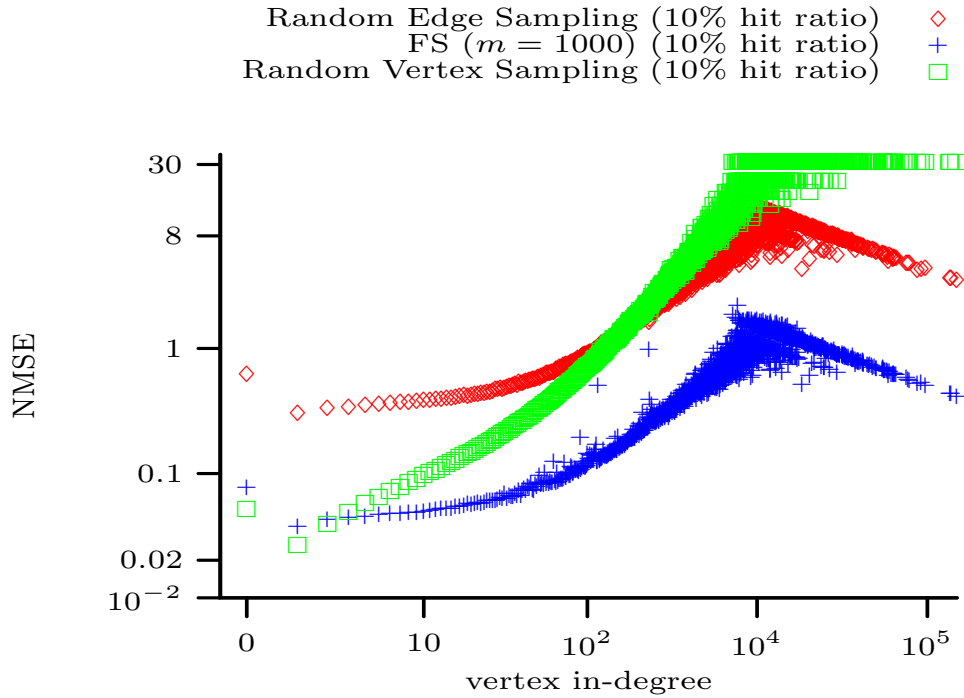


Figure 3.11. (Livejournal) The log-log plot shows the NMSE of the in-degree distribution estimation with budget $B = |V|/100 = 52844$ (MSE over 10,000 runs).

Table 3.3 presents a Monte Carlo estimate of this relative difference. The 95% confidence interval of the Monte Carlo simulation is $\pm 1\%$. The estimates show that the difference between the transient and the stationary edge sampling probabilities of independent random walkers are between 5 and 42 times larger than the difference of Frontier sampling. This means that Frontier sampling converges faster to stationarity edge sampling probability.

In what follows we see that FS sampling is well suited to be used in large scale (parallel, asynchronous) experiments.

3.6 Distributed Frontier Sampling

In what follows we see that FS sampling can be achieved by multiple independent random walkers (MultipleRW) where the cost of sampling a vertex v is an exponentially distributed random variable with parameter $\deg(v)$. The proof uses the Uniformization principle of Markov chains [10, Chapter 7.5] and the Poisson de-

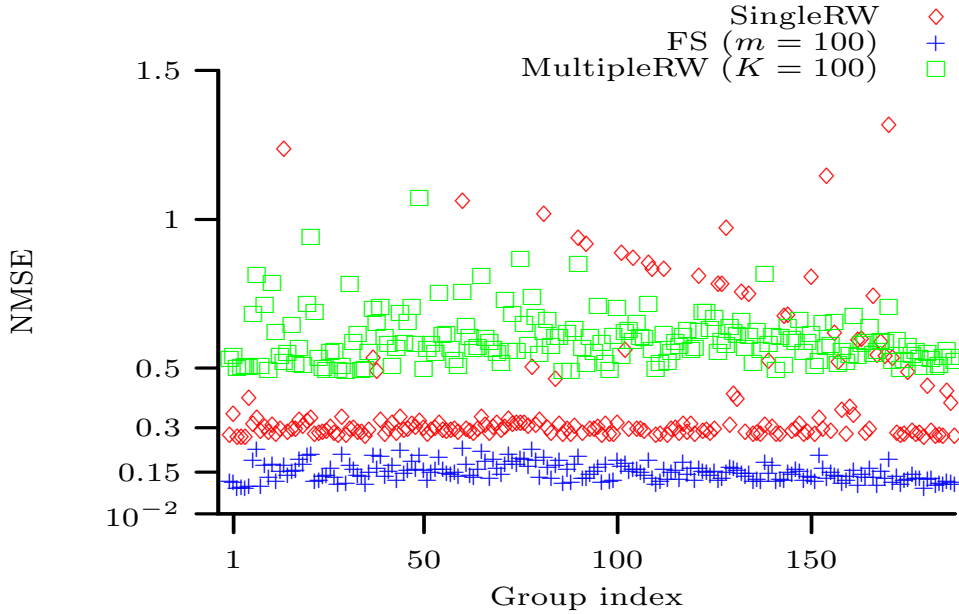


Figure 3.12. (Flickr) The NMSE of the density estimates of the most popular groups in the Flickr graph.

composition property to show that a MultipleRW with random exponential costs is equivalent to the FS sampling process described in Section 3.4, with the appropriate choice of budget B .

Let \mathbf{P} be the transition probability matrix of the Markov chain associated to a random walker over $G^m = (V^m, E_m)$, the m -th Cartesian power of G . Following Section 2.5.3 we have

$$\mathbf{P} = D^{-1}\mathbf{A},$$

where \mathbf{A} is the adjacency matrix of G^m and D is a diagonal matrix with $D_{i,i} = \sum_{v_j} \mathbf{A}_{i,j}$. According to Lemma 3.4.1 \mathbf{P} is also the transition probability matrix of FS in G . Let $\mathcal{M} = \{L_n \in V^m : n = 0, \dots\}$ denote the FS Markov chain (discrete-time), i.e., the transition probability matrix of \mathcal{M} is \mathbf{P} . Now let $\chi = \{X(t) \in V^m : t \geq 0\}$ be a continuous-time Markov chain with transition rate matrix

$$\mathbf{Q} = \mathbf{A} - D,$$

Graph	B (sampling budget)	Sampling prob. error		
		FS	MRW	SRW
Internet RLT	100	17%	257%	156%
YouTube	20	43%	236%	216%
Hep-Th	20	36%	1510%	781%

Table 3.3. Relative worst-case difference between the steady state and the transient edge sampling probabilities after $B - K$ steps. Frontier edge sampling probabilities are closer to steady state in all graphs. Legend: (FS) = Frontier sampling ($K = 10$), (SRW) = Single ($K = 1$) Random Walker, and (MRW) = Multiple ($K = 10$) Random Walkers.

observed during the (time) interval $[0, B]$. It is easy to see that the transition probability matrix of the embedded (discrete-time) Markov chain of χ , denoted by χ' , is

$$\mathbf{P}' = I - D^{-1}\mathbf{Q} = \mathbf{P}.$$

In the literature \mathbf{P}' is known as the *Uniformized* counterpart of \mathbf{Q} (with unitary uniformization rate) [10, Chapter 7.5]. Because $\mathbf{P}' = \mathbf{P}$, the stochastic processes χ' and \mathcal{M} are equivalent.

Let $L'_n = (v_1, \dots, v_m)$ denote the state of χ before the n -th step. Now note that because all off-diagonal non-zero transition rates in \mathbf{Q} are equal to one, the probability that the k -th random walker transitions out of vertex v_k is independent of the state of all the other random walkers in L_n . Thus, we can decompose the Poisson process describing a departure from the state $L'_n = (v_1, \dots, v_m)$ into m independent stochastic processes, where the i -th process is a Poisson process with rate $\lambda_i = \deg(v_i)$, $i = 1, \dots, m$. The above is equivalent to the stochastic process of a discrete-time MultipleRW with m random walkers and budget B , where the cost of sampling a vertex v is an exponentially distributed random variable with rate $\mu_v = \deg(v)$.

3.7 Related work

This section is devoted to review the related literature. FS can be classified as a Markov Chain Monte Carlo (MCMC) method. Other MCMC-based methods are applied to characterize complex networks. Applications include, but are not limited to estimating: characteristics of a population [64] (e.g. estimation of HIV seroprevalence among drug users [48]), content density in peer-to-peer networks [30, 47, 54, 61], uniformly sampling Web pages from the Internet [33, 59], and uniformly sampling Web pages from a search engine’s index [4]. The above literature is mostly concerned with random walks that seek to sample vertices uniformly (also known as Metropolized Random Walks or Metropolis-RW) [30, 33, 59, 4, 61]. The accuracy of RW and Metropolis-RW is compared in [29, 54], and in a variety of experiments RW estimates are consistently more accurate than or equal to MRW estimates.

The above literature does not consider the use of multiple random walks to address the problem of estimating characteristics of disconnected or loosely connected graphs. While multiple independent random walkers have been used as a convergence test in the literature, the simulations presented in Section 3.5 show that independent walkers are not suited to sample loosely connected graphs when the starting vertices are selected uniformly at random.

A number of real complex networks are known to have disconnected or loosely connected subgraphs. A large body of MCMC literature is dedicated to overcome the locality problem described in Section 3.3.3. However, the literature either assumes that the graph is very structured, e.g., a 2 dimensional lattice, or that the graph is completely known. These assumptions make the solutions inapplicable to our problem. A comprehensive list of MCMC methods and their characteristics can be found in [58].

Projecting a RW onto a higher dimensional space has been used in [11] to turn the Markov chain associated to the random walker nonreversible, which can speed

up the mixing of the original RW. Unfortunately, it is unclear if this method can be successfully used to estimate characteristics of complex networks.

In networks that cannot be crawled (e.g., the Internet topology), samples must be obtained along shortest paths, and vertex degrees cannot be queried, [1] shows that observed vertex degrees are biased. This chapter, however, assumes graph can be crawled and vertex degrees queried. The scenario described in this chapter admits a RW with a unbiased estimator. Multiple random walks also find other applications besides the one presented in this work. They are used to collect Web data [13], search P2P networks [7, 70], and decrease the time to discover “new wireless nodes” [2]. Dependent multiple random walks are also used in percolation theory [3].

3.8 Discussion and Future work

This chapter presented a new promising random walk-based method (*Frontier sampling*) that mitigates the estimation errors caused by subgraphs that “trap” a random walker. Frontier sampling (FS) uses multiple (m) mutually *dependent* random walker. The dependence between walkers is designed to “better balance” their samples. These samples are shown to be the projection (onto the original graph) of a special type of m -dimensional (single) random walker. Simulations over real world graphs in Section 3.5 show that Frontier sampling (FS) is more robust than single and multiple independent random walkers to estimate degree distributions and the fraction of users that belong to a social group. This chapter also presents evidence, using an analytical argument (also substantiated by simulations), that random walks (in particular, FS) are better suited to estimate the tail of power law graphs than random vertex sampling.

Moreover, FS sampling is well suited to be used in large scale (parallel, asynchronous) experiments. This is because FS sampling can be achieved by multiple independent random walkers where the cost of sampling a vertex v is an exponentially

distributed random variable with parameter $\deg(v)$. Using the the Uniformization principle of Markov chains [10, Chapter 7.5] and the Poisson decomposition property, Section 3.6 shows that this MultipleRW with random exponential costs is equivalent to the FS sampling process described in Section 3.4, with the appropriate choice of budget B .

The ideas behind FS can have far reaching implications, from estimating characteristics of dynamic networks to the design of new MCMC-based approximation algorithms.

CHAPTER 4

DESIGNING TCP FLOW-LEVEL ESTIMATORS FROM SAMPLED PACKETS

4.1 Introduction

Estimation can be difficult if the characteristic of interest (e.g., the TCP flow size) cannot be directly measured from the observations (e.g, sampled packets). We refer to these characteristics as *latent network characteristics*. Often, latent characteristics need to be estimated, by applications or measurement apparatus, from other observable characteristics. Such estimation requires a model-based *measurement method* that correlates the unobserved characteristics with the observed ones using a mathematical model.

The theory behind the estimation of latent characteristics had a significantly step forward in 1935, when R. A. Fisher’s book *The design of experiments* [23] shed a new light onto agricultural field experiments. Fisher’s questions ignited a revolution in the field of *estimation theory*. In 1947, Fisher summarized his views of the statistician’s job [24]:

“[The finding] that the amount of information extracted in the process of estimation could never exceed the quantity supplied by the data [...], combined with the practical fact that directly available processes of computation would extract almost always a very large fraction of the total available [information], shifted the moral balance. ... The weight of [the statistician’s] responsibility was thrown back on to the process by which

the data had come into existence. So armed with [the] amount of information as a practical tool, statisticians came to study what forms of experiment, or what types of observational programs would yield the most information for a given expenditure in time, money and labor.”

Since Fisher’s foundational work, statisticians have used the tools Fisher helped develop to design *measurement methods*. Here *measurement method* (which Fisher refers to as “experiment” or “observational program”) refers to the procedure in which one collects observable data that helps to estimate (partially) latent characteristics of the system under study.

It is rather surprising to find that the above part of Fisher’s lifetime work has been all but ignored in the Computer Science literature. This absence is especially noticeable in the design of network measurements. In what follows we consider the estimation of TCP flow size distributions from randomly sampled packets. We use this example to illustrate how Fisher’s foundational work can be applied to help design better network measurement methods.

4.2 Estimating the TCP flow size distribution from randomly sampled packets

This chapter considers the problem of estimating flow size distributions by sampling packets at a chosen point (router) in the network. Packets are sampled according to a Bernoulli process with sampling probability p , $0 < p < 1$. Random packet sampling is widely used in network monitoring to reduce the workload of the monitoring apparatus (the monitoring apparatus is typically a router). These sampled packet streams can then be used to estimate flow-level characteristics of network traffic [19]. Flows are disjoint subsets of packets such that every packet belongs to a flow and no packet belongs to more than one flow. The conventional IP flow definition is a set of packets that obey the following rules:

- Any two packets have the same 5-tuple, i.e., the same IP Source, IP Destination, source port number, destination port number, and protocol number.
- Maximum inter-packet arrival time must be less than a threshold t , where t is a value given by the network operator, typically between 30 to 60 seconds.

One of the most relevant flow-level characteristics is the flow size distribution, i.e. the fraction of flows that contains i packets, $i = 1, 2, \dots$. This is an important metric for many applications, such as traffic engineering, denial of service attack monitoring, and worm/virus outbreak detections. It is easy to see that sampling packets can affect the flow size distribution. Large flows are sampled with high probability but have their original number of packets, i , reduced in average to $i \cdot p$, while most small flows will not be sampled at all. To date there is conflicting evidence on the quality of the resulting estimates.

In 2003 Hohn and Veitch [34] proved that it is impossible (in practice) to accurately estimate the fraction of small flow sizes using the inversion estimator. The inversion estimator is a simple and fast estimator based on linear algebra. However, it is not the most accurate estimator for most problems. In the same year and using a similar data set, Duffield, Lund, and Thorup [19] provided several estimators and argued that, in practice, they do a good job at estimating the same metric. Duffield et al. [19], in the light of this seemingly contradictory result, argues that the inversion estimator has a higher variance than their proposed estimator. They also argues, without a formal proof, that its maximum likelihood estimator does not exhibit the same high variance problem. In 1947 Fisher believed that this type of debate was all but extinct [24]. These contradicting results make this a good application to the Fisher information, the constrained Cramér-Rao bound, and the data processing inequality, which were introduced in Chapter 2.

4.3 Contributions

In 2006 Ribeiro et al. [57] used the Fisher information, the Cramér-Rao and data processing inequalities to show that:

- Using the measurement method of Hohn and Veitch [34] and Duffield et al. [19] (over similar data sets¹) **no algorithm** can obtain accurate unbiased flow size distribution estimates **under a realistic scenario**. An unbiased estimate is an estimate whose average is the true average, i.e., if $\hat{\theta}$ is an unbiased estimate of θ then $E[\hat{\theta}] = \theta$.
- If flows are TCP flows, a measurement method that takes protocol information into account can obtain accurate flow size distributions estimates. (A TCP flow is a flow that contains only packets from a single TCP session.)

In this chapter I present these results. In what follows the flows prior to sampling are referred to as *original flows*. A sampled (or thinned) flow is a flow that has at least one packet sampled. A flow of size i is a flow that originally has i packets. Likewise, a sampled flow of size m is a flow that has m packets sampled, where $m \geq 1$. In practice some original flows are not sampled and therefore not observed. Some original flows may split into multiple sampled flows. Here we do not account for flow splitting. Our goal is to estimate the original flow size distribution from the sampled flow size distribution.

4.4 Fisher information from sampled packets

As packets are sampled independently according to a Bernoulli process, then flows are also sampled independently. Applying Lemma 2.3.1 we have that if J is the Fisher information of one sampled flow, then the Fisher Information of n sampled flows is

¹These results should also hold for other real Internet data sets.

nJ . Therefore, we focus on the Fisher information of a single sampled flow. Assume a maximum flow size $w \geq 2$ and let $\theta = (\theta_1, \dots, \theta_w)$, where $\theta_i > 0$, $1 \leq i \leq w$, be the fraction of flows with size i . Let n be the number of sampled flows and \hat{s}_j , $j \geq 0$, denote the total number of sampled flows with j sampled packets. We can also further define the empirical sampled flow size distribution $\hat{d} = [\hat{s}_j/n]$. Let d be the true sampled flow size distribution. Distributions d and θ are related by

$$d_j = \sum_{i=1}^w b_{ij} \theta_i, \quad (4.1)$$

where b_{ij} is the binomial probability of sampling j packets out of i original packets given sampling probability p . Equation (4.1) can be written in matrix notation as

$$d = \mathbf{B}\theta, \quad (4.2)$$

where \mathbf{B} is a $w \times w$ matrix whose element (i, j) is b_{ji} . Matrix \mathbf{B} is an upper triangular matrix and thus (4.2) has a unique solution. The inversion estimator of Hohn and Veitch [34] is just $\hat{\theta} = \mathbf{B}^{-1}\hat{d}$. Now let D denote a random variable of the sampled flow size distribution and $P[D = \hat{d}]$ denote the probability that the sampled distribution is \hat{d} . It is easy to see that $E[D] = d$ and \hat{d} is a sample of D .

Lets now define the likelihood function over a single sampled flow. As we have just a single sampled flow, if the flow contains j sampled packets, $\hat{d}_j = 1$ and $\hat{d}_k = 0$, $\forall k \neq j$. The probability that a flow has j packets sampled is $d_j = (\mathbf{B}\theta)_j$. Thus,

$$P[D = \hat{d} | \theta] = \sum_{j=1}^w \hat{d}_j (\mathbf{B}\theta)_j. \quad (4.3)$$

Note that θ is constrained by:

$$\sum_{\forall i} \theta_i = 1 \quad (4.4)$$

and

$$0 < \theta_i < 1, \forall i. \quad (4.5)$$

Using the likelihood function of equation (4.3) and the results of Chapter 2.3.2 we obtain the constrained Cramér-Rao bound for the flow size distribution problem.

Applying the constrained Cramér-Rao inequality: An example with $w = 2$

Let $w = 2$ be the maximum flow size, $\theta = (0.88, 0.12)$, and $p = 0.01$. From eq. (2.3.2), $(\mathbf{G})_i = \theta_i^2/(\theta_i - 1)$. Equation (2.3.2) yields

$$J(\beta^{-1}(\theta_1)) = - \sum_{j=1}^2 \mathbf{a}^{(j)} (\mathbf{a}^{(j)})^\top / d_j,$$

where $\mathbf{a}^{(j)} = (b_{1,j}, b_{2,j}) \cdot (\theta^2/(\theta - 1))$. Let j denote the number of sampled packets in a flow with a SYN packet. Then $b_{1,1} = 1$, $b_{1,2} = 0$, $b_{2,1} = 0.99$ and $b_{2,2} = 0.01$. The inverse of the Fisher information \mathcal{I}^{-1} (equation (2.7)) of one sampled flow is

$$\mathcal{I}^{-1} = \begin{bmatrix} -1078 & 1078 \\ 1078 & -1078 \end{bmatrix}$$

Now assume n flows are sampled. Thus the lower bound on the mean squared error of estimates $\hat{\gamma}_1$ and $\hat{\gamma}_2$ obtained using the Cramér-Rao bound will be $E[(\gamma_1 - \hat{\gamma}_1)^2] \geq 1078/n$ and $E[(\gamma_2 - \hat{\gamma}_2)^2] \geq 1078/n$. The Cramér-Rao bound of parameters θ comes from the delta method as seen in Section 2.3.2. Matrix \mathbf{H} is

$$\mathbf{H} = \begin{bmatrix} 0.105 & 0 \\ 0 & 0.105 \end{bmatrix}.$$

An application of eq. (2.8) yields that the mean squared error of any unbiased estimates $\hat{\theta}_1$ and $\hat{\theta}_2$ of θ_1 and θ_2 respectively to be: $E[(\theta_1 - \hat{\theta}_1)^2] \geq 1092/n$ and $E[(\theta_2 - \hat{\theta}_2)^2] \geq 1092/n$ for n sampled flows, given n sufficiently large.

4.5 Simplifications to constrained Cramér-Rao inequality

The results presented above (published in Ribeiro et al. [57]), were later simplified assuming that the inequality constraint in equation (4.5) does not contribute to the Fisher information or the Cramér-Rao lower bound [63]. It is worth noting that the Cramér-Rao lower bound obtained ignoring the inequality of equation (4.5) can be violated by an estimator that “projects” the unconstrained estimated results into the constrained space of the parameters (“Remark 8” in [31], also there is an extra complication that many (but not all) of such estimators are biased). However, I believe that in most practical scenarios and for a large enough maximum flow size (e.g. $W_{max} > 5$) the inequality and equality (equation (4.4)) constraints contribute little to the value of the Fisher information. In fact, our experiments indicate that the contribution of the inequality constraint to the Fisher information is minimal. Moreover, if considering only the equality constraint, equation (2.7) simplifies to [63]

$$\mathcal{I}^+ = J^{-1} - \theta\theta^\top, \quad (4.6)$$

where J^{-1} is the inverse of the unconstrained Fisher information (equation (2.3.2)). Equation (4.6) shows that the equality constraint reduces the Cramér-Rao bound over the estimate of θ_i by θ_i^2/n if compared to the unconstrained bound $(J^{-1})_{ii}/n$. Thus, if $(J^{-1})_{ii}$ is large, the equality constraint contributes very little to the accuracy of the estimates.

4.6 Designing summary functions

The inverse of the Fisher information allows us to verify if the “statistical information” contained in the summary d suffices to accurately estimate θ . Even better, we are able to determine how many sampled flows are necessary to achieve a given Mean Squared Error (MSE) bound (Cramér-Rao bound). Moreover, we can test different

types of summaries. Distinct summaries modify the matrix \mathbf{B} of equation (4.2). In this section we see how much these different matrices \mathbf{B} affect the Fisher information.

The data processing inequality (Chapter 2.4) states that adding information can only increase the amount of Fisher information. Thus, we expect that a good measurement method using extra information to perform better, or at least no worse, than a good measurement method that does not use the extra information. This clearly holds as one can always throw the extra information away inside the estimator. Here we consider a measurement method similar to the one used in Hohn and Veitch [34] and Duffield et al. [19] but that also extracts the following TCP protocol information from the sampled packets:

- **SYN flag:** Here only the first sampled packet of the flow can have the SYN flag. (Duffield et al. [19] also presents a method with SYN flag information, but in their method all flows without SYN packets are discarded, which is not our case).
- **TCP sequence number (SEQ):** Here, given the assumptions discussed in Section 4.6.4, the TCP sequence number can be used to obtain the number of packets between any two sampled packets of the same flow.

4.6.1 Real Internet Traces

The Fisher information depends on the flow size distribution θ . To evaluate distinct summary functions we use an empirical flow size distribution, θ , based on packet traces collected from a Tier-1 ISP's backbone network. These packet traces are collected using IPMON, a passive measurement system that captures the first 64 bytes IP packet header of every packet on an optical link [26]. The statistics of these traces are listed in Table 4.1. The BB-East-1 and BB-East-2 traces are taken from two OC-48 links between backbone routers on the east coast. The Access-East trace is from an access link in the east coast.

Unfortunately, the Fisher information matrix analysis requires flow sizes to be in the range $\{1, \dots, w\}$. The following evaluation focuses on small flow sizes. To reduce the computational cost, all numerical analysis uses $w = 10, 20, 30, 40, 50$ and renormalizes θ accordingly. It is interesting to note that in my experiments increasing w beyond 50 has little impact over the Fisher information of small flows. The numerical results presented next were obtained using the flow size distribution of the BB-East-2 trace. The results for the other packet traces are similar.

Trace	Avg. Rate	Active Flows	Duration
Access-East	373Mbps	61,000/sec	2 hours
BB-East-1	867Mbps	140,000/sec	2 hours
BB-East-2	25Mbps	5,000/sec	2 hours

Table 4.1. Trace Statistics

4.6.2 No protocol information

Without protocol information the summary function can only count the number of packets in a flow. In this case b_{ij} is the binomial probability of sampling j packets out of i original packets given sampling probability p . The inverse of the constrained Fisher information (\mathcal{I}_{ii}^+) (for the BB-East-2 trace) is $\mathcal{I}_{11}^+ < 10^{15}$. From the Cramér-Rao inequality we know that $\sqrt{\text{MSE}} \geq \sqrt{\mathcal{I}_{11}^+/n}$, where n is the number of sampled flows. This means that achieving $\sqrt{\text{MSE}} < 0.5$ requires $n > 4 \times 10^{15}$. This is a **huge** number of sampled flows. To give an idea how large is 4×10^{15} flows, since its creation the whole Internet has not yet carried these many flows. Thus, without protocol information it is **impossible** (in practice) to obtain **accurate unbiased estimates** of θ using packet sampling. This result generalizes the observation in Hohn and Veitch [34] to any type of unbiased estimator. Figure 4.1 shows the inverse Fisher information \mathcal{I}_{ii}^+ for flow sizes $i = 1, \dots, 20$.

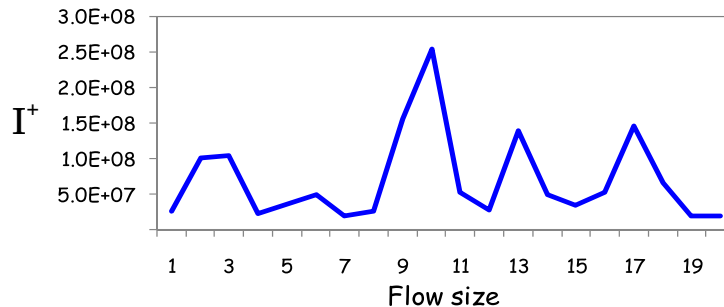


Figure 4.1. Inverse Fisher information $(\mathcal{I}^+)_{ii}$ (i is the flow size) without protocol information.

In what follows we see whether adding protocol information to the summary function improves the information content of the data used in the estimation phase.

4.6.3 TCP SYN flag information

Here we consider adding the TCP SYN flag information to the summary. In this scenario matrix \mathbf{B} needs to be redefined. The modification to \mathbf{B} is simple to understand. The SYN packet is one unique specially marked packet in the flow. If we do not sample the SYN packet of the flow, it means that the flow has one extra packet that was not sampled. This is all of the information the SYN packet encodes. The change in \mathbf{B} is straightforward. b_{ij} is the binomial probability of sampling $j - 1$ packets out of $i - 1$ original packets given sampling probability p , with $b_{11} = 1$. In this scenario, achieving $\sqrt{\text{MSE}} < 0.5$ requires $n > 1.6 \times 10^{15}$. While there is the TCP SYN flag increases in accuracy (Fisher information), it is still insufficient to obtain an accurate estimate. Figure 4.2 shows the inverse Fisher information \mathcal{I}_{ii}^+ for flow sizes $i = 1, \dots, 20$.

4.6.4 TCP SEQ + SYN flag information

TCP uses a 32-bit sequence number that counts payload bytes in a flow. An estimator that measures flow sizes in number of bytes can clearly benefit from TCP sequence numbers. The question is whether an estimator using packet counts can

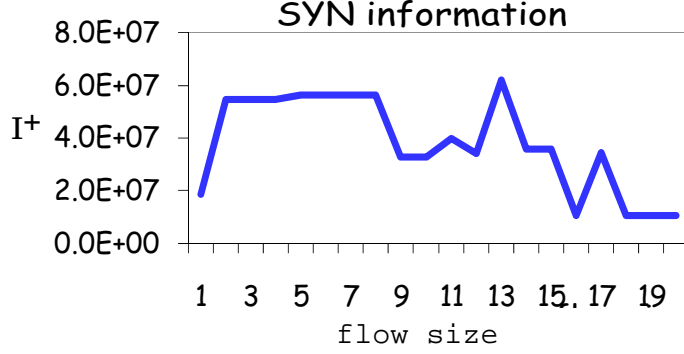


Figure 4.2. Inverse Fisher information $(\mathcal{I}^+)_{ii}$ (i is the flow size) with TCP SYN protocol information.

also benefit from sequence numbers. Assume that there is a function $h(s_a, s_b)$ that takes two TCP sequence numbers s_a and s_b from two distinct packets a and b of the same flow and returns the number of packets sent between a and b including a and b . This is not trivial to compute h in real life experiments due to duplicated packets and packets of different sizes that belong to the same flow. Appendix A.1 provides a reasonably an approximation to h that performs well in practice.

Let $s_{min}^{(u)}, s_{max}^{(u)}$ be the smallest/greatest sampled TCP sequence number values of flow u (wraps around are easily treated if the sampling probability is not too low, e.g., $> 10^{-4}$). Let $r = h(s_{min}^{(u)}, s_{max}^{(u)}) \in \{0, \dots, w - 2\}$ be the number of packets sent between the smallest and the greatest sampled TCP sequence numbers. Let subscript SYN (NOSYN) denote a sampled flow with (without) a sampled SYN packet. Let

$$b'_{i,(\text{SYN},r)} = p(1-p)^{i-r}$$

denote the probability that a flow of size i has a sampled SYN packet and has $r = h(s_{min}^{(u)}, s_{max}^{(u)})$. Let

$$b'_{i,(\text{NOSYN},r)} = (i-r)p(1-p)^{i-r}$$

denote the probability that a flow of size i does not have a sampled SYN packet and has $r = h(s_{min}^{(u)}, s_{max}^{(u)})$. An element i, j of matrix \mathbf{B} is

Measurement method	Minimum no. of sampled flows
Hohn and Veitch [34] and Duffield et al. [19]	4×10^{15}
Duffield et al. [19] (SYN flag)	1.6×10^{15}
Ribeiro et al. [57] (SYN + SEQ)	1.6×10^3

Table 4.2. Minimum number of sampled flows that an unbiased estimator needs in order to achieve $\sqrt{E[(\hat{\theta}_1 - \theta_1)^2]} < 0.5$. Results for $w = 50$, $p = 1/200$, obtained with the flow size distribution of the BB-East-2 trace.

$$b_{i,j} = b'_{i,j} / \sum_{\forall j} b'_{i,j},$$

where

$$j \in \{(\text{NOSYN}, 0), \dots, (\text{NOSYN}, w - 1), (\text{SYN}, 0), \dots, (\text{SYN}, w - 1)\}.$$

In this scenario, achieving $\sqrt{\text{MSE}} < 0.5$ requires $n > 1.6 \times 10^3$, which is 12 (!) orders of magnitude fewer samples than estimating with SYN flags alone or without protocol information. Table 4.2 summarizes the number of samples required to achieve $\sqrt{\text{MSE}} < 0.5$ for flows of size one in the BB-East-2 trace.

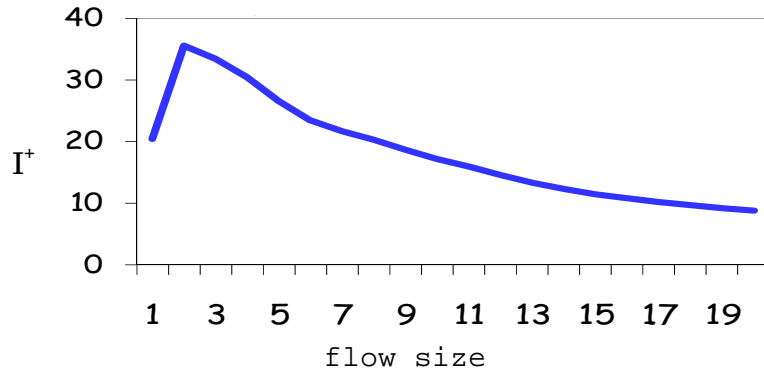


Figure 4.3. Inverse Fisher information $(\mathcal{I}^+)_{ii}$ (i is the flow size) with TCP SYN and TCP SEQ protocol information.

4.7 Simulation results

The Maximum log-Likelihood Estimator (MLLE), finds a set of parameters $\hat{\theta}$ that maximize the log-likelihood of the sampled data. Under the same regularity conditions required by the Cramér-Rao bound, the MLLE is an asymptotically *efficient* unbiased estimator of θ , i.e., its error achieves the Cramér-Rao lower bound as the number of samples tends to infinity. As in practice we do not have a very large number of samples, we would like it to be close to optimal even with the number of samples typically collected at Tier-1 backbone routers. This section presents the estimators of the summary functions proposed in Section 4.6. In particular we see that using the TCP SEQ + SYN flag summary, the MLLE does not require a large number of samples to be unbiased and achieve the Cramér-Rao error lower bound. In addition, we present a conjugate gradients algorithm for the MLLE, a faster convergence algorithm than the commonly used Expectation Maximization algorithm.

The MLLE uses penalty functions to make sure the estimate $\hat{\theta}$ lies within the region defined by the constraints in equations (4.4) and (4.5). When a value θ_i violates one of the constraints, the MLLE receives a penalty, which then forces the search to remain within the constrained region. In the first part of this section we estimate the flow size distribution using only SYN flags in the summary function. This, of course, does not account for the “noise” introduced by flow-splitting, which splits one long original flow into two or more shorter ones. This chapter does not account for flow splitting, although [37] shows that is possible to do so. Next we review an algorithm that computes the MLLE.

4.7.1 MLE with conjugate gradients

Let n denote the number of sampled flows and \hat{d}_j be the fraction of sampled flows with index j . The interpretation of index j depends on the summary function used. For instance, in the summary with no protocol information j is the number of sampled

packets. The MLLE can be written as

$$\hat{\theta} = \arg \max_{\hat{\theta}} n \sum_{\forall j} \hat{d}_j \ln(\mathbf{B}\hat{\theta})_j, \quad (4.7)$$

subject to $\sum_i \hat{\theta}_i = 1$ and $0 < \hat{\theta}_i < 1, \forall i \in \{1, \dots, W\}$.

There are multiple ways to find a value for $\hat{\theta}$ that satisfies equation (4.7). One of them, used in [19], is the Expectation Maximization algorithm. Although the EM algorithm is sound, needs no fine tuning, and is guaranteed to always improve the estimate at each step, in practice it can suffer from slow convergence [55]. More specifically, Theorem 5.2 in [55] shows that if the parameters θ are “poorly separable” in the likelihood function then EM exhibits a slow convergence rate. The term “poorly separable” can be quantified as the difficulty of distinguishing whether a sample j came from flow sizes i or i' with $i \neq i'$, i.e., if $b_{i,j}\theta_i \approx b_{i',j}\theta_{i'}$. Unfortunately, flow size estimation suffers from this problem. Although one expects that other maximum likelihood algorithms will also suffer with these “poorly separable” parameters, it is believed that in practice the effect is felt more by EM [55] (conjecture is strengthened by the practical experience accumulated in this research, trying to apply the EM and the conjugate gradients methods to the flow size estimation problem).

Instead, we use the method of conjugate gradients [53] to compute a solution to (4.7). The conjugate gradient MLE algorithm is implemented with the help of the `wnlib` library². For the above algorithm to work, we need to provide the matrix \mathbf{B} and the gradient of the log likelihood function, $\nabla_{\theta} \ln P[D = \hat{d} | \theta]$, conditioned on $\sum_{i=1}^w \theta_i = 1$. The i th component of this gradient is

$$\frac{\partial}{\partial \theta_i} P[D = \hat{d} | \theta] = \sum_{\forall j} \frac{b_{i,j} \hat{d}_j}{\sum_{r=1}^w \hat{\theta}_r b_{r,j}} - 1.$$

²<http://www.willnaylor.com/wnlib.html>

The constraints $0 < \hat{\theta}_i < 1$, $\forall i \in \{1, \dots, W\}$ are introduced as penalty functions. The algorithm is initialized with $\hat{\theta}^{(0)} = (1/w, \dots, 1/w)$, where w is the maximum flow size.

4.7.2 Results

The following simulation uses packet sampling probability $p = 1/200$ and maximum flow size $w = 50$. Figure 4.4 shows a graph with the original flow size distribution and its MLE estimate using the TCP SYN and TCP SEQ+SYN summaries (showing the mean and root mean squared error of the estimates). We denote the MLE applied to the SYN and SEQ+SYN summaries as SYN MLE and SEQ+SYN MLE, respectively. As expected, the estimates from the SYN MLE are inaccurate because the summary does not contain enough information. The estimates from the SYN MLE are also extremely sensitive to the initialization point $\hat{\theta}^{(0)}$. On the other hand, Figure 4.4 shows that the SEQ+SYN MLE estimate is accurate and insensitive to the initialization point $\hat{\theta}^{(0)}$. In what follows we compare the mean squared error of the SEQ+SYN MLE estimates with the error bound given by the constrained Cramér-Rao inequality of Chapter 2.3.2.

4.7.3 MLE for SEQ+SYN summary: an efficient estimator

Here we see that the SEQ+SYN MLE is able to come close to the smallest possible error allowed by the Cramér-Rao inequality. Figure 4.5 shows the mean standard deviation error of SEQ+SYN MLE estimates compared to its respective inverse Fisher information. For a large number of sampled flows (10^8) we see that the Cramér-Rao inequality is tight and the SEQ+SYN MLE is an *efficient* estimator when we have 10^8 sampled flows. An estimator is called *efficient* when its mean squared error reaches the lower bound of the Cramér-Rao inequality (i.e., it is equal to the inverse of the Fisher information matrix). For a much smaller sample set, 260,000 sampled flows, there is a small bias in the estimates (which allows the estimator MSE to violate

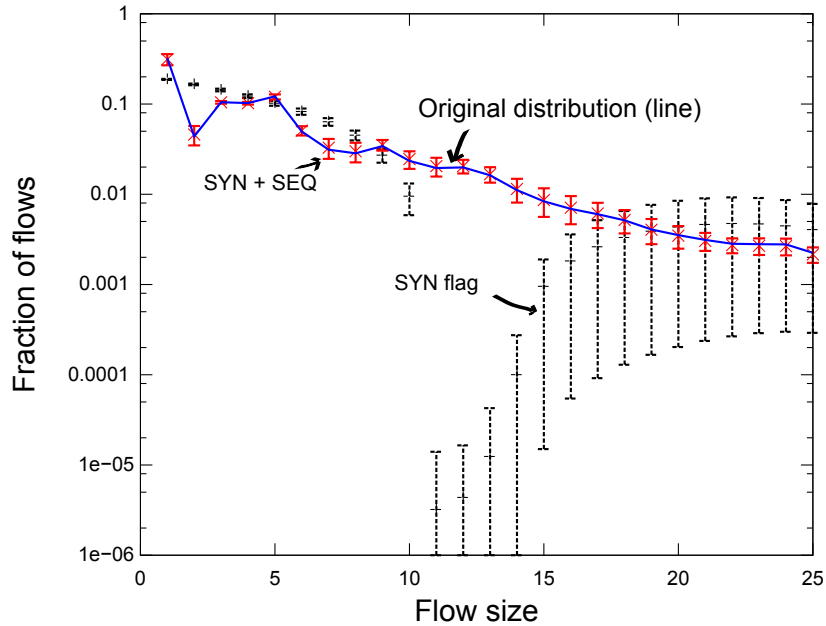


Figure 4.4. Estimates from 120 runs with 5×10^9 sampled flows and $p = 1/200$. Summary function protocol information: TCP SYN flag against TCP SEQ+SYN flag. Note the strange behavior of the estimates from the SYN flag summary. This happens due to the low Fisher information in the sampled flows.

the Cramér-Rao bound). The root mean squared error is fairly close to the lower bound of the Cramér-Rao inequality. Thus, one can argue that the SEQ+SYN MLE is almost *efficient* for practical purposes even with only 260,000 sampled flows.

4.8 Conclusions

The Fisher information, the Cramér-Rao inequality, and the data processing inequality are powerful tools for the design of measurement methods. In the example presented in this chapter (flow size distribution estimation) these tools allow us to answer the questions proposed in Chapter 1:

- *It possible to accurately estimate the original flow size distribution from the sampled packets?*

Answer: Yes, as long as protocol information is used in the summary function and the estimator.

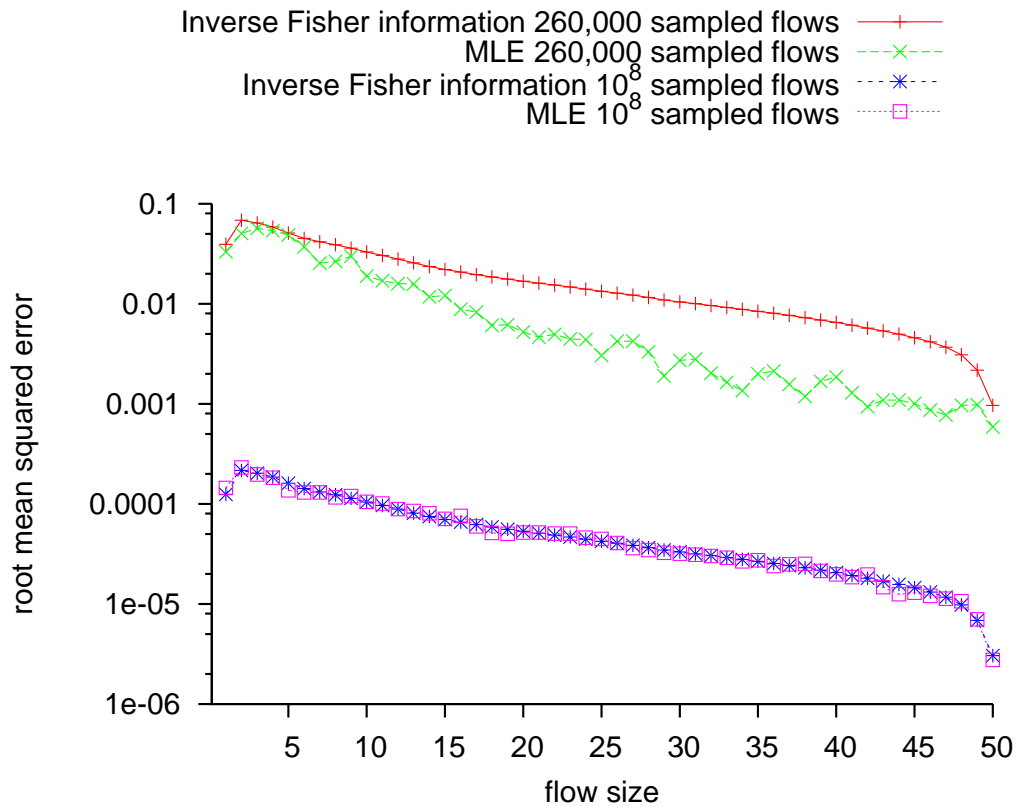


Figure 4.5. This graph compares the root mean squared error of the MLE estimate with the inverse of the Fisher information (which, according to the Cramér-Rao inequality, is a bound for the mean squared error of the MLE).

- *Are there other observable characteristics (besides the number of packets sampled) that could yield more information about the original flow size distribution?*

Answer: Yes, the TCP Sequence Numbers contain a lot of information about θ . The SYN flag, on the other hand, contains a small amount of information.

This chapter also shows that the SEQ+SYN MLE comes close to the Cramér-Rao bound, even for small sample set sizes.

CHAPTER 5

DESIGNING A STREAMING ALGORITHM

5.1 Introduction

A data stream is an ordered sequence of n data items that can be read only once. A streaming algorithm is allowed access to all of the data stream but can only retain a small amount of information (a.k.a. *sketch* or summary) about the data items it has seen so far. A streaming algorithm is considered efficient if the size of the *sketch* is bounded by $O(\text{polylog}(n))$ and the update time for any data item is also bounded by $O(\text{polylog}(n))$. Thus, a streaming algorithm that seeks to estimate parameters of the data stream is a measurement method that has restrictions on its available computational resources. Streaming measurement methods are represented by the schematic in Figure 5.1. In contrast to Figure 1.1 (in Chapter 1) it is clear that the “sampling” and “summary” steps of Figure 1.1 are combined in Figure 5.1. This happens because combining both steps eliminates the need to store the raw sampled data, thus, saving memory.

DESCRIBE SET. SUBSET OF WHAT? This chapter presents an efficient streaming algorithm to compute the *subset size histogram*. The subset size histogram, λ_i , is the number of disjoint subsets with $i = 1, \dots, w$ elements, where w is the maximum subset size. The subset size histogram is an important metric to detect anomalies in computer networks, such as the Internet. For instance, the flow size distribution seen in Chapter 4 is a normalized subset size histogram, where subsets are flows. In what follows we restrict our attention to the problem of estimating the flow size histogram. However, the following techniques are applicable to the more general problem of es-

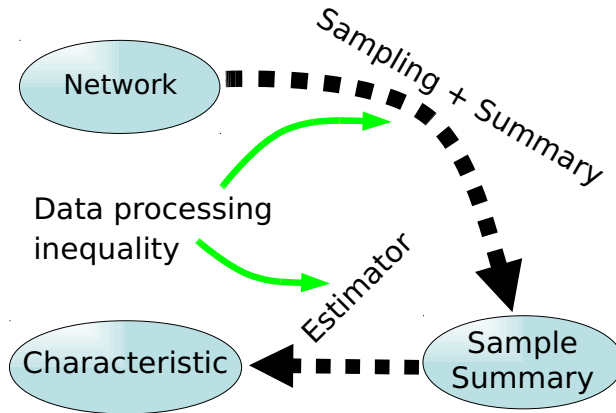


Figure 5.1. Schematics of a measurement method.

estimating the subset size histogram. In what follows *histogram* refers to the flow size histogram. Lets first consider a naive algorithm to compute the histogram.

5.2 A naive algorithm

A simple algorithm to compute the histogram counts the number of packets on each flow. We can accomplish that using hash tables. A hash table is an array of counters indexed by a hash value h . Let H be a hash function such that all packets in the same flow hash to the same hash value. Once a packet arrives, we compute its hash value, h , and increment counter h in the hash table. Ideally H should map each possible flow to a different counter. However, a simple (and fast) function H would allow packets from distinct flows to hash into the same counter [14]. This means that the hash table needs to resolve hash collisions (pairs of different keys with the same hash values). Hash collisions are resolved by assigning a flow identifier to each counter, in order to guarantee that no two flows increment the same counter. A newly arrived packet that is assigned to a counter with value zero increments it and associates its unique flow identifier with the counter. A newly arrived packet that is assigned to a non-zero counter either: (1) increments the counter if its unique flow identifier matches the counter flow identifier; or (2) triggers a collision event if its flow

identifier does not match the counter flow identifier. Collision events can be expensive to resolve, both in terms of the memory and the processing power required. One of the reasons is that unique flow identifiers can be large, which means that storing one for each counter may increase the memory requirements tremendously. As many as half a billion flows arrive every hour at an Internet core router, the above algorithm is not practical.

5.3 An algorithm without collision resolution

The naive algorithm can be made faster and more memory efficient if hash collisions are not resolved. The idea behind this new algorithm, motivated by [40], is that although a counter value may count the number of packets of two or more flows, the original flow size distribution can still be resolved using an estimator. In a streaming algorithm it is important that the whole measurement method, from updating the sketch to outputting the estimates, is fast and memory efficient. Thus, applying the framework developed in Chapter 4 to the statistical information stored in the sketch motivates the development of a new measurement method that: (1) includes a CPU efficient estimator, and (2) requires a small memory footprint.

The algorithm presented in this chapter breaks the packet stream down into measurement epochs. Each epoch has two phases: (1) at each packet arrival the sketch is updated; (2) at the end of the epoch the algorithm uses an estimator to compute the histogram from the sketch. Let $\lambda = \{\lambda_i\}_{i=1}^w$ denote the histogram, where w is the maximum flow size, of flows that arrived during the measurement epoch. The sketch consists of packet counters $\{C_1, \dots, C_m\}$, where m is the total number of counters in the sketch. At each packet arrival the algorithm increments C_h , where $h \in \{1, \dots, m\}$ is the value returned by a universal hash function, H , such that all packets in the same flow hash to the same value. An universal hash function H is a function that has the following property [14]: if x and y are packets from different

flows, $H(x)$ and $H(y)$ are independent and uniformly distributed random variables over $\{1, \dots, m\}$. The algorithm presented here assumes that the total number of flows of size $i = 1, 2, \dots$ observed during a measurement epoch is Poisson distributed with parameter λ_i . This assumption holds true for the Internet traces used in this thesis and has been reported true for other Internet traces [6]. Also note that this is a fairly weak assumption. The streaming algorithm presented in this chapter relies on the analysis of the Fisher information of the sketch. The following property of the counters simplifies the search for the Fisher information.

5.3.1 Counter independence

Let $C_h^{(i)}$ denote the total number of flows of size i that are assigned to counter h by the hash function (in the end of the measurement epoch). Let b denote the number of bits of a counter and thus, $k_{max} = 2^b - 1$ is the maximum value that the counter can assume. The value of counter h at the end of a measurement epoch is a random variable C_h , where

$$C_h = \min(C_h^{(1)} + 2C_h^{(2)} + 3C_h^{(3)} + \dots + wC_h^{(w)}, k_{max}).$$

Under the assumption that the total number of flows of size i is Poisson distributed and that H is an universal hash function we have the following lemma.

Lemma 5.3.1. *The sequence of counters $\{C_h^{(i)}; h = 1, \dots, m\}$ is a sequence of independent Poisson random variables each with parameter λ_i/m .*

Proof. Let N be the number of flows of size i that arrive during the measurement epoch. We know that N is a Poisson random variable with parameter λ_i . Let $X_h^{(i)}(n)$ be a random variable that is equal to 1 if the n -th flow of size i is assigned to counter h , otherwise $X_h^{(i)}(n) = 0$. Because h is obtained from an universal hash function we know that each flow is assigned to a counter independently and with probability

$1/m$. Note that $\sum_{h=1}^m X_h^{(i)}(n) = 1$ and, as each flow in N is assigned to a counter independently and with probability $1/m$, $X_h^{(i)}(n)$ is a Bernoulli random variable that is independent of N and has parameter $p = 1/m$. The total number of flows of size i assigned to counter h can be described by the sum

$$C_h^{(i)} = \sum_{n=1}^N X_h^{(i)}(n).$$

The probability-generating function (p.g.f.) of N is $f_N(s) = E[s^N] = e^{\lambda_i(s-1)}$ and the p.g.f. of each $X_h^{(i)}(n)$ is $E[s^{X_h^{(i)}(n)}] = 1 - p + ps$, where $p = 1/m$. As

$$E[s^{X_h^{(i)}(1)+\dots+X_h^{(i)}(N)}] = E[(1 - p + ps)^N] = e^{\lambda_i(1-p+ps-1)} = e^{\lambda_i p(s-1)},$$

the p.g.f. of $C_h^{(i)}$ is $e^{\lambda_i p(s-1)}$ which is the p.g.f. of a Poisson random variable with parameter $\lambda_i p$. This proves that $C_h^{(i)}$ is Poisson distributed with parameter λ_i/m . It is left to prove that the random variables in the sequence $\{C_h^{(i)}; h = 1, \dots, m\}$ are mutually independent. Thus, I need to show that

$$P \left[C_k^{(i)} \mid \bigcap_{\forall h \neq k} C_h^{(i)} \right] = P \left[C_k^{(i)} \right], \forall k.$$

Let $k \in \{1, \dots, m\}$,

$$\begin{aligned} & P \left[C_k^{(i)} = c_k \mid \bigcap_{\forall h \neq k} C_h^{(i)} = c_h \right] = \\ &= \sum_{n=0}^{\infty} P \left[C_k^{(i)} = c_k \mid \bigcap_{\forall h \neq k} C_h^{(i)} = c_h \cap N = n \right] P \left[N = n, \mid \bigcap_{\forall h \neq k} C_h^{(i)} = c_h \right] \\ &= \sum_{n=0}^{\infty} \mathbf{1}(c_k = n - \sum_{\forall h \neq k} c_h) P \left[N = n \mid \bigcap_{\forall h \neq k} C_h^{(i)} = c_h \right] \\ &= P \left[\sum_{h=1}^m C_h^{(i)} = c_k + \sum_{\forall h \neq k} c_h \mid \bigcap_{\forall h \neq k} C_h^{(i)} = c_h \right] = P[C_k^{(i)} = c_k], \quad (\text{as } N = \sum_{h=1}^m C_h^{(i)}). \end{aligned}$$

□

5.3.2 The likelihood function

In what follows we obtain the likelihood function of C_h in terms of parameters $\lambda = \{\lambda_i; i = 1, \dots, w\}$. Assume that $w > k_{max}$. The likelihood function for $C_h = 0$

$$P[C_h = 0 | \lambda] = e^{-\sum_{\forall j} \lambda_j/m}.$$

The likelihood function for $0 < C_h < k_{max}$ is

$$P[C_h = k | \lambda] = \sum_{\forall (x_1+2x_2+\dots+wx_w=k)} \prod_{i=1}^k \frac{(\lambda_i/m)^{x_i}}{x_i!} e^{-\sum_{\forall j} \lambda_j/m}, \quad (5.1)$$

where the outermost summation sums over all combinations of flow sizes that add up to k (this takes $O(k^3)$ time and can be computed using the recursion presented in Appendix A.2.2). The likelihood function for $C_h = k_{max}$ is

$$P[C_h = k_{max} | \lambda] = 1 - \sum_{k=0}^{k_{max}-1} P[C_h = k | \lambda]. \quad (5.2)$$

5.3.3 The Fisher information

To obtain the Fisher information matrix we need the derivatives of the likelihood function. The derivative with respect to λ_i of the likelihood function for counter value $C_h = k$, where $0 \leq k < k_{max}$ is:

$$\frac{\partial P[C_h = k | \lambda]}{\partial \lambda_i} = \sum_{\forall (x_1+2x_2+\dots+wx_w=k)} \frac{x_i - \lambda_i/m}{\lambda_i} \prod_{m=1}^k \frac{(\lambda_j/m)^{x_m}}{x_m!} e^{-\sum_{\forall j} \lambda_j/m}, \quad (5.3)$$

where, by definition, $0! = 1$. If the counter value is $k = k_{max}$, the derivative of the likelihood function is

$$\frac{\partial P[C_h = k_{max} | \lambda]}{\partial \lambda_i} = - \sum_{k=0}^{k_{max}-1} \frac{\partial P[C_h = k | \lambda]}{\partial \lambda_i}, \quad (5.4)$$

Note that a counter C_h can be no greater than k_{max} .

Applying Lemma 5.3.1 (that counters are independent) and Lemma 2.3.1 (that the Fisher information of a set of n independent samples is n times the Fisher information of one sample) we get that the Fisher information of the whole sketch is m times the Fisher information of a single counter. The Fisher information matrix of a single counter is

$$J_{ji} = \left[\left(\sum_{k=0}^{k_{max}} \frac{(\partial P[C_h = k | \lambda] / \partial \lambda_i) (\partial P[C_h = k | \lambda] / \partial \lambda_j)}{P[C_h = k | \lambda]} \right)_{ij} \right].$$

The following example illustrates the computation of the Fisher information matrix. Suppose $k_{max} = 2$. The derivative of the likelihood function for $C_h = 0$

$$\frac{\partial P[C_h = 0 | \lambda]}{\partial \lambda_i} = -\frac{1}{m} e^{-\sum_{\forall j} \lambda_j / m},$$

For $C_h = 1$:

$$\frac{\partial P[C_h = 1 | \lambda]}{\partial \lambda_1} = (1/m - \lambda_1/m^2) e^{-\sum_{\forall j} \lambda_j / m},$$

and

$$\frac{\partial P[C_h = 1 | \lambda]}{\partial \lambda_i} = -(\lambda_1/m^2) e^{-\sum_{\forall j} \lambda_j / m}, \forall i > 1$$

And for $C_h = k_{max} = 2$:

$$\frac{\partial P[C_h = 2 | \lambda]}{\partial \lambda_1} = -\left(-\frac{1}{m} e^{-\sum_{\forall j} \lambda_j / m} + (1/m - \lambda_1/m^2) e^{-\sum_{\forall j} \lambda_j / m}\right) = \lambda_1 e^{-\sum_{\forall j} \lambda_j / m},$$

and

$$\frac{\partial P[C_h = 2 | \lambda]}{\partial \lambda_i} = -(1/m + \lambda_1/m^2) e^{-\sum_{\forall j} \lambda_j / m}, \forall i > 1.$$

Assembling the Fisher information matrix from the above equations is trivial.

5.3.4 The Cramér-Rao inequality

Let $\hat{\lambda} = \{\hat{\lambda}_i\}_{i=1}^w$ be an estimate of $\lambda = \{\lambda_i\}_{i=1}^w$. In order to simplify our analysis lets assume that $\lambda_i \in \mathbb{R}$, $i = 1, \dots, w$. Estimating λ using a maximum likelihood

estimator, such as the expectation maximization (EM) algorithm, requires multiple iterations over the estimates $\hat{\lambda}$ in order to maximize the likelihood function. This is because changing an estimate $\hat{\lambda}_i$, while keeping the sketch values fixed, forces the estimator to revise all the other estimates $\hat{\lambda}_j, \forall j \neq i$. These changes are clear in the following example. Suppose we seek to estimate λ_1 . Note that counters with value 2 are the result of the collision of two flows of size one or only one flow of size 2. Suppose that inside this iterative estimator we set $\hat{\lambda}_2$ to be zero. The estimate $\hat{\lambda}_1$ must be revised as the number of counters of value 2 has not changed and we now have $\hat{\lambda}_2 = 0$. Likewise, a change in $\hat{\lambda}_1$ most likely translates into a change in $\hat{\lambda}_2$ for the same reason. Thus, it is natural to understand why Kumar et al. [40] resort to a computationally expensive EM algorithm (based on eq.(5.1)) to compute the MLE. The computationally expensive EM is, however, ill-suited as an estimator for a streaming algorithm. In this section we seek to design a computationally faster estimator from the analysis of the inverse of the Fisher information matrix.

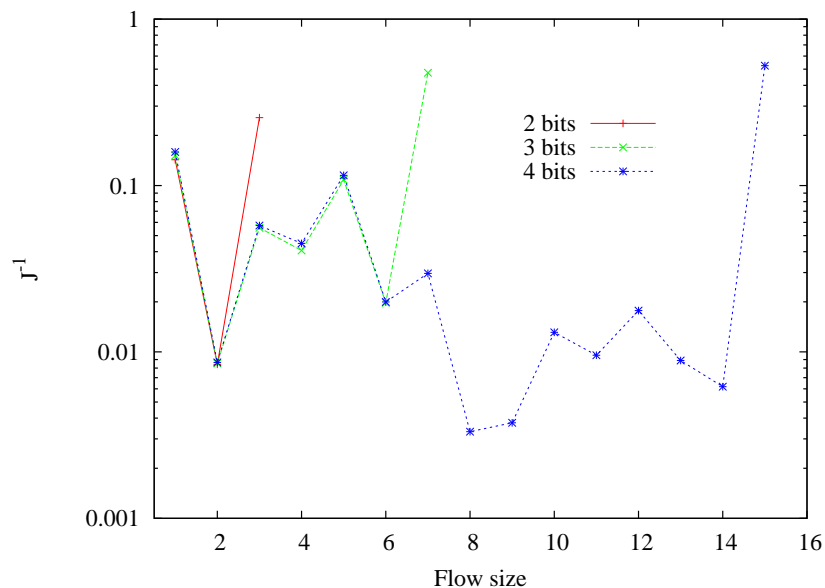


Figure 5.2. The figure plots the inverse Fisher information with varying maximum counter values ($k_{max} = 2^b - 1$).

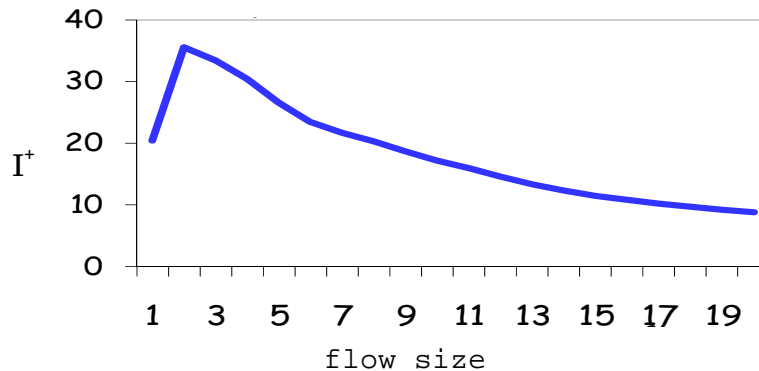


Figure 5.3. Accuracy of the best packet sampling estimator. Inverse Fisher information $(\mathcal{I}^+)_{ii}$ (i is the flow size) with TCP SYN and TCP SEQ protocol information.

In what follows we consider the Fisher information of a single counter, i.e., $m = 1$. In this experiment, λ_i is the fraction of flows of size i in the BBEast-2 trace (presented in Section 4.6.1). This means that $\sum_{\forall i} \lambda_i = 1$, i.e., there is, on average, one flow hashed into the counter. Figure 5.2 plots the inverse of the Fisher information for three different maximum counter values: $k_{max} = 2^b - 1$, where $b \in \{2, 3, 4\}$ bits. It is interesting to compare the information content of the sketch with the information content obtained from packet sampling, i.e., compare the result in Figure 5.2 with Figure 5.3 (the best result of Chapter 4). Note that the sketch needs two orders of magnitude fewer samples (counters) than the packet sampling schemes presented in Chapter 4 need flows in order to achieve the same MSE (mean squared error).

Using Figure 5.2 we can assess how much information counter values greater than k give about flows of size k . Figure 5.2 shows that the information content about λ_i from a counter with maximum value k_{max} is practically independent of the value of k_{max} as long as $i < k_{max}$. This implies that an estimator of, say, λ_1 can aggregate all counter values and flow sizes greater than 1 with little loss of information. The above remark motivates the following estimator.

5.3.5 A fast estimator

Let $m \geq 1$ and $g = (g_0, \dots, g_{k_{max}})$, where g_i is the number of counters with value i . We seek to estimate λ_i , where $i < k_{max}$. Because the number of flows of size i hashed to a counter are i.i.d. Poisson random variables with parameter λ_i/m (Lemma 5.3.1), the average number of counters with value zero is

$$E[g_0] = m e^{-\sum_{i=1}^{\infty} \lambda_i/m}, \quad (5.5)$$

where $E[g_0]$ is the expected value of g_0 . The next equation derives the average number of counters with value one

$$E[g_1] = \frac{\lambda_1}{m} E[g_0]. \quad (5.6)$$

More generally, for $2 \leq j < k$, we have

$$E[g_j] = \left(\frac{\lambda_j}{m} + \sum_{c=2}^{j-1} f_{\lambda}(j, c) \right) E[g_0], \quad (5.7)$$

where function f_{λ} gives the probability that c flows are hashed into the same counter and their sizes sum up to j ; a recursive $O(j^3)$ time algorithm to compute f_{λ} is given in Appendix A.2.2.

Using equations (5.5) and (5.6) we get an estimate for λ_1 :

$$\hat{\lambda}_1 = m(g_1/g_0).$$

Using $\hat{\lambda}_1$ and g_2 we can also estimate λ_2 . More generally, we can estimate λ_j using equation (5.7) and $\hat{\lambda}_i$ for $i = 1, \dots, j - 1$

$$\hat{\lambda}_j = m(g_j/g_0) - m \sum_{c=2}^{j-1} f_{\hat{\lambda}}(j, c). \quad (5.8)$$

Thus, estimating all flow sizes with size less than i takes $O(i^3)$ operations if intermediate results are saved. These estimates are quite accurate as we will observe in the next section.

5.4 Further improvements: reducing the memory footprint

This section provides further improvements to the baseline measurement method presented in Section 5.3. A histogram should provide a general overview of the traffic, rather than a fine grained view. Thus, typical monitoring applications do not need fine grained counts of all flow sizes. In security event detection, we are interested in very small flow sizes (mice) such as 1, 2, but only up to a certain value k . To measure the impact of medium and elephants flows, these larger flow sizes can be estimated in a binned fashion. Therefore, I propose a new multi-resolution algorithm to estimate the size histogram with aggregated and probabilistic counting of large flows, and fine-grained counting for flow sizes up to k packets. As an example, let $k = 16$. In this case we maintain per flow counters for each flow sizes 1,...,16, but flows of sizes from 17 packets to 32 packets, 33 to 64, etc, are counted probabilistically and estimated together. Using this approach 6 bit counters (and $k = 16$) are enough to (probabilistically) count flows of sizes up to $w \approx 10^{14}$.

The estimator presented in Section 5.3.5 is extended to include histogram bins proposed above while producing accurate histogram estimates. This is achieved by designing a space efficient low collision sketch. The sketch divides and folds (multiplexes) Z virtual sketches into the physical space of one sketch. The low flow collision probability allows the estimator to obtain accurate histogram values with $O(k^3 + \log w)$ operations in total. Note that w and k can be made as small as the network operator wants with no loss in accuracy. When faced with very high speed links and relatively low computing resources for monitoring and statistics gathering,

this simple resource minimalist design makes a strong case for an in-line inside the router implementation.

The rest of this chapter is organized as follows. Section 5.5 provides an overview of the algorithm. Section 5.6 illustrates data structure design and the estimator in details. Experiment results using trace data are shown in Section 5.7. Section 5.8 presents the related work. Finally, I conclude with Section 5.9.

5.5 Overview of the measurement method

The data structure of the proposed sketch works as follows. Each newly arrived packet is used to update a counter through a hash function, where all packets in a flow hash to the same counter. The algorithm keeps M of such counters in a vector (*sketch*). A *universal hash function* is a function where a randomized algorithm is used to generate hash values for distinct flows. Hence, the collision probability of different flows hashed to the same counter is a simple function of the number of flows divided by the size of the sketch. The algorithm increases counter values probabilistically, using a variation of the approach in Morris [51]. Counters are incremented by 1 with probability 1 if the counter value, C , is less than k (a small constant defined by the network operator). Otherwise, if $C \geq k$ the counter is incremented with probability 2^{-C+k-1} . This translates into grouping medium to large flow sizes into histogram bins $B_m = [k + 2^m - 1, k + 2^{m+1} - 1]$, $m = 0, \dots, \log_2 w - 1$, where w is the largest flow size as defined by the network operator. As a result of this binning, we only need to use small counters (the experiments in Section 5.7 use 7 bit counters) as compared to other schemes, drastically lowering memory requirements. Note that this approach is performed entirely in software and does not require specialized hardware. In Section 5.6 I explain the details and also present a practical and efficient way to emulate a probabilistic counter without resorting to (slow) pseudo-random number generators.

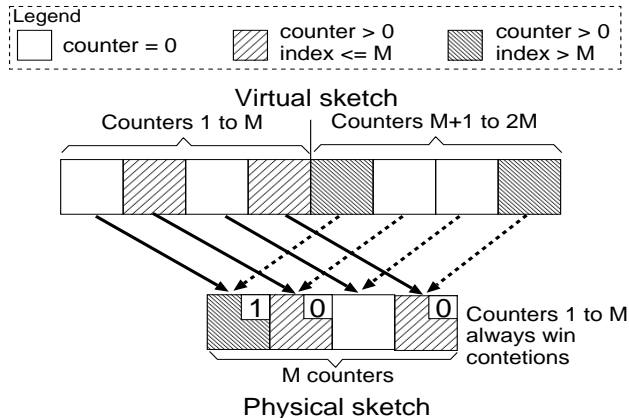


Figure 5.4. Multiplexing a sketch. One extra bit is used to store ownership in the physical sketch. Note that counters with index $\leq M$ in the virtual sketch always win contentions against counters with index $> M$.

At each sketch update the algorithm also updates a histogram of the sketch values. In the estimation phase, the estimator uses this sketch value histogram to estimate the size histogram. To estimate a flow of size i we need at least $O(i^3)$ operations to untangle the corresponding hash collisions. This results in a prohibitively high CPU usage when maximum flow sizes are large, $w \gg 1$. In a sketch with very low collision rate the untangling of hash collisions does not play a significant role in the estimation phase. Moreover, the accuracy of the estimates depends on the fraction of hash collisions, a low collision rate means high estimation accuracy. One way to reduce collisions is to increase the sketch size. However, a large sketch size is not desirable due to the corresponding increase in memory requirements.

Therefore, we arrive at the sketch design exemplified in Figure 5.4. The idea is to “multiplex” Z virtual sketches with M counters each into the physical space of one M -counter sketch. The “multiplexing” works as follows: Z virtual sketch counters share the same physical sketch counter if both counters are zero. If one or both virtual counters are not zero we use the contention resolution algorithm described in Section 5.6.1.1. The contention resolution has an overhead of $\lceil \log Z \rceil$ bits *per* physical counter. Let’s look at an example of contention in Figure 5.4. Counters

with indexes M and $2M$ in the virtual sketch contend for the same physical counter. Virtual counter M wins and virtual counter $2M$ is evicted from the virtual sketch. In Section 5.6.1.1 we see that virtual counter eviction is equivalent to flow thinning. In the experiments shown in Section 5.7 at most 15% of the flows are discarded due to counter eviction. Extra memory space could be used to store these evicted counters if flow thinning must be avoided. It is important to note that the extra CPU overhead using this approach when compared to a regular sketch is negligible.

Small counters with their exponential histogram bin sizes and small hash collision probabilities allow us to propose a $O(k^3 + \log w)$ histogram estimator. Note that k is a small constant typically $k \ll w$ (the experiments in Section 5.7 use $k = 16$ and $w = 10^{14}$). The following details the sketch data structure and its estimator.

5.6 Measurement method description

The measurement method consists of two components, the sketch data structure and the estimator. The following section describes the data structure used to sketch flow sizes based on each packet. Later we present the estimator that obtains the histogram.

5.6.1 Data structures

The sketch data structure consists of three structures:

1. Sketch: This is the main data structure that has M counters, denoted C_h , $h = 1, \dots, M$. Each newly arrived packet increments its corresponding counter. There are also M b -bit *auxiliary* counters labeled $[a_i]$, $i = 1, \dots, M$, plus one ownership bit *per* counter (in the experiments $b = 6$).
2. Sketch histogram: $g = (g_0, \dots, g_{k_{max}})$ is the normalized histogram of the above counter values; it is updated upon changes of the ownership bits.

3. Pseudo-random auxiliary counters: $O(\log w)$ auxiliary counters are kept in order to implement a fast approximate random sampling algorithm.

Here w is the maximum flow size of interest and

$$b \geq \lceil \log_2(\log_2 w + 1 + k) \rceil.$$

In what follows I illustrate the role of each of the above three data structures in the algorithm.

5.6.1.1 Sketch

The sketch is a virtual sketch with ZM counters, $Z \in \{2, 3, \dots\}$, occupying the physical space of a sketch with M counters. I refer to this virtual sketch as a *Z-fold virtual sketch* or just a *virtual sketch* if the value of Z is clear from the context. Counters in the physical sketch are indexed from 1 to M . A counter in the physical sketch is shared by Z virtual sketch counters; each physical sketch counter has $\lceil \log_2 Z \rceil$ ownership bits.

Counters in the virtual sketch are called *virtual counters* or just *counters*. Counters in the physical sketch are called *physical counters*. To simplify the exposition, consider $Z = 2$. Let's follow the example shown in Figure 5.4. Virtual counters with indexes c and $M + c$, $c = 1, \dots, M$, are mapped into the physical counter with index c . A physical counter value represents the value of a virtual counter with index $\leq M$ if its ownership bit is zero. Otherwise it represents a virtual counter with index $> M$. Physical counters are initialized with value zero and with ownership bits set to one. Packets of a flow assigned to a virtual counter with index $> M$ will not change its corresponding physical counter if the physical counter has ownership bit zero. These flows are considered to belong to *evicted virtual counters*. Also, if a packet assigned to a counter with index $\leq M$, arrives and finds its corresponding virtual counter

with ownership bit one, it sets the counter to one and the ownership bit to zero. This means that the previous virtual counter (of index $> M$) that occupies the same physical position is evicted from the virtual sketch. Note that counters are evicted uniformly at random (because the hash function assigns flows to counters randomly); this is equivalent to randomly discarding flows, also called *flow thinning*. In the example of Figure 5.4 virtual counter $2M$ is evicted from the virtual sketch. The following assumes that evicted counters are discarded.

Flow sampling [18] in its simplest form can be seen as a particular case of the above sketch where the number of virtual sketches, Z , goes to infinity, ownership bits are unique flow IDs, and virtual counters can only evict zero-valued counters in the physical sketch. Note that when Z goes to infinity there are no flow collisions in the virtual sketch. However, using a simple flow multiplexing argument, one can show that the amount of flow thinning increases with Z . Thus we want to keep Z as small as possible provided that flow size histogram estimates are accurate. We return to this topic when evaluating this approach in Section 5.7.

5.6.1.2 Sketch histogram

A histogram of the counter values of the virtual sketch is kept in vector $g = (g_0, \dots, g_{k_{max}})$. This vector is initialized with zero except for $g_0 = 2M$. Whenever a counter with value j has its ownership bit changed from one to zero, g_j is decremented by one. This simple operation reflects the reduction in the number of virtual counters due to contention. The remaining histogram updates are quite trivial.

5.6.1.3 Pseudo-random auxiliary counters

The sketch counters perform random (Bernoulli) sampling with probabilities taken from $\{2^{-j} \mid j = 1, \dots, k_{max} - k + 1\}$ for counter values larger than k . At a high level this approach follows the same simple principle of Morris [51], which requires us to perform pseudo-random sampling at line speed. Since traditional pseudo-random

number generators are computationally intensive, there is an alternative that is best-case deterministic and worst-case probabilistic.

Assume there are N i.i.d. (independent and identically distributed) flows that increment their respective hash counters with probability 2^{-j} , $j = 1, \dots, k_{max} - k + 1$. We start by creating an auxiliary counter a_j and initialize it with $a_j \leftarrow 2^j - 1$. Upon a packet arrival (from any of these N flows) a_j is decremented by one. If $a_j = -1$ we sample the packet (i.e. increment the respective sketch counter) and reinitialize $a_j \leftarrow 2^j - 1$. Note that for $N = 1$ this corresponds to deterministic sampling. Since we only need to maintain one additional counter *per* value j we need $O(\log W)$ auxiliary counters for the sketch. Appendix A.2.1 shows that as $N \rightarrow \infty$, packets are sampled randomly (according to a Bernoulli process) at rate 2^{-j} , as if they were sampled by a true random number generator.

In the next section we see how to estimate the histogram using $g = (g_0, \dots, g_{k_{max}})$.

5.6.2 Histogram estimator

This section presents a histogram estimator that uses the empirical sketch histogram g and outputs a flow size histogram in $O(k^3 + \log W)$ operations. Let the sketch *load* define the number of measured flows divided by the virtual sketch size. The estimator works as follows: As soon as either the measurement epoch is reached or the load achieves $L = 1/2$, we save g (which is always up-to-date), reinitialize all variables and start another measurement epoch. I use g to refer to the “saved g ”. g can be used to estimate the size histogram using a two step estimator. Section 5.3.5 presents the first step where we estimate flows of size smaller than k (k is the deterministic counting threshold defined in Section 5.5); Section 5.6.2.1 presents the second step where we estimate the histogram bins for flow sizes $\geq k$. This section shows that there is little information gain if we only seek to estimate flows of size

smaller than i and the estimator has sketch counters that are able to count more than i packets.

To estimate the histogram of flows of size $< k$ we use the estimator presented in Section 5.3.5. Moreover, the sketch load is estimated with $\hat{L} = -\ln(g_0)$ (from equation (5.5)). Thus, estimating the histogram of all flow sizes with sizes less than k takes $O(k^3)$ operations. These estimates are quite precise as we will observe in the next section. The following shows how low sketch loads can help us design a fast estimator for flows of size $\geq k$.

5.6.2.1 Estimates of large flows sizes ($\geq k$)

Estimating large flow sizes encounters a problem: sketch counters are counted probabilistically for values $\geq k$. We can derive an equation similar to eq. (5.7) that accounts for the probabilistic nature of g_j for $j \geq k$

$$E[g_j] = E[g_0] \sum_{i=j}^{\infty} \left(f(i-k, j-k) \lambda_i/m + f(i-k, j-k) \sum_{c=2}^i f_{\lambda}(i, c) \right), \quad (5.9)$$

where function f , described in Appendix A.2.2, is the probability that $i-k$ packets triggers $j-k$ increments on a counter with value k . From equation (5.9) we see that estimating λ_j is not an easy task. In what follows I derive a rough approximation to equation (5.9) that lead to a very simple estimator. Let

$$B_j = \{k + 2^{j-1} - 1, \dots, k + 2^j - 2\}, \quad j = k, \dots, (k_{max} - k) \quad (5.10)$$

be the bins of the histogram for $j \geq k$ and let

$$\Lambda_j = \sum_{\forall i \in B_j} \lambda_i, \quad j = k, \dots, (k_{max} - k)$$

be the total number of flows with size $i \in B_j$. Assume that j is large. In what follows I approximate probabilistic counting by deterministic counting, i.e. $f(i-k, j-k) = 1$

if $i \in B_j$ and zero otherwise. Also assume that most of the flow size distribution probability rests in flows with sizes much smaller than 2^j . In this case, collisions of flows whose flow size sums are in B_j are due to either: (1) a large number of small flows; or (2) few small and large flows. Let's look at the first case, a large number of small flow collisions. The probability of 3 or more flows hashing into the same counter is small when $L = 1/2$. Thus the effect of a large number of flows colliding is negligible as the summation over f_λ in equation (5.9) becomes vanishingly small as m (the number of counters) increases.

Now, consider the second case: a small number of collisions between small and large flows. As j is large, most flows of sizes $i \in B_j$ are at least twice as large as flows of smaller sizes not in B_j . This means that two or three collisions of flows with sizes not in B_j are unlikely to sum up to a size in B_j . Then, apart from degenerate cases such as $\sum_{\forall i \in B_j} \lambda_i \ll \sum_{\forall c \in B_{j-1}} \lambda_c$, most of the collisions between small and large flows that fall into B_j are between small flows and flows whose sizes are in B_j . This motivates us to propose the following approximation to equation (5.9)

$$E[g_j] \approx E[g_0]L\Lambda_j.$$

With the above equation we have the following estimate for Λ_j :

$$\hat{\Lambda}_j \approx (g_j/g_0)/L. \tag{5.11}$$

The following section evaluates the above algorithm using Internet traces and a synthetic hard-to-estimate distribution.

5.7 Evaluation

This section evaluates the proposed algorithm using Internet traces and one synthetic extreme-case distribution. All experiments use parameters: $k = 16$ and

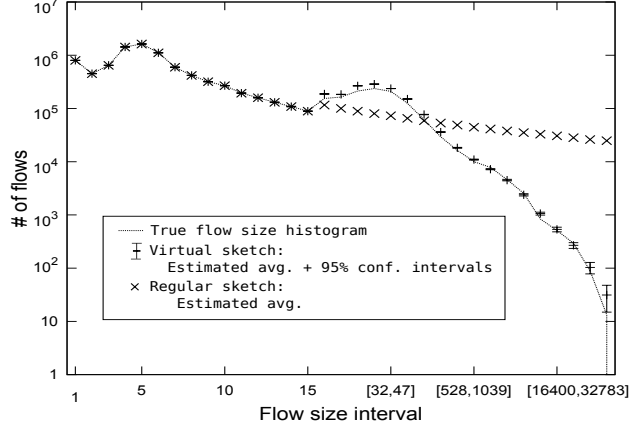


Figure 5.5. Histogram estimates with 8MB of memory. BB-East-2 trace histogram (line) v.s. histogram estimates (with a virtual sketch and with a regular sketch). Experiment: 9.6 million flows (average), 6 bit counters (7 bits *per* flow), 37 runs.

$W \approx 10^{14}$ (thus the sketch counter requires $b = 6$ bits). The first experiment uses the histogram of trace BBEast-2 (described in Chapter 4). This trace contains 9.5 million distinct flows collected over a two hour period. This means that an 8MB physical sketch has a 2-fold virtual sketch load $L \approx 1/2$. In these experiments $Z = 2$ (a 2-fold virtual sketch) as it has a low virtual sketch load, L , and $Z = 2$ is the folding value with the smallest flow thinning probability.

The empirical flow arrival rates are used to generate 37 (Poisson) synthetic traces that will feed the streaming algorithm. Another experiment uses the same scenario replacing the multiplexed sketch by a regular sketch. A regular sketch does not need to maintain an extra ownership bit and can use this space to reduce its load. Measuring the same number of flows the regular sketch has load $L = b/(b+1) = 0.86$, in contrast to the load $L = 1/2$ of a 2-fold virtual sketch. The estimator takes less than one second to compute all estimates in both scenarios. Figure 5.5 show the results of both experiments. The first experiment (with the virtual sketch) also shows the 95% percentile confidence intervals. For the virtual sketch we observe that the algorithm was able to obtain very good histogram estimates as well as very tight confidence intervals. Note that for all flow sizes $< k$ the estimator is unbiased. In the

case where flow sizes are $\geq k$ we see that equation (5.11) provides us with estimates that are fairly close to the actual histogram values. On the other hand, the results of my second experiment (with a regular sketch) indicate that a regular sketch performs poorly for flow sizes greater than k .

A more heavily utilized backbone link BBEast-1 (this trace is described in Chapter 4) contains 250 million measured flows during a 30 minute interval. This trace requires a 220MB sketch and the estimator still takes less than one second to obtain estimates for the complete histogram. The estimates are even more precise than the ones obtained for BBEast-2, due to the order of magnitude increase in the number of counters.

Next I test the estimator with a histogram whose tail decreases exponentially, i.e. $\lambda_i \propto e^{-\alpha i}$. This is a good extreme-case test of the estimator's ability to measure the histogram tail. In this scenario $f_\lambda(i, c) = f_\lambda(i, c')$ for any $c, c' \in \{1, \dots, i\}$, which makes equation (5.11) a much worse approximation to Λ_j than the case where histograms are heavy tailed. Figure 5.6 shows the results of an experiment where $\lambda_i \propto e^{-0.01i}$. We observe that the estimator performs reasonably well for flows of size as large as 32,000 packets. Note that we are also able to capture the overall trend of the tail, although the actual values for very large flow sizes are quite over-estimated. However, this flow size histogram is not based on Internet traces and is presented here to assess the performance of the estimator in a worst-case scenario.

5.8 Related work

Measuring the histogram of network flow sizes has been the subject of a number of studies [15, 19, 39, 40, 57]. Although Internet routers handle traffic on a packet-by-packet basis at the IP layer, the statistics of the underlying flows are vital to network operators. The histogram of flow sizes is an important traffic metric that

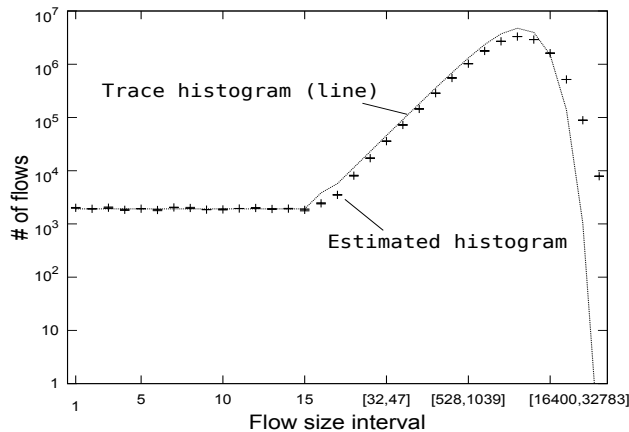


Figure 5.6. Histogram estimates with 16MB of memory. Exponential histogram (line) with $\lambda_i = 2 \times 10^6 e^{-i/10^4} / 9999.5$ v.s. histogram estimates (95% confidence interval is too small). Experiment: 20 million flows (average), 6 bit counters (7 bits *per flow*), 43 runs.

gives insights to network monitoring applications such as traffic profiling, and aids fast detection of security attacks.

In Chapter 4 we saw that reducing the load by aggressively sampling packets at random [19] (without much side information) leads to inaccurate flow size estimates. More recently a number of alternative solutions employ data streaming algorithms to solve the histogram problem (without the need for protocol information) with more accuracy [15, 39, 40]. Kumar et al. [40] proposed sketches to maintain approximate flow size counts in an array of counters.

The algorithm in [40] hashes data items into a sketch with very fast update speeds, $O(1)$. Updates are fast because the algorithm does not resolve (or check) hash collisions. As expected, the speed comes at the expense of loss of information about the original subset sizes (due to unresolved hash collisions). In a (slow) off-line phase, Kumar et al. [40] use a computationally intensive Expectation Maximization (EM) algorithm to compute the estimation from these counters. Moreover, the authors advise network operators to use at least one 32 bit counter *per flow* in average. In this chapter I argue that the EM step in [40] is unnecessary and that estimates can

be obtained with a much faster algorithm that uses much smaller counters. Cormode et al. [15] estimates the histogram using a sketch that maintains a tuple (flow size counter, flow id) in memory. It relies on flow filtering (thinning) to compute histogram estimates. The drawback of this method comes from its high memory requirements, which can be as high as dozens of bytes *per* flow. Lu et al. [46] uses a streaming algorithm tailored specifically to measure heavy tailed distributions that requires little memory space. Their estimator however needs to compute all flow sizes and use (even in a best-case scenario) $O(W)$ time, where W is the largest flow size of interest to the network operator.

5.9 Conclusions & Contributions

This chapter presented a resource minimalist flow size histogram estimator. By trading-off flow size granularity, we count flow sizes in an aggregated and probabilistic manner. This leads to a small memory footprint and an exponential speed up of the time required to compute estimates over previous streaming methods. I tested the proposed algorithm using both Tier-1 backbone traces and synthetic distributions with satisfactory accuracy.

This chapter improves upon the approach of Kumar et al. [40] by exponentially speeding up the estimation phase while reducing the memory requirement from 32 bits *per* flow to 7 bits *per* flow. In contrast to Lu et al. [46], the proposed approach is exponentially faster and does not assume a specific distribution shape. We also see that the amount of Fisher information stored in the sketch with respect to the flow size distribution is two orders of magnitude higher than the Fisher information contained in the sampled packets of Chapter 4. This finding reinforces the reputation of streaming algorithms against traditional sampling approaches. Furthermore, the Fisher information also presents an interesting structure, namely: counter values greater than i have little information that can help estimate flows of size $\leq i$. This

motives the design of a exponentially faster off-line phase and a four fold memory saving.

The proposed streaming algorithm:

- has a small memory footprint: A counter is incremented until the counter reaches value k , where k is a constant. After that, the counter is incremented according to the probabilistic counting scheme proposed by Morris [51]. This reduces the counter size to a few bits (e.g. 7 bits) without harming our ability to precisely estimate flows of size smaller than k ; and
- has a fast estimation phase: Flows are estimated from the smallest to the largest sizes in one pass. Flow sizes larger than k are grouped in exponentially increasing groups. Groups of flow sizes are estimated as a whole.

The complexity of the estimation phase is $O(k^3 + \log w)$. These savings come with a loss of information. However, because the original amount of information is high, we can afford to lose some information.

This chapter also present other contributions that are possibly of interest in their own:

- A hash folding technique that trades-off discarding a fraction of the collided flows for the reduction in the amount of memory required by the sketch. Flow sampling is shown to be a limiting case of this technique.
- A pseudo-random sampling scheme that speeds up the probabilistic counting scheme proposed by Morris [51].

CHAPTER 6

CONCLUSIONS & FUTURE WORK

This thesis provides a framework within which common measurement tasks are analyzed and new, principled, measurement methods are designed. Chapter 3 presented a promising random walk-based method (*Frontier sampling*) that mitigates the estimation errors caused by disconnected or loosely connected graphs. Frontier sampling (FS) uses multiple (m) mutually *dependent* random walker. The dependence between walkers is designed to “better balance” their samples. These samples are shown to be the projection (onto the original graph) of a special type of m -dimensional (single) random walker.

Simulations over real world graphs in Section 3.5 showed that Frontier sampling (FS) is more robust than single and multiple independent random walkers to estimate degree distributions and the fraction of users that belong to popular social groups. An analytical argument (also substantiated by simulations), showed that random walks (in particular, FS) are better suited to estimate the tail of power law graphs than random vertex sampling. Moreover, FS sampling is well suited to be used in large scale (parallel, asynchronous) experiments. The ideas behind FS can have far reaching implications, from estimating characteristics of dynamic networks to the design of new MCMC-based approximation algorithms.

Chapter 4 built upon Fisher’s foundational work to present a framework within which one can design better network measurement methods. Using the Internet flow size distribution as an example, this framework allows us to answer the following (previously open) questions:

- *It possible to accurately estimate the original flow size distribution from the sampled packets?*

Answer: Yes, as long as protocol information is used in the summary function and the estimator.

- *Are there other observable characteristics (besides the number of packets sampled) that could yield more information about the original flow size distribution?*

Answer: Yes, TCP Sequence Numbers contain a lot of information about θ . The SYN flag, on the other hand, contains a small amount of information.

Chapter 4 also presented a measurement method that uses the TCP sequence numbers and is *efficient* even for small sample set sizes.

Chapter 5 presented a resource minimalist flow size histogram estimator. We saw that the amount of Fisher information stored in the sketch with respect to the flow size distribution is two orders of magnitude higher than the Fisher information contained in the sampled packets of Chapter 4. This finding reinforces the reputation of streaming algorithms against traditional sampling approaches. Furthermore, the Fisher information also presents an interesting structure, namely, counter values greater than i contain little information to help estimate flows of size $\leq i$. This motivated the design of a exponentially faster off-line phase and a four fold memory saving. Chapter 5 also presents other contributions that are possibly of interest in their own:

- A hash folding technique that trades-off discarding a fraction of the collided flows for the reduction in the amount of memory required by the sketch. Flow sampling is shown to be a limiting case of this technique.
- A pseudo-random sampling scheme that speeds up the probabilistic counting scheme proposed by Morris [51].

APPENDIX

DESIGNING TCP FLOW-LEVEL ESTIMATORS FROM SAMPLED PACKETS

A.1 An approximation to $h(s_{min}, s_{max})$

Before proceeding to the actual estimation of the flow size distribution we need to address one last issue. Function h introduced in Section 4.6.4 takes as arguments two TCP sequence numbers of two packets in a flow and returns the number of packets sent between these two packets. Before we can estimate flow sizes from real Internet traces we need to approximate h using real Internet sampled flows. We describe this next.

The baseline for our approximation $\tilde{h}(s_1, s_2)$ to $h(s_1, s_2)$ is to use $|s_1 - s_2|$ divided by the maximum data segment transmitted on the flow, where s_1 and s_2 are two TCP sequence numbers of packets belonging to the same flow. The reasoning here is that while a TCP application has enough data to send, most TCP protocol stacks will send packets with data up to the maximum payload size. Most TCP implementations use maximum payload sizes of 1460, 1448 or 536. Notice that we are looking at only one direction of the flow, i.e., we only have access to one side of the two-way TCP connection. Unfortunately a good approximation of h requires enhancements to the baseline approach.

Zero sized packets and modern web browsers present two difficult issues to resolve in finding a good \tilde{h} . (1) Since zero sized packets do not increase the TCP sequence number counter, they are almost totally invisible to us if not sampled. (2) Modern web browsers use persistent HTTP 1.1 connections since an user is expected to follow

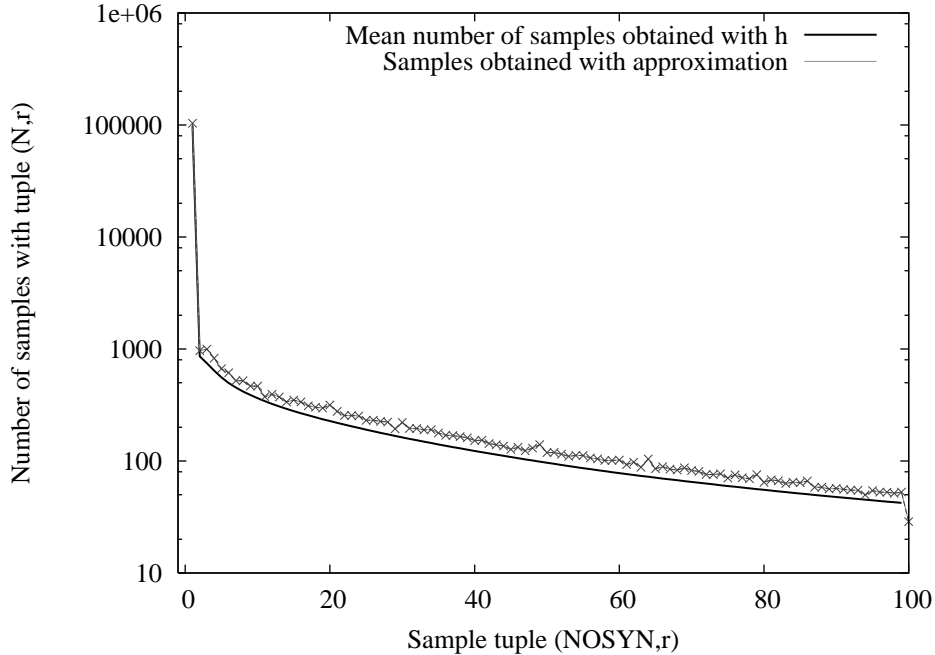


Figure A.1. Number of sampled flows with label (S, r) , $r \geq 1$ obtained from both h drawn synthetically, and \tilde{h} obtained using the real sampled trace. Results from the BB-East-2 trace. Packet sampling rate $p = 0.01$. This graph shows $nd_{(N,r)}$, the number of sample tuples (N, r) (from flows without a SYN sampled packet). Notice that the average is slightly overestimated.

many links on the same web server. Upon receiving a request for a page, the web server sends all packets with the same size *except for the last one*. The user's browser keeps the TCP connection open, and in the event of a new user requested page, it asks for more data over the same TCP connection. This creates a TCP flow from possibly many independent flows. One can argue that these are independent TCP flows and should be treated as such. However, as they share the same SYN packet, our model groups them into a single flow.

We first deal with the multiple payload size problem. A sizable amount of the web-servers on the Internet are Linux machines. Linux machines have an interesting behavior on their IPID field, they are all sequential for a given a TCP flow (a reference to the many uses of the IPID field can be found on [12]). With distinct payload sizes

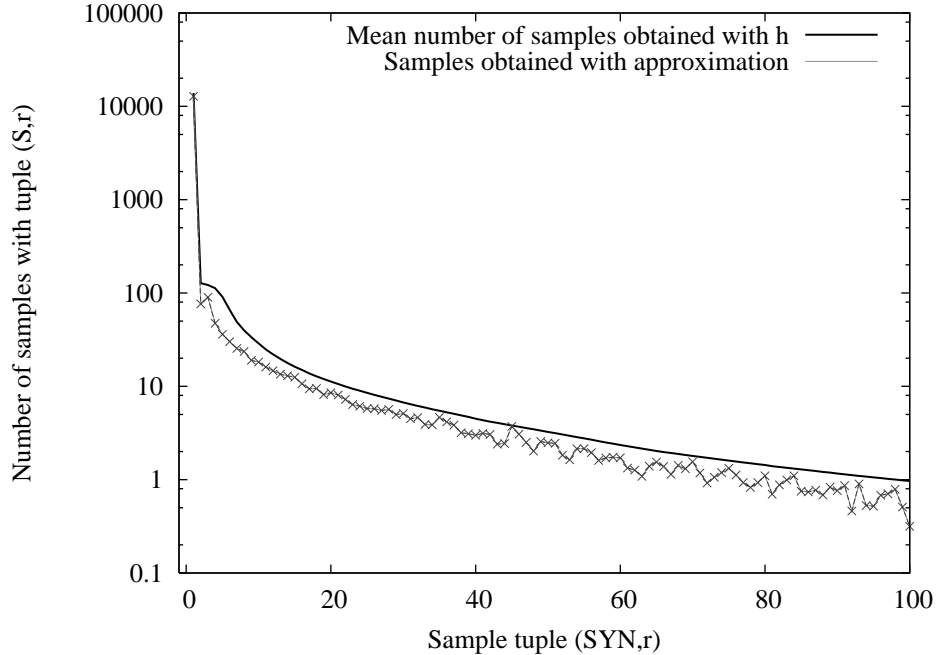


Figure A.2. Number of sampled flows with label (S, r) , $r \geq 1$ obtained from both h drawn synthetically, and \tilde{h} obtained using the real sampled trace. Results from the BB-East-2 trace. Packet sampling rate $p = 0.01$. This graph shows $n\hat{d}_{(S,r)}$, the number of sample tuples (S, r) (from flows with a SYN sampled packet). Notice that the average is slightly underestimated.

inside the same flow, most of them not sampled, $|s_1 - s_2|$ will likely not give us a number that is a multiple of the maximum payload size per packet in the flow. If these small sized payloads are not a large fraction of the total number of packets we can verify whether the number of packets obtained using the IPID difference of the packets is close to the number obtained using Sequence Numbers. If so, we will use the IPID difference.

In most TCP flows the majority of the data is sent in one direction, i.e., the TCP sequence number difference on one direction is much larger than on the other. If most of the data is being sent in the direction being sampled, we obtain maximum payload sizes from the sampled flow, by discarding FIN and SYN packets (usually smaller), assuming sampled packets are representative of the unsampled packets. Otherwise,

we denote the flow as a TCP ACK flow. TCP ACK flows usually have many zero sized packets. One can estimate the value of h on TCP ACK flows by looking at the TCP ACK sequence numbers, which are sequence numbers of the data being sent on the opposite direction of the sampled packets. We keep statistics on the distribution of some specific payload sizes (such as sizes 1460, 536) of non TCP ACK flows and assume that the payload size distribution in both directions is the same. Using the TCP ACK sequence numbers and the above mentioned distribution we obtain an estimate of the value of h .

The above function \tilde{h} is a rather simplistic application of TCP protocol information; however it works reasonably well although the proposed estimator can certainly benefit from a more accurate model of h . We leave the construction of a better model for future research.

The above observations were made from trace Access-East, and then tested on BB-East-2. Sampling flows on the BB-East-2 trace at rate $p = 1/100$ generates, on average, approximately 125,000 sampled TCP flows to be used by the estimator. Figures A.2 and A.1 show how well we can approximate the sample tuples $n\hat{d}_{(S,r)}$ and $n\hat{d}_{(N,r)}$, respectively, obtained from \tilde{h} over real sampled data from BB-East-2. Recall that $n\hat{d}_{(S,r)}$ ($n\hat{d}_{(N,r)}$) are the counts of the sampled SYN (NON-SYN) flows where $r = h(s_{max}^{(u)}, s_{min}^{(u)})$.

Note that the use of \tilde{h} results in a slight underestimate of the number of sampled SYN flows and a slight overestimate of the number of sampled NON-SYN flows. This matter needs further investigation but it might indicate that sampled flows are suffering from flow splitting [37]. A future research topic is to account for flow splitting in the model.

A.2 Designing a streaming algorithm

A.2.1 Pseudo-random counting

Here is the proof that when N , the number of flows using pseudo-random counter c_h , approaches infinity then packets of any of these flows are sampled according to a Bernoulli process. Initialize $c_h \leftarrow h - 1$, remembering that c_h counts down to zero. When c_h is decremented, p is the probability that a given flow decremented it and $1 - p$ be the probability that other flows decremented it. Let X be the number of subtractions by a flow between c_h rollovers (a rollover is when counter goes instantly from $c_h = 0$ back to $c_h = h$). Define

$$P_m = P[X = m], m = 1, \dots, h - 1.$$

It is easy to see that if we assume that $\binom{k}{j} = 0$ whenever $j > k$ we have

$$\begin{aligned} P_m &= \binom{h-1}{m-1} (1-p)^{h-m} p^m + \\ &\sum_{i=0}^{m-1} \binom{h-1}{i} (1-p)^{h-i} p^i P_{m-i}, \quad m = 1, 2, \dots, h. \end{aligned} \tag{A.1}$$

Now we see that the flow is Bernoulli sampled. Proof by induction on m .

$$P_1 = (1-p)^{h-1} p + (1-p)^h P_1 \rightarrow P_1 = \frac{(1-p)^{h-1} p}{1 - (1-p)^h}$$

as $N \rightarrow \infty$, $p \rightarrow 0$ and then $P_1 = 1/h$.

Induction:

Assume $P_i = (1 - 1/h)^{i-1} 1/h$ for $i = 1, 2, \dots, m - 1$. Then equation (A.1) becomes

$$\begin{aligned} P_m &= (h-1)p P_{m-1} + (1-p)^h P_m + O(p^2) \\ &= (h-1)p(1 - 1/h)^{m-2} 1/h + (1-p)^h P_m + O(p^2), \end{aligned}$$

passing all variables P_m to the left side

$$\begin{aligned} P_m &= ((h-1)p(1-1/h)^{m-2}1/n + O(p^2))/(1-(1-p)^h) \\ &= (1-1/h)^{m-1}1/h + O(p). \end{aligned}$$

As $p \rightarrow 0$, $P_m = (1-1/h)^{m-1}1/h$, which is geometrically distributed and thus the flow is Bernoulli sampled.

A.2.2 Counter increment probability

The probability of having j counter increments out of i packets is given by $f(i, j) = 2^{-(j(j+1)/2)} f'(i-j, j+1)$, where

$$f'(i, j) = \begin{cases} \sum_{m=0}^i (1-2^{-j})^m f'(i-m, j-1) & \text{if } j \geq 2 \\ (1-2^{-1})^i & \text{otherwise.} \end{cases}$$

A.2.3 Flow collision function

Function $f_\lambda(j, m) = f'_\lambda(j, 1, m)$ can be computed using the following recursion.

$$f'_\lambda(j, w, m) = \begin{cases} e^{-\sum_i \lambda_i} & \text{if } j = m = 0 \\ 0 & \text{if } w > j \text{ or } w m > j \end{cases}$$

otherwise

$$f'_\lambda(j, w, m) = \sum_{r=0}^{\min(\lfloor j/w \rfloor, m)} \frac{(\lambda_w)^r}{r!} f'_\lambda(j-rw, w+1, m-r);$$

caching its intermediate results f_λ is known to have complexity $O(j^3)$ [40].

BIBLIOGRAPHY

- [1] Achlioptas, Dimitris, Clauset, Aaron, Kempe, David, and Moore, Cristopher. On the bias of traceroute sampling: Or, power-law degree distributions in regular graphs. *J. ACM* 56, 4 (2009), 1–28.
- [2] Avin, Chen, and Krishnamachari, Bhaskar. The power of choice in random walks: An empirical study. *Comput. Netw.* 52, 1 (2008), 44–60.
- [3] Balister, P.N., Bollobás, B., and Stacey, A.M. Dependent percolation in two dimensions. *Probability Theory and Related Fields* 117, 4 (2000), 495–513.
- [4] Bar-Yossef, Ziv, and Gurevich, Maxim. Random sampling from a search engine’s index. *J. ACM* 55, 5 (2008), 1–74.
- [5] Barabási, A.-L., and Albert, R. Emergence of scaling in random networks. *Science* 286 (1999), 509–512.
- [6] Barakat, Chadi, Thiran, Patrick, Iannaccone, Gianluca, Diot, Christophe, and Owezarski, Philippe. Modeling internet backbone traffic at the flow level. *IEEE Transactions on Signal Processing* 51, 8 (August 2003), 2111–2124.
- [7] Bisnik, Nabhendra, and Abouzeid, Alhussein A. Optimizing random walk search algorithms in p2p networks. *Computer Networks* 51, 6 (2007), 1499–1514.
- [8] Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., and Hwang, D.-U. Complex networks: Structure and dynamics. *Physics Reports* 424, 4-5 (2006), 175–308.
- [9] Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. Graph structure in the web. *Computer Networks* 33 (2000), 309–320.
- [10] Cassandras, Christos G., and Lafortune, Stephane. *Introduction to Discrete Event Systems*. Springer-Verlag, Inc., 2006.
- [11] Chen, Fang, Lovász, László, and Pak, Igor. Lifting Markov chains to speed up mixing. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing* (New York, NY, USA, 1999), ACM, pp. 275–281.
- [12] Chen, Weifeng, Huang, Yong, Ribeiro, Bruno F., Suh, Kyoungwon, Zhang, Honggang, de Souza e Silva, Edmundo, Kurose, Jim, and Towsley, Don. Exploiting the IPID field to infer network path and end-system characteristics. In *Proceeding of the 2005 Passive and Active Measurement (PAM'05) Workshop* (March 2005).

- [13] Cho, Junghoo, and Garcia-Molina, Hector. Parallel crawlers. In *WWW '02: Proceedings of the 11th international conference on World Wide Web* (New York, NY, USA, 2002), ACM, pp. 124–135.
- [14] Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., and Stein, Clifford. *Introduction to Algorithms*, 2nd revised edition ed. The MIT Press, September 2001.
- [15] Cormode, Graham, Muthukrishnan, S., and Rozenbaum, Irina. Summarizing and mining inverse distributions on data streams via dynamic inverse sampling. In *VLDB (2005)*, pp. 25–36.
- [16] Cover, Thomas M., and Thomas, Joy A. *Elements of Information Theory*. John Wiley & sons, 1991.
- [17] Datta, Souptik, and Kargupta, Hillol. Uniform data sampling from a peer-to-peer network. In *ICDCS '07: Proceedings of the 27th International Conference on Distributed Computing Systems* (Washington, DC, USA, 2007), IEEE Computer Society, p. 50.
- [18] Duffield, Nick, Lund, Carsten, and Thorup, Mikkel. Flow sampling under hard resource constraints. *SIGMETRICS Perform. Eval. Rev.* 32, 1 (2004), 85–96.
- [19] Duffield, Nick, Lund, Carsten, and Thorup, Mikkel. Estimating flow distributions from sampled flow statistics. *IEEE/ACM Transactions on Networking* 13, 5 (2005), 933–946.
- [20] Eagle, Nathan, Pentland, Alex S., and Lazer, David. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences* 106, 36 (August 2009), 15274–15278.
- [21] Facebook. <http://www.facebook.com>, 2009.
- [22] Faloutsos, Michalis, Faloutsos, Petros, and Faloutsos, Christos. On power-law relationships of the internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication* (New York, NY, USA, 1999), ACM, pp. 251–262.
- [23] Fisher, R. A. *The Design of Experiments*, 1st ed. Oliver & Boyd, Edinburgh, UK, 1935.
- [24] Fisher, R.A. Development of the theory of experimental design. In *Proc. International Statistical Conferences* (1947), vol. 3, pp. 434–439.
- [25] for Internet Data Analysis, Coperative Association. CAIDA’s Internet topology data kit #0304. San Diego Supercomputer Center, UCSD, 2003.
- [26] Fraleigh, C., Moon, S., Lyles, B., Cotton, C., Khan, M., Moll, D., Rockell, R., Seely, T., and Diot, C. Packet-level traffic measurements from the sprint IP backbone. *IEEE Network* (2003).

- [27] Gelman, A., and Rubin, D.B. Inference from iterative simulation using multiple sequences (with discussion). *Statistical Science* 7 (1992), 457–511.
- [28] Geyer, Charles J. Practical Markov Chain Monte Carlo. *Statistical Science* 7, 4 (1992), 473–483.
- [29] Gjoka, Minas, Kurant, Maciej, Butts, Carter T., and Markopoulou, Athina. A walk in Facebook: Uniform sampling of users in online social networks. In *Proc. of the IEEE Infocom* (Jun 2010 (to appear)).
- [30] Gkantsidis, Christos, Mihail, Milena, and Saberi, Amin. Random walks in peer-to-peer networks: algorithms and evaluation. *Perform. Eval.* 63, 3 (March 2006), 241–263.
- [31] Gorman, John D., and Hero, Alfred O. Lower bounds for parametric estimation with constraints. *IEEE Transactions on Information Theory* 36, 6 (Nov 1990), 1285–1301.
- [32] Heckathorn, Douglas D. Respondent-driven sampling: A new approach to the study of hidden populations. *Social Problems* (1997).
- [33] Henzinger, Monika R., Heydon, Allan, Mitzenmacher, Michael, and Najork, Marc. On near-uniform url sampling. In *Proceedings of the 9th international World Wide Web conference on Computer networks : the international journal of computer and telecommunications networking* (Amsterdam, The Netherlands, The Netherlands, 2000), North-Holland Publishing Co., pp. 295–308.
- [34] Hohn, Nicolas, and Veitch, Darryl. Inverting sampled traffic. *IEEE/ACM Trans. Netw.* 14, 1 (2006), 68–80.
- [35] Ibragimov, A., and Hasminskii, R.Z. *Statistical Estimation - Asymptotic Theory*. Springer, New York, 1981.
- [36] Kac, Mark. Random walk and the theory of Brownian motion. *The American Mathematical Monthly* 54, 7 (1947), 369–391.
- [37] Kompella, Ramana Rao, and Estan, Cristian. The power of slicing in internet flow measurement. In *IMC '05: Proceeding of the 5th ACM/USENIX Internet Measurement Conference* (October 2005).
- [38] Konrath, Marlom A., Barcellos, Marinho P., and Mansilha, Rodrigo B. Attacking a swarm with a band of liars: evaluating the impact of attacks on bittorrent. In *P2P '07: Proceedings of the Seventh IEEE International Conference on Peer-to-Peer Computing* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 37–44.
- [39] Kumar, Abhishek, Sung, Minho, Xu, Jun, and Zegura, Ellen W. A data streaming algorithm for estimating subpopulation flow size distribution. In *Proceeding of the ACM SIGMETRICS* (2005), pp. 61–72.

- [40] Kumar, Abhishek, Sung, Minh, Xu, Jun (Jim), and Wang, Jia. Data streaming algorithms for efficient and accurate estimation of flow size distribution. In *Proceeding of the ACM SIGMETRICS* (2004), pp. 177–188.
- [41] Lee, Sang Hoon, Kim, Pan-Jun, and Jeong, Hawoong. Statistical properties of sampled networks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)* 73, 1 (2006), 016102.
- [42] Leskovec, Jure, and Faloutsos, Christos. Sampling from large graphs. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2006), ACM, pp. 631–636.
- [43] Leskovec, Jure, Lang, Kevin J., Dasgupta, Anirban, and Mahoney, Michael W. Statistical properties of community structure in large social and information networks. In *WWW '08: Proceeding of the 17th international conference on World Wide Web* (New York, NY, USA, 2008), ACM, pp. 695–704.
- [44] Levin, David A., Peres, Yuval, and Wilmer, Elizabeth L. *Markov Chains and Mixing Times*. AMS, 2009.
- [45] Lovász, L. Random walks on graphs: a survey. *Combinatorics* 2 (1993), 1–46.
- [46] Lu, Yi, Montanari, Andrea, Prabhakar, Balaji, Dharmapurikar, Sarang, and Kabbani, Abdul. Counter braids: A novel counter architecture for per-flow measurement. In *Proceeding of the ACM SIGMETRICS* (2008).
- [47] Massoulié, Laurent, Le Merrer, Erwan, Kermarrec, Anne-Marie, and Ganesh, Ayalvadi. Peer counting and sampling in overlay networks: random walk methods. In *PODC '06: Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing* (New York, NY, USA, 2006), ACM, pp. 123–132.
- [48] McKnight, Courtney, Jarlais, Don Des, Bramson, Heidi, Tower, Lisa, Abdul-Quader, Abu S., Nemeth, Chris, and Heckathorn, Douglas. Respondent-driven sampling in a study of drug users in New York City: Notes from the field. *Journal of Urban Health* 83, 6 (2006), 154–159.
- [49] Meyn, Sean, and Tweedie, Richard L. *Markov Chains and Stochastic Stability*. Cambridge University Press, New York, NY, USA, 2009.
- [50] Mislove, Alan, Marcon, Massimiliano, Gummadi, Krishna P., Druschel, Peter, and Bhattacharjee, Bobby. Measurement and Analysis of Online Social Networks. In *Proceedings of the 5th ACM/Usenix Internet Measurement Conference (IMC'07)* (San Diego, CA, October 2007).
- [51] Morris, Robert. Counting large numbers of events in small registers. *Communications of the ACM* 21, 10 (1978), 840–842.

- [52] Newman, M. E. J. The structure and function of complex networks. *SIAM Review* 45, 2 (2003), 167–256.
- [53] Press, William H., Flannery, Brian P., Teukolsky, Saul A., and Vetterling, William T. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, October 1992.
- [54] Rasti, Amir H., Torkjazi, Mojtaba, Rejaie, Reza, Duffield, Nick, Willinger, Walter, and Stutzbach, Daniel. Graph sampling in unstructured Peer-to-Peer and undirected online social networks. In *IEEE INFOCOM Mini-Conference* (2009).
- [55] Redner, Richard A., and Walker, Homer F. Mixture Densities, Maximum Likelihood and the EM Algorithm. *SIAM Review* 26, 2 (April 1984), 195–239.
- [56] Ribeiro, Bruno, Gauvin, William, Liu, Benyuan, and Towsley, Don. On MySpace account spans and double Pareto-like distribution of friends. Tech. Rep. UMass CMPSCI UM-CS-2010-001, University of Massachusetts at Amherst, Jan 2010.
- [57] Ribeiro, Bruno, Towsley, Don, Ye, Tao, and Bolot, Jean. Fisher information of sampled packets: an application to flow size estimation. In *Proceedings of the 6th ACM Internet measurement conference* (2006), pp. 15–26.
- [58] Robert, Christian P., and Casella, George. *Monte Carlo Statistical Methods*, 2nd ed. Springer-Verlag, Secaucus, NJ, USA, 2005.
- [59] Rusmevichientong, Paat, Pennock, David M., Lawrence, Steve, and Giles, Lee C. Methods for sampling pages uniformly from the world wide web. In *AAAI Fall Symposium on Using Uncertainty Within Computation* (2001), pp. 121–128.
- [60] Salganik, Matthew, and Goel, Sharad. Respondent-Driven Sampling as Markov Chain Monte Carlo. *Statistics in Medicine* 28, 17 (2009), 2202–2229.
- [61] Stutzbach, Daniel, Rejaie, Reza, Duffield, Nick, Sen, Subhabrata, and Willinger, Walter. On unbiased sampling for unstructured peer-to-peer networks. *IEEE/ACM Trans. Netw.* 17, 2 (2009), 377–390.
- [62] Trees, Harry L. Van. *Detection, Estimation, and Modulation Theory, Part I*, 1st ed. Wiley-Interscience.
- [63] Tune, Paul, and Veitch, Darryl. Towards optimal sampling for flow size estimation. In *IMC '08: Proceedings of the 8th ACM SIGCOMM conference on Internet measurement* (New York, NY, USA, 2008), ACM, pp. 243–256.
- [64] Volz, Erik, and Heckathorn, Douglas D. Probability based estimation theory for Respondent-Driven Sampling. *Journal of Official Statistics* (2008).
- [65] Wang, Chao, and Ma, Xiaoli. Deriving cramer-rao bounds and maximum likelihood estimators for traffic matrix inference. *SIGMETRICS Perform. Eval. Rev.* 37, 2 (2009), 12–14.

- [66] Watts, D.J., and Strogatz, S.H. Collective dynamics of “small world” networks. *Nature* 393 (June 1998), 440–442.
- [67] Yoon, Sooyeon, Lee, Sungmin, Yook, Soon-Hyung, and Kim, Yup. Statistical properties of sampled networks by random walks. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)* 75, 4 (2007), 046114.
- [68] KDD cup 2003 dataset (<http://www.cs.cornell.edu/projects/kddcup/datasets.html>).
- [69] Zamir, Ram. A Proof of the Fisher Information Inequality via a Data Processing Argument. *IEEE Transactions on Information Theory* 44, 3 (1998), 1246–1250.
- [70] Zhong, Ming, and Shen, Kai. Random walk based node sampling in self-organizing networks. *SIGOPS Oper. Syst. Rev.* 40, 3 (2006), 49–55.